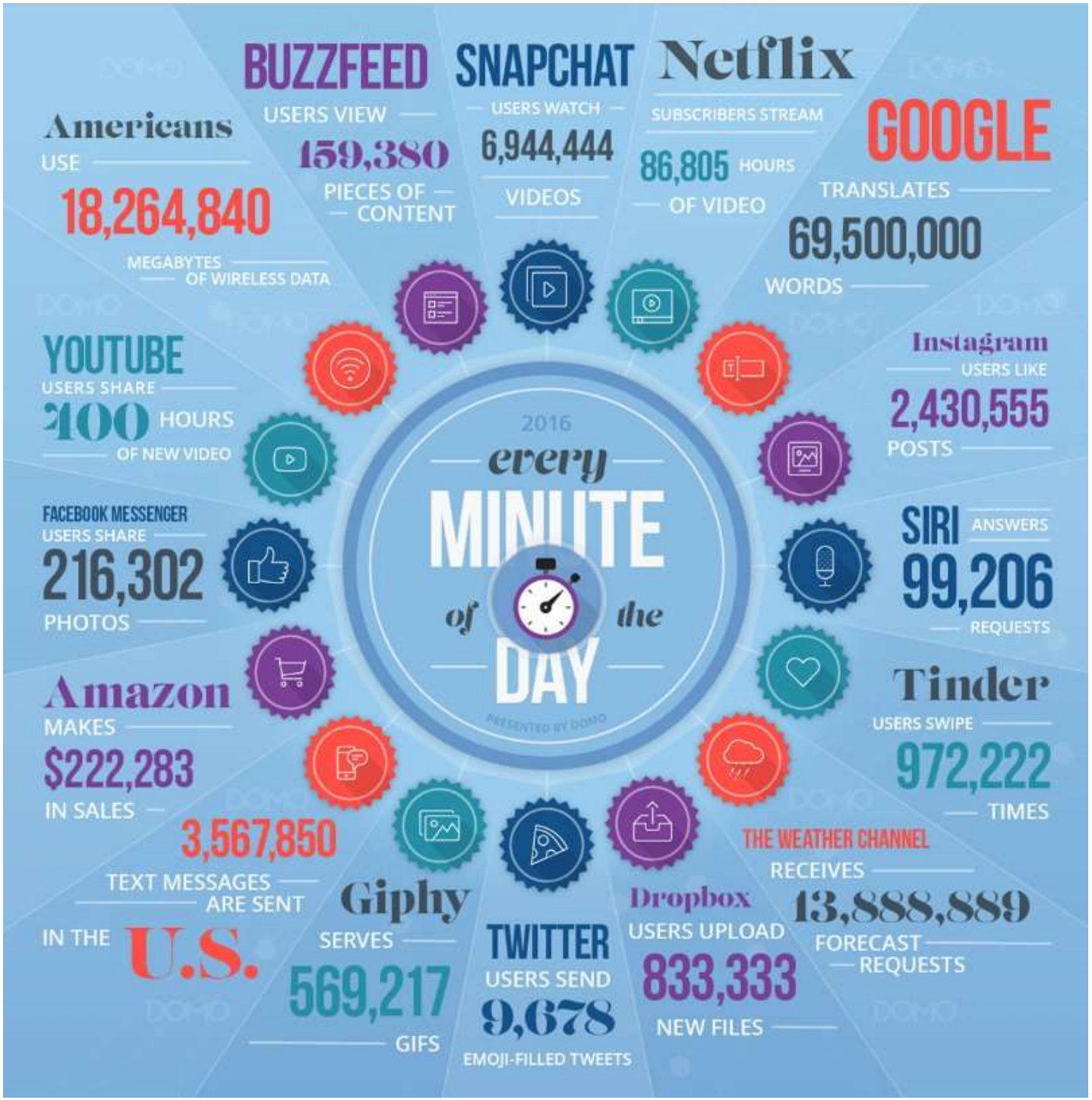
A Quick Intro to Machine Learning

"The field of study that gives computers the ability to learn without being explicitly programmed"


2016
every
MINUTE
of the
DAY
PRESENTED BY DOMO
BUZZFEED
USERS VIEW
159,380
PIECES OF CONTENT
SNAPCHAT
USERS WATCH
6,944,444
VIDEOS
Netflix
SUBSCRIBERS STREAM
86,805 HOURS
OF VIDEO
GOOGLE
TRANSLATES
69,500,000
WORDS
Americans
USE
18,264,840
MEGABYTES
OF WIRELESS DATA
YOUTUBE
USERS SHARE
400 HOURS
OF NEW VIDEO
Instagram
USERS LIKE
2,430,555
POSTS
FACEBOOK MESSENGER
USERS SHARE
216,302
PHOTOS
SIRI ANSWERS
99,206
REQUESTS
Amazon
MAKES
$222,283
IN SALES
Tinder
USERS SWIPE
972,222
TIMES
3,567,850
TEXT MESSAGES
ARE SENT
IN THE U.S.
Giphy
SERVES
569,217
GIFS
TWITTER
USERS SEND
9,678
EMOJI-FILLED TWEETS
Dropbox
USERS UPLOAD
833,333
NEW FILES
THE WEATHER CHANNEL
RECEIVES
13,888,889
FORECAST
REQUESTS

Dati ≠ Informazioni Utili

- Un approccio per cercare di estrarre informazioni utili dalla enorme quantità di dati che viene memorizzata ogni minuto è di usare metodi di

  (Statistical) ***Machine Learning***

- Prima di definire il *Machine Learning* dovremmo chiederci che cos'è o come avviene lo *Human Learning*. Di solito uno studente impara in due modi:
  1. **Memorizzando**: ricordandosi informazioni utili
  2. **Generalizzando**: deducendo nuove informazioni utili da informazioni memorizzate in precedenza

Definizione informale

“The field of study that gives computers the ability to learn without being explicitly programmed”

[VEDERE VIDEO CLIPS]

Definizione di T.M.Mitchell

Definizione [Mitchell97]: Un programma impara dall'esperienza **E** rispetto ad un dato insieme di compiti **T** e una misura di prestazioni **P**, se la sua prestazione per i compiti in **T**, misurata da **P**, migliora con l'esperienza **E**.

Esempio: Un Sistema di guida autonomo

- **COMPITO T**: guidare in autostrada
- **MISURA DI PRESTAZIONE P**: km guidati prima che si verifichi un errore (giudicato da un umano)
- **ESPERIENZA E**: sequenze di immagini e comandi registrati osservando un guidatore umano

Definizione di ML

Definizione [Mitchell97]: Un programma impara dall'esperienza **E** rispetto ad un dato insieme di compiti **T** e una misura di prestazioni **P**, se la sua prestazione per i compiti in **T**, misurata da **P**, migliora con l'esperienza **E**.

Esempio: Un Sistema di riconoscimento dei caratteri

- **COMPITO T**: riconoscere e classificare parole scritte a mano all'interno di immagini
- **MISURA DI PRESTAZIONE P**: percentuale di parole classificate in modo corretto
- **ESPERIENZA E**: un database di immagini contenente parole scritte a mano

Esempio: Cosa possiamo “imparare”?

Dati due insiemi di ***elementi*** (nomi di presidenti):

A = {A. Lincoln, G. Washington, C. de Gaulle}
B = {B. Harrison, J. Madison, L. Napoleon}

E per ogni elemento il ***feature vector*** (vettore delle COVARIATE):

Elemento	Nazionalità	Titolo	Altezza
A. Lincoln	Americano	Presidente	193 cm
G. Washington	Americano	Presidente	189 cm
C. de Gaulle	Francese	Presidente	196 cm
B. Harrison	Americano	Presidente	168 cm
J. Madison	Americano	Presidente	163 cm
L. Napoleon	Francese	Presidente	169 cm

Esempio: Cosa possiamo “imparare”?

Se ora arriva un nuovo elemento:

T. Jefferson: [Americano, Presidente, 189 cm]

In che insieme lo mettiamo? Perchè?

A = {A. Lincoln, G. Washington, C. de Gaulle}
B = {B. Harrison, J. Madison, L. Napoleon}

Elemento	Nazionalità	Titolo	Altezza
A. Lincoln	Americano	Presidente	193 cm
G. Washington	Americano	Presidente	189 cm
C. de Gaulle	Francese	Presidente	196 cm
B. Harrison	Americano	Presidente	168 cm
J. Madison	Americano	Presidente	163 cm
L. Napoleon	Francese	Presidente	169 cm

Supervised vs. Unsupervised

***Supervised Learning*:** Si parte da un insieme di coppie (feature vector, valore) e l'obiettivo è di dedurre una regola che permetta di assegnare un valore ad un nuovo feature vector. Per esempio:

1. **Modelli di regressione** assegnano un numero reale ad ogni feature vector
2. **Modelli di classificazione** associano un etichetta (presa da un dato insieme finite) ad un feature vector

Esempio: Supervised Learning

Dato l'insieme di coppie (feature vector, label):

Elemento	Nazionalità	Titolo	Altezza	LABEL
A. Lincoln	Americano	Presidente	193 cm	**A**
G. Washington	Americano	Presidente	189 cm	**A**
C. de Gaulle	Francese	Presidente	196 cm	**A**
B. Harrison	Americano	Presidente	168 cm	**B**
J. Madison	Americano	Presidente	163 cm	**B**
L. Napoleon	Francese	Presidente	169 cm	**B**

Si deve classificare un nuovo feature vector di cui non si conosce la label:

[T. Jefferson, Americano, Presidente, 189 cm]: **A or B??**

Supervised vs. Unsupervised

Supervised Learning: Si parte da un insieme di coppie (feature vector, valore) e l'obiettivo è di dedurre una regola che permetta di assegnare un valore ad un nuovo feature vector. Per esempio:

1. **Modelli di regressione** assegnano un numero reale ad ogni feature vector
2. **Modelli di classificazione** associano un etichetta (presa da un dato insieme finite) ad un feature vector

Unsupervised Learning: Si parte da un insieme di coppie feature vector (senza label) e l'obiettivo è di dedurre una struttura "latente" nei feature vector. Per esempio:

1. **Modelli di clustering** partizionano gli elementi in gruppi, in modo che gli elementi all'interno dello stesso gruppo siano simili tra loro
2. **Modelli a variabili latente**, che cercando di dedurre valore non osservato in modo diretto, a ciascun feature vector.

Esempio: Unsupervised Learning 1/3

Dato l'insieme di coppie feature vector senza label:

Elemento	Nazionalità	Titolo	Altezza	
A. Lincoln	Americano	Presidente	193 cm	
G. Washington	Americano	Presidente	189 cm	
C. de Gaulle	Francese	Presidente	196 cm	
B. Harrison	Americano	Presidente	168 cm	
J. Madison	Americano	Presidente	163 cm	
L. Napoleon	Francese	Presidente	169 cm	

Come possiamo raggruppare i feature vector in gruppi con elementi simili tra loro e diversi dagli altri?

Esempio: Unsupervised Learning 2/3

Dato l'insieme di coppie feature vector senza label:

Elemento	Nazionalità	Titolo	Altezza	GRUPPO
A. Lincoln	Americano	Presidente	193 cm	**A**
G. Washington	Americano	Presidente	189 cm	**A**
C. de Gaulle	Francese	Presidente	196 cm	**B**
B. Harrison	Americano	Presidente	168 cm	**A**
J. Madison	Americano	Presidente	163 cm	**A**
L. Napoleon	Francese	Presidente	169 cm	**B**

Come possiamo raggruppare i feature vector in gruppi con elementi simili tra loro e diversi dagli altri?

Esempio: Unsupervised Learning 3/3

Dato l'insieme di coppie feature vector senza label:

Elemento	Nazionalità	Titolo	Altezza	GRUPPO
A. Lincoln	Americano	Presidente	193 cm	**A**
G. Washington	Americano	Presidente	189 cm	**A**
C. de Gaulle	Francese	Presidente	196 cm	**A**
B. Harrison	Americano	Presidente	168 cm	**B**
J. Madison	Americano	Presidente	163 cm	**B**
L. Napoleon	Francese	Presidente	169 cm	**B**

Come possiamo raggruppare i feature vector in gruppi con elementi simili tra loro e diversi dagli altri?

Alcuni concetti di base

Signal-to-Noise-Ratio (SNR): rapporto tra informazioni utili (il *segnale*) e dati irrelevanti (il *rumore*)

Dimensionalità: numero di covariate diverse (problema quando maggiore del numero di campioni)

Feature Elimination: metodi per eliminare in automatico le covariate che generano solo *rumore*

Feature Engineering: metodi per individuare le covariate che contribuiscono al *segnale* rispetto a quelle che generano solo rumore. Le covariate possono anche essere calcolate; esempio il Body Mass Index (BMI), ovvero il rapporto tra il peso in kg e il quadrato dell'altezza in metri

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**

Se abbiamo solo informazioni sul cobra, possiamo solo **ricordarci** che il cobra è un rettile

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**

Se ora abbiamo anche le informazioni sul serpente a sonagli potremmo pensare che un animale con queste covariate sia un rettile

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**
Boa	**No**	**Si**	**No**	**Si**	**0**	**?**

Ma come potremmo classificare in automatico (con un algoritmo) un boa date le informazioni del cobra e del serpente a sonagli?

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**
Boa	No	Si	No	Si	0	**SI**

A questo punto saremmo tentati di dire che un animale è un rettile se ha squame, sangue freddo, e zero zampe.

In pratica abbiamo **ELIMINATO** la covariate “ovipari” e “veleno”.

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**
Boa	No	Si	No	Si	0	**SI**
Alligatore	**Si**	**Si**	**No**	**Si**	**4**	**?**
Rana	**Si**	**No**	**Si**	**No**	**4**	**?**
Salmone	**Si**	**Si**	**No**	**Si**	**0**	**?**

E ora come classifichiamo **l'alligatore, la rana e il salmone?**

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**
Boa	No	Si	No	Si	0	**SI**
Alligatore	**Si**	**Si**	**No**	**Si**	**4**	**SI**
Rana	**Si**	**No**	**Si**	**No**	**4**	**NO**
Salmone	**Si**	**Si**	**No**	**Si**	**0**	**NO**
Python	**Si**	**Si**	**No**	**Si**	**0**	**SI**

E se consideriamo **il pitone?**

Ha il vettore delle covariate identico a quello del salmone

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**
Boa	No	Si	No	Si	0	**SI**
Alligatore	Si	Si	No	Si	4	**SI**
Rana	Si	No	Si	No	4	**NO**
Salmone	Si	Si	No	Si	0	**NO**
Python	Si	Si	No	Si	0	**SI**

DOBBIAMO QUINDI ARRENDERCI?

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE	REGOLA
Cobra	Si	Si	Si	Si	0	**SI**	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**	**SI**
Boa	No	Si	No	Si	0	**SI**	**SI**
Alligatore	Si	Si	No	Si	4	**SI**	**SI**
Rana	Si	No	Si	No	4	**NO**	**NO**
Salmone	Si	Si	No	Si	0	**NO**	**SI**
Python	Si	Si	No	Si	0	**SI**	**SI**

POSSIBILE REGOLA: *un animale è un rettile se ha le squame ed è a sangue freddo*.

Esempio: Classificazione di rettili

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE	REGOLA
Cobra	Si	Si	Si	Si	0	**SI**	**SI**
Serpente a sonagli	Si	Si	Si	Si	0	**SI**	**SI**
Boa	No	Si	No	Si	0	**SI**	**SI**
Alligatore	Si	Si	No	Si	4	**SI**	**SI**
Rana	Si	No	Si	No	4	**NO**	**NO**
Salmone	Si	Si	No	Si	0	**NO**	**SI**
Python	Si	Si	No	Si	0	**SI**	**SI**

VALUTAZIONE: *6 valutazioni corrette e una sbagliata:*

5 ***vero positivo****, un* ***vero negativo****, un* ***falso positivo****, nessun* ***falso negativo***

Metriche di distanza

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	Si	Si	Si	Si	0	**SI**
Rana	Si	No	Si	No	4	**NO**

Come possiamo misurare la somiglianza tra due animali?

Metriche di distanza

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	1	1	1	1	0	**SI**
Rana	1	0	1	0	4	**NO**

Come possiamo misurare la somiglianza tra due animali?

Trasformiamo il vettore delle covariate in vettori numerici e utilizziamo la **distanza di Minkowski**:

$$D(x,y) = (\sum_i |x_i - y_i|^p)^{1/p}$$

Casi particolari:

- p=1 distanza di Manhattan
- p=2 distanza euclidea

Metriche di distanza

Nome	Ovipari	Squame	Veleno	Sangue Freddo	Num. Zampe	RETTILE
Cobra	1	1	1	1	0	**SI**
Rana	1	0	1	0	4	**NO**

Come gestire il numero di zampe tra le covariate?

- Trasformo la covariate in
  - 1 ha zampe
  - 0 non ha zampre
- Normalizzo il numero di zampe ad un valore compreso in [0..1]
- Altro …

Modelli di Classificazione

Un applicazione molto commune di Machine Learning è quella di costruire **modelli di classificazione (classifiers)**, usati per determinare se un dato oggetto appartiene ad una delle categorie date. Esempi:

1. **One-class learning**: il training set contiene solo esempi di una singola classe. Il compito è di decidere se nuovi esempi appartengono a quella classe oppure no.
2. **Two-class learning (classificazione binaria)**: il training set contiene esempi di due classi (tipicamente chiamati positivi e negativi), e l'obiettivo è di trovare il confine tra le due classi
3. **Multi-class learning**: il training set contiene diversi classi e si deve trovare il confine tra ciascuna classe

Valutazione di Classifiers

Per valutare un modello di ML si deve dividere il campione di dati in due parti:

1. **TRAINING SET**: usato per “apprendere” un modello (80%)
2. **TEST SET**: usato per “valuare” il modello (20%)

Nel calcolare l’apprendimento di un modello si cerca di **MINIMIZZARE** una misura dell’errore del modello applicato agli elementi del training set, cercando di soddisfare alcuni **VINCOLI**.

Accuracy

$$Accuracy = \frac{veri\ positivi + veri\ negativi}{veri\ pos + veri\ neg + falsi\ pos + falsi\ neg}$$

L'accuratezza è un metodo valido per classificare due classi che sono circa della stessa dimensione (**classi bilanciate**).

In caso di classi sbilanciate possiamo guardare ad altri statistiche.

Sensitivity e Specificity

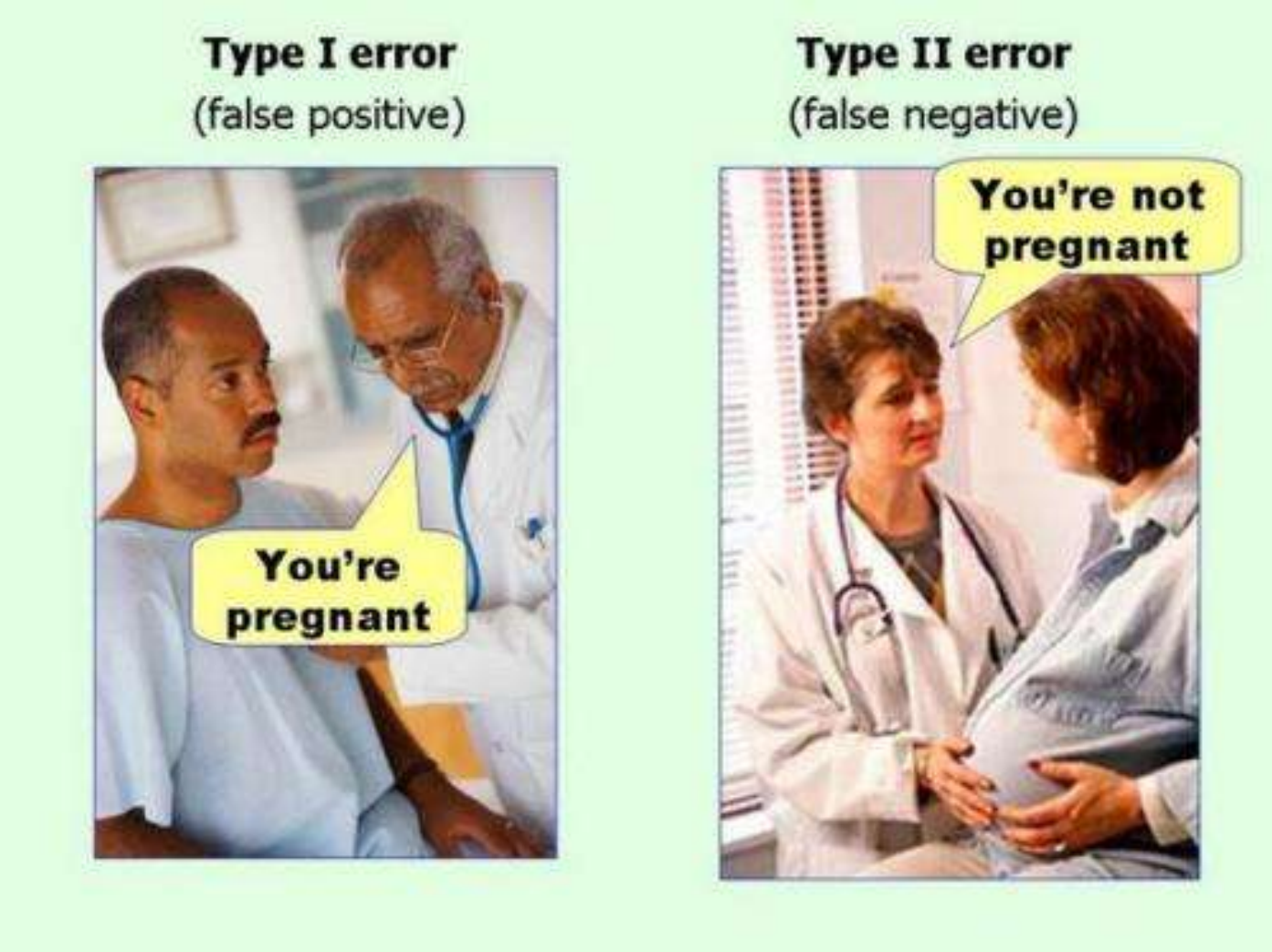
$$Sensitiviy\ (Precisione) = \frac{veri\ positivi}{veri\ positivi\ + falsi\ negativi}$$

$$Specificity = \frac{veri\ negativi}{veri\ negativi\ + falsi\ positivi}$$

$$Positive\ pred.value(Recall) = \frac{veri\ positivi}{veri\ positivi\ + falsi\ positivi}$$

$$Negative\ pred.value = \frac{veri\ negativi}{veri\ negativi\ + falsi\ negativi}$$

Falsi positivi e Falsi negativi

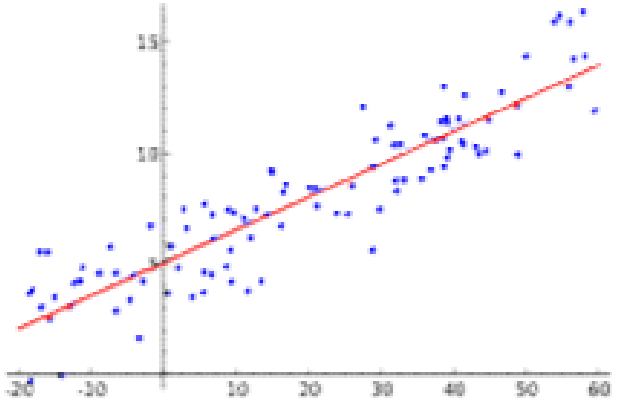


Confusion Matrix

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	**Positive predictive value** = Σ True Positive / Σ Test Outcome Positive
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = Σ True Negative / Σ Test Outcome Negative
		Sensitivity = Σ True Positive / Σ Condition Positive	Specificity = Σ True Negative / Σ Condition Negative	

(Alcuni) Metodi di Classificazione

1. **K-nearest neighbours**: i nuovi esempi sono classificati in base alla loro somiglianza agli esempi presenti nel training sets (si considerano solo i primi “k” più simili, con opportuni pesi).

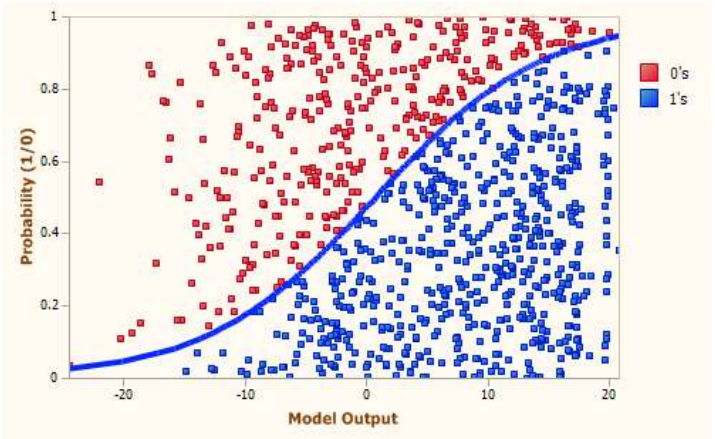


2. **Regressione lineare:**

   y = ax + b

3. **Regressione logistica:**

   y = 1/(1+exp(-ax-b))



Fitting di un Modello 1/3

Supponiamo di aver scelto un modello di regressione lineare. Dobbiamo poter stimare i parametri del nostro modello, ovvero i coefficienti **(a, b)** nella relazione lineare **y = h(x) = ax + b** (h(x) => hypothesis function)

Ovvero, dato un insieme di dati (x_i, y_i) con $i = 1, \ldots, n$ si devono trovare i parametri **(a, b)**, tali per cui è minimo:

$$Risk(h(x)) = E[L((h(x), y)]$$

in cui *L((h(x), y)* è la *loss function* che misura la distanza tra il valore predetto dal modello ***h(x)*** e il valore vero ***y***, mentre ***E[ ]*** indica il valore atteso.

Fitting di un Modello 2/3

Ovvero, dato un insieme di dati (x_i, y_i) con $i = 1,\ldots,n$ si devono trovare i parametri **(a, b)**, tali per cui è minimo:

$$Risk(h(x)) = E[L((h(x), y)]$$

Nella pratica non siamo in grado di calcolare il valore del rischio atteso, ma usiamo un training set per calcolarne una stima (una sua media campionata):

$$EmpiricalRisk(h(x)) = 1/n \sum_{i=1..n} L((h(x_i), y_i)]$$

Fitting di un Modello 3/3

In caso di regressione lineare, dato un insieme di dati (x_i, y_i) con $i = 1,\ldots,n$ (Training Set) si devono trovare i parametri **(a, b)**, tali per cui è minimo:

$$ER(h(x)) = 1/n \sum_{i=1..n} |h(x_i) - y_i|^2 = 1/n \sum_{i=1..n} |ax_i+b - y_i|^2$$

Ovvero, si usano i minimi quadratati come *L(h(x),y)*:

$$\arg\min_{a,b} \{ 1/n \sum_{i=1..n} |ax_i+b - y_i|^2 \}$$

Fitting di un Modello 3/3

In caso di regressione lineare, dato un insieme di dati (x_i, y_i) con $i = 1,\ldots,n$ (Training Set) si devono trovare i parametri **(a, b)**, tali per cui è minimo:

$$ER(h(x)) = 1/n \sum_{i=1..n} |h(x_i) - y_i|^2 = 1/n \sum_{i=1..n} |ax_i+b - y_i|^2$$

Ovvero, si usano i minimi quadratati come $L(h(x),y)$:

$$\arg\min_{a,b} \{ 1/n \sum_{i=1..n} |ax_i+b - y_i|^2 \}$$

NOTA: Il problema di ML viene ridotto ad un problema di OTTIMIZZAZIONE

Valutazione del Modello

Dato un secondo insieme di dati (x_i, y_i) con $i = 1,\ldots,q$ (Testing Set) si valutare i parametri **(a, b)** scelti tali andando a valutare

$$\textit{Mean Squared Error } (h(x)) = 1/n \sum_{i=1..q} |ax_i+b - y_i|^2$$

Altre misura di valutazione solitamente usate:

1. **Mean Absolute Error = MAE = $1/n \sum_{i=1..q} |ax_i+b - y_i|$**
2. **Mean Absolute Percentage Error …**
3. **Coefficient of Determination (link…)**

ML in Python: Scikit-Learn

Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science Interactively at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning	
`>>> lr.fit(X, y)`	Fit the model to the data
`>>> knn.fit(X_train, y_train)`	
`>>> svc.fit(X_train, y_train)`	
Unsupervised Learning	
`>>> k_means.fit(X_train)`	Fit the model to the data
`>>> pca_model = pca.fit_transform(X_train)`	Fit to data, then transform it

Prediction

Supervised Estimators	
`>>> y_pred = svc.predict(np.random.random((2,5)))`	Predict labels
`>>> y_pred = lr.predict(X_test)`	Predict labels
`>>> y_pred = knn.predict_proba(X_test)`	Estimate probability of a label
Unsupervised Estimators	
`>>> y_pred = k_means.predict(X_test)`	Predict labels in clustering algos

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Estimator score method

Metric scoring functions

Classification Report

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Precision, recall, f1-score and support

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

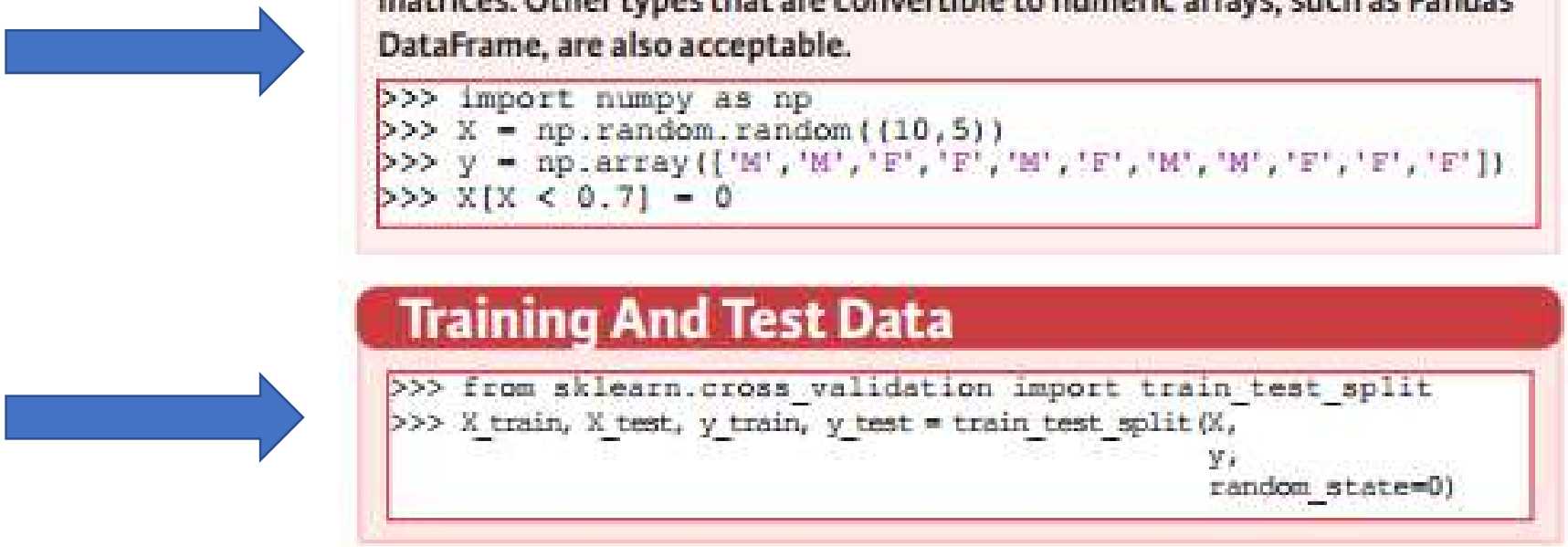
```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

DataCamp
Learn Python for Data Science Interactively

ML in Python: Scikit-Learn



Loading The Data

Also see NumPy & Pandas

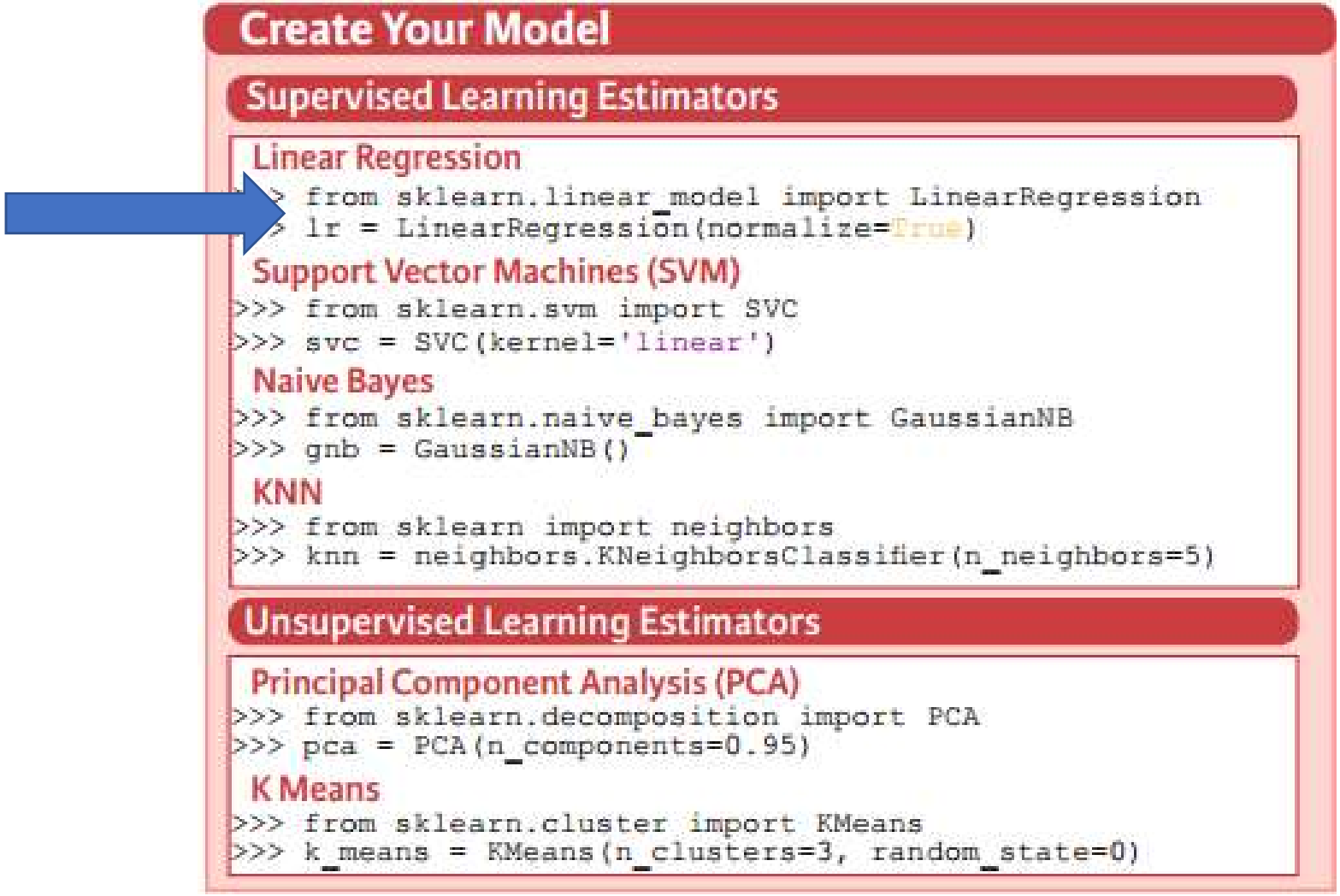
Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

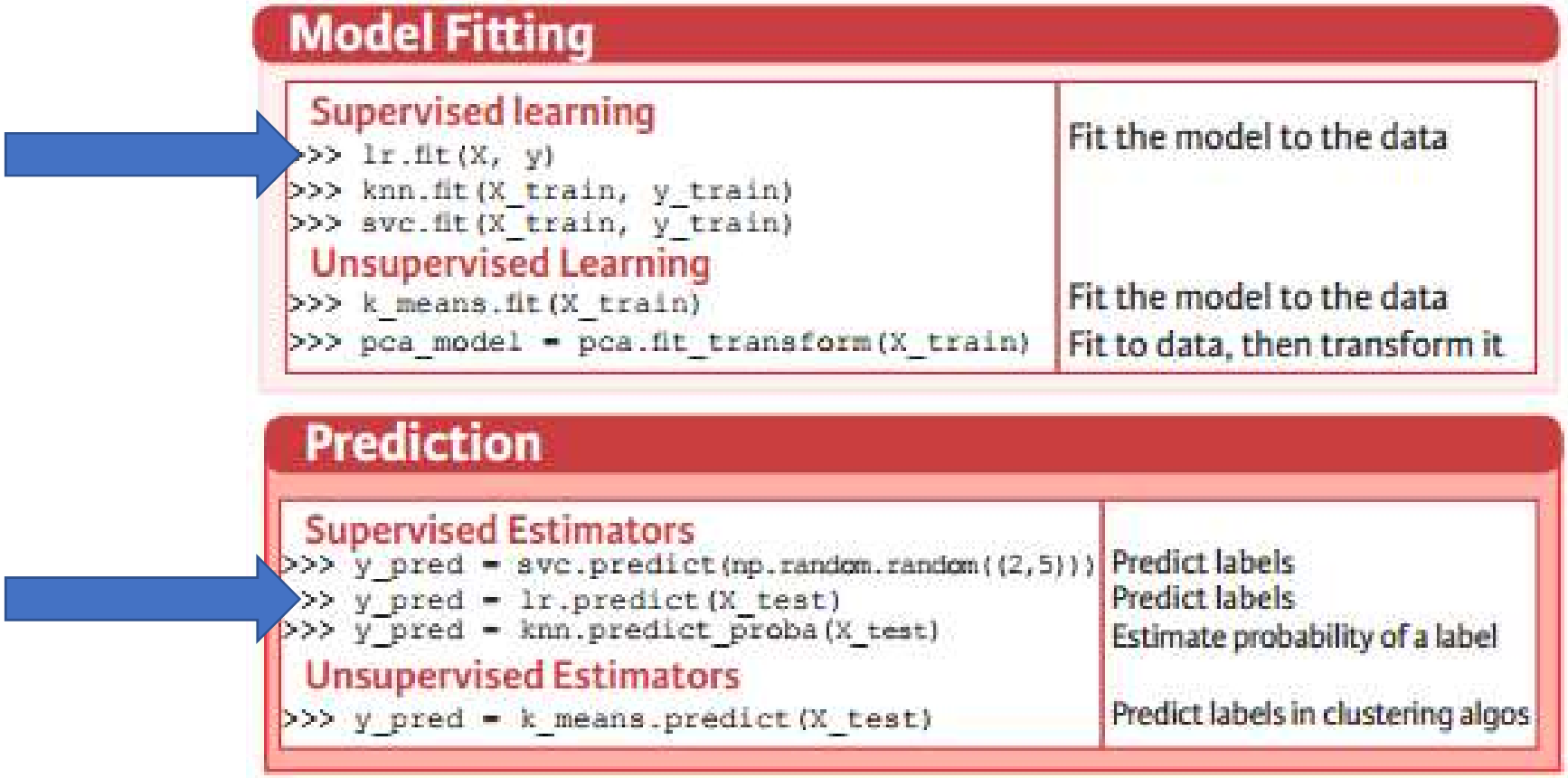
Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

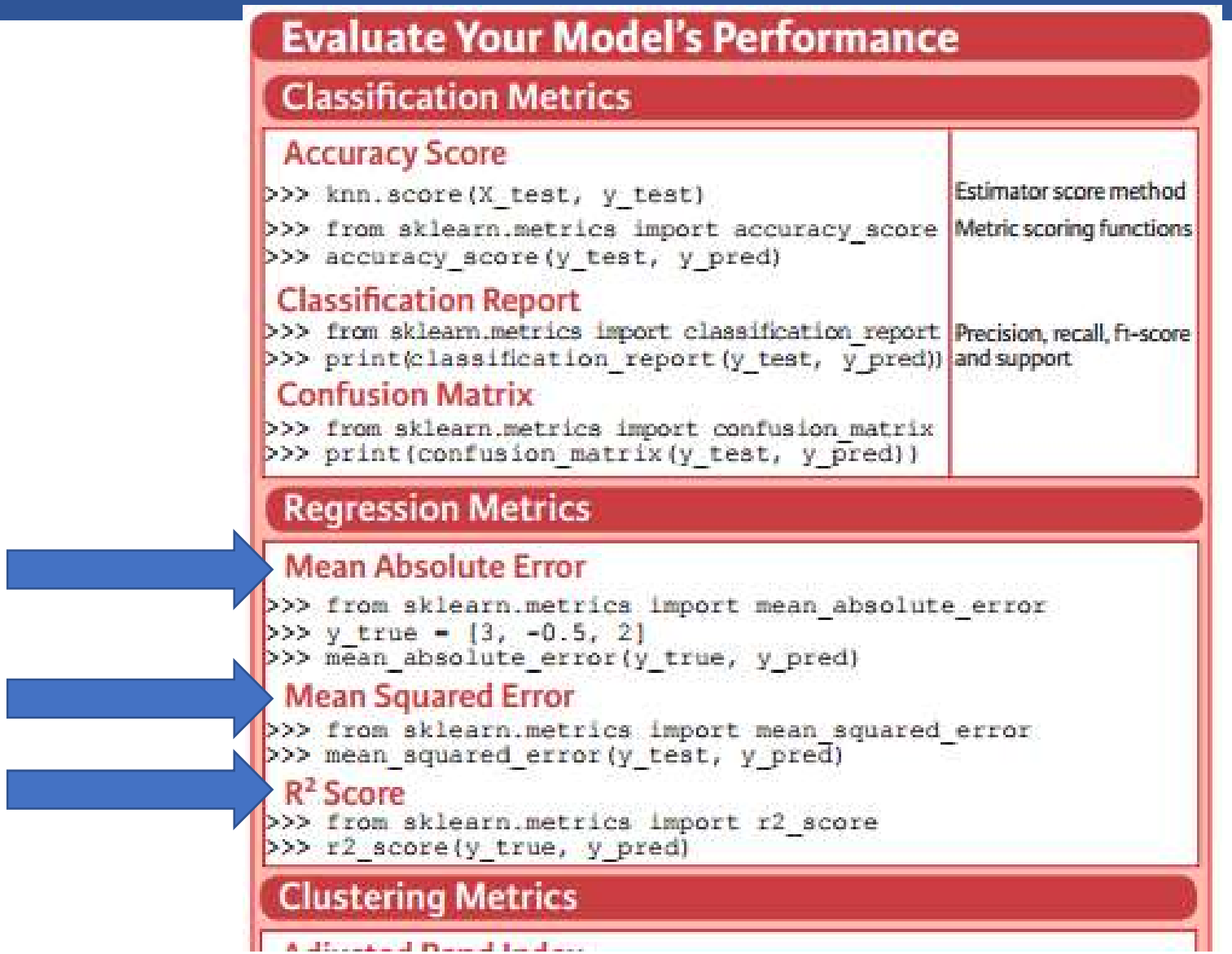
ML in Python: Scikit-Learn



ML in Python: Scikit-Learn



ML in Python: Scikit-Learn



Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test)
```
Estimator score method

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```
Metric scoring functions

Classification Report

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```
Precision, recall, f1-score and support

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics