ITSAR ISTITUTO TECNICO SUPERIORE ANGELO RIZZOLI

Algoritmi per il Machine Learning

Dott. Antonio Giovanni Lezzi

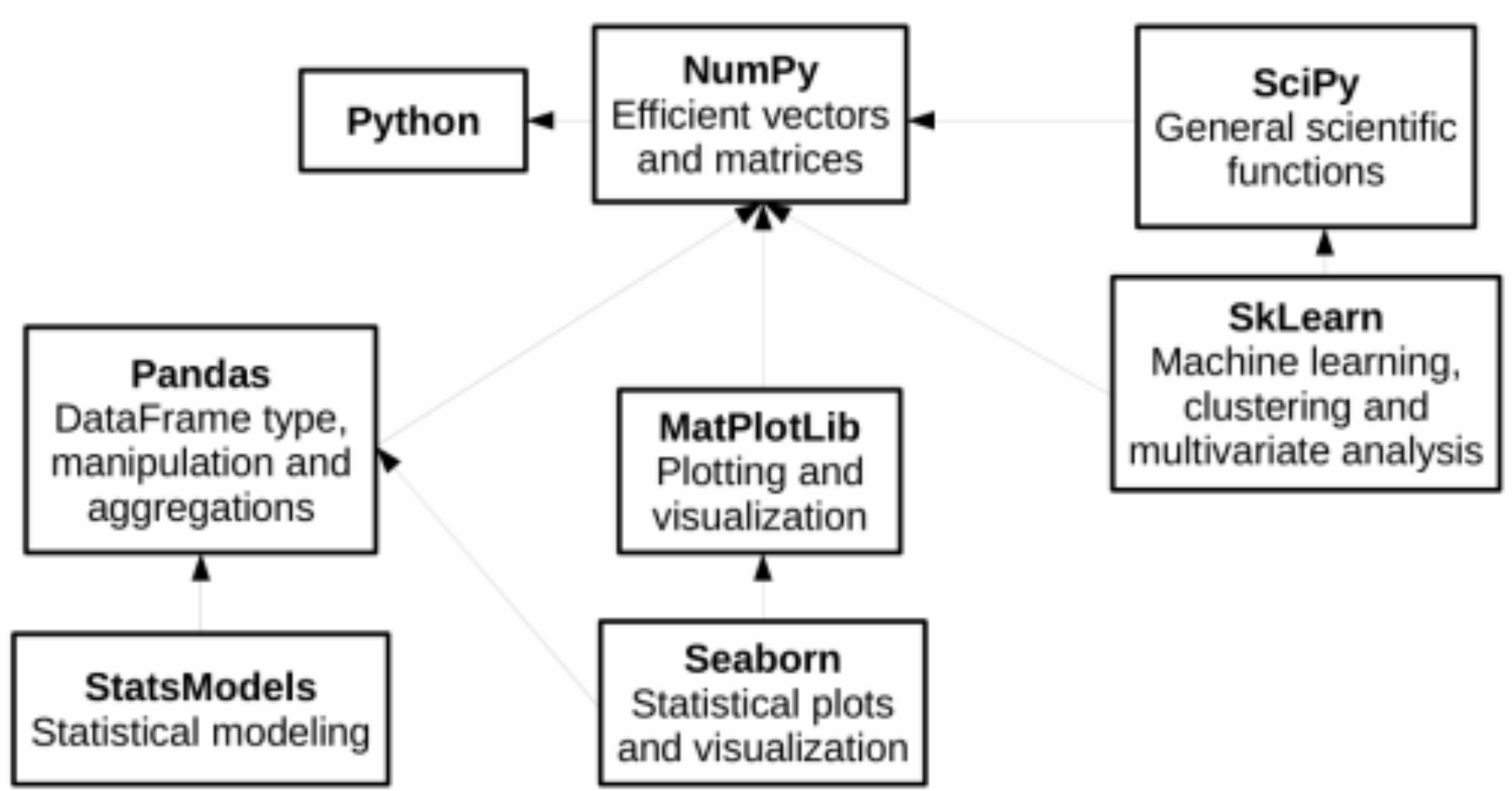
ITSAR

MACHINE LEARNING

Dott. Antonio Giovanni Lezzi

LE LIBRERIE PIÙ USATE IN ML

- Per iniziare occorre aver installato:
- NUMPY
- PANDAS
- SciKit-Learn


Python
NumPy
Efficient vectors and matrices
SciPy
General scientific functions
Pandas
DataFrame type, manipulation and aggregations
MatPlotLib
Plotting and visualization
SkLearn
Machine learning, clustering and multivariate analysis
StatsModels
Statistical modeling
Seaborn
Statistical plots and visualization

ITSAR

ALGORITMI DI MACHINE LEARNING

- **NumPy**
- Pandas
- KNN con SciKit-Learn
- Dimensionamento fattore K

NumPy

Dott. Antonio Giovanni Lezzi

NUMPY

- NumPy è una libreria open source per il linguaggio di programmazione Python
- Aggiunge supporto a grandi matrici e array multidimensionali insieme a una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati.

NUMPY

- Questa libreria offre numerosi vantaggi rispetto al semplice utilizzo di Python tra cui:

  A. definizione della struttura dati nd-array, che permette la creazione di liste, matrici per effettuare calcoli matematici;

  B. l'archiviazione efficiente dei dati attraverso un utilizzo ottimizzato della memoria per la gestione degli nd-array e notevolmente inferiore rispetto agli array definiti in Python;

  C. la possibilità di specificare diversi tipi di dati per gli nd-array che consentono a Numpy di integrarsi con i database;

  D. strumenti per l'integrazione di codice C/C ++ e Fortran;

  E. ampio insieme di funzioni ottimizzate per operazioni aritmetiche, algebra lineare, elaborazione numerica;

  F. la definizione di funzioni ottimizzate di ricerca e ordinamento dei dati;

ITSAR

NUMPY

- NumPy (Numerical Python) è un modulo che estende Python con strutture dati e metodi utili per il calcolo tecnico scientifico.
- In Python puro abbiamo a disposizione:
    1. Oggetti numerici di alto livello: interi, floating point
    2. container: liste (insert ed append a costi bassi) e dizionari (operazioni di lookup veloci)
- NumPy
    3. Introduce un modalità naturale ed efficiente per utilizzare array multi dimensionali
    4. Aggiunge una serie di utili funzioni matematiche di base (algebra lineare, FFT, randomnumber, …)

ITSAR

NUMPY

- Gli array in NumPy sono omogenei, a differenza delle liste di Python, tutti gli elementi sono dello stesso tipo (dtype).
- Gestiscono operazioni con dati di elevate dimensioni.
- Fornisce metodi di accesso, trasformazione, iterazione e inizializzazione di grande efficienza. Generalmente sono più efficienti delle liste convenzionali.
- Si può creare un array direttamente da una lista o da una tupla.

NUMPY

- Un **data type (dtype) object**, istanzanziato tramite la classe numpy.dtype, definisce come devono essere interpretati i byte corrispondenti a un elemento dell'array.

- tipi dei dati: come integer, float, Python Object, ecc.

NUMPY

- il tipo di dato è structured: un tipo di dato creato dall'utente e che può includere array e dtypes, o un aggregato di altri tipi di dati.

- In particolare, i tipi di dati structured vengono formati creando un tipo di dati il cui campo contiene altri tipi di dati. Ogni campo ha un nome con il quale è possibile accedervi.

- Un oggetto dtype può essere costruito utilizzando il costruttore numpy.dtype(). Se volessimo definire un oggetto dtype basato sullo scalare int32 basterà scrivere dt_int32 = np.dtype('int32')

NUMPY

- Poiché molti di questi hanno definizioni dipendenti dalla piattaforma, viene fornito un set di alias a dimensione fissa come ad esempio: numpy.int32, numpy.int64, numpy.float32, numpy.float64 / numpy.float_, numpy.complex64

- Conversione di un int in float

  x = np.float32(1)

  1.0

NUMPY

- Un array viene creato tramite il metodo costruttore array() e vengono forniti due valori: una lista contenente i valori e il tipo di dato

```
>>> a = np.array([1, 4, 5, 8], float)
>>> a
array([ 1., 4., 5., 8.])
>>> type(a)
<type 'numpy.ndarray'>
```

NUMPY

- Uso del metodo range()

```
>>>a = np.array(range(6), float).reshape((2, 3))
>>> a
array([[ 0., 1., 2.], [ 3., 4., 5.]])
```

- Uso del metodo arange()

```
>>> np.arange(5, dtype=float)
array([ 0., 1., 2., 3., 4.])
```

NUMPY

- Un **n-dimensional array (ndarry)** è un tipo di struttura dati perfetta per il calcolo matematico, ottimizzando le performance in termini di costo computazionale e velocità.
- Un ndarry rappresenta un array multidimensionale e omogeneo di elementi di dimensioni fisse e si definisce:
  - il suo ordine di byte;
  - quanti byte occupa in memoria;
  - se è un numero intero, un numero in virgola mobile o altro.
- Possiamo definire più semplicemente un ndarray come un contenitore multidimensionale di elementi dello stesso tipo e dimensione.

ITSAR

NUMPY

- Il numero di dimensioni ed elementi in un array è definito dal suo attributo shape, che è rappresentato tramite una tupla di N numeri interi e non negativi, che specificano le dimensioni di ciascuna dimensione.

```
import numpy as np
matrix = np.array([[1.2, 2.4, 3.1], [4.5, 5.2, 6.9], [7.2, 8.8, .9]], np.float32)
```

- La variabile matrix è quindi un ndarray, come facilmente ricavabile dall'utilizzo della funzione type.

```
type(matrix)
numpy.ndarray
```

- A questo punto, per conoscerne la dimensione basterà utilizzare l'attributo shape come segue.

```
matrix.shape
(3, 3)
```

ITSAR

NUMPY

- Invece, per verificarne il tipo, basterà usare l'attributo dtype.
  matrix.dtype
  dtype('float32')

- Se vogliamo conoscere il numero di elementi e il numero di dimensioni che compongono l'oggetto matrix, possiamo usare l'attributo size e ndim, rispettivamente, ottenendo.
  print(f'matrix dimension: {matrix.ndim} - total elements: {matrix.size}')
  matrix dimension: 2 - total elements: 9

- Quelli visti fin qui sono solo alcuni degli attributi offerti dall'oggetto ndarray, ve ne sono altri come imag o real per ottenere la parte immaginaria o reale dell'array.

- Per maggiori informazioni, si rimanda alla documentazione ufficiale.

NUMPY

- Erroneamente, si tende a pensare che gli ndarray siano simili alle list in Python, ma le differenze sono notevoli. In Python un oggetto list:
  - è definito inserendo uno o più elementi tra parentesi quadre;
  - è ordinato mostrando i dati in un ordine specifico;
  - è mutabile;
  - non deve essere dichiarato;
  - non può gestire direttamente le operazioni matematiche.

NUMPY

- Contrariamente gli ndarray:
  - devono essere dichiarati
  - vengono creati tramite l'apposita funzione array() del modulo numpy;
  - possono definire un solo tipo di dato;
  - possono memorizzare i dati in modo compatto, risultando efficienti per l'archiviazione di grandi quantità di dati;
  - sono ottimi per le operazioni matematiche.

NUMPY

- Ad esempio, si può dividere ogni elemento di un array per lo stesso numero con una sola riga di codice.

```
array = np.array([3, 6, 9, 12])
division = array/3
print(division)
[1. 2. 3. 4.]
print (type(division))
<class 'numpy.ndarray'>
```

- Se provassimo a fare lo stesso con una list, otterremmo un errore.

```
list = [3, 6, 9, 12]
division = list/3
---------------------------------------------------------------------------
TypeError Traceback (most recent call last)
<ipython-input-18-f12723541 4b2> in <module>()
list = [3, 6, 9, 12]
----> 2 division = list/3
TypeError: unsupported operand type(s) for /: 'list' and 'int'
```

ITSAR

ALGORITMI DI MACHINE LEARNING

- NumPy
- **Pandas**
- KNN con SciKit-Learn
- Dimensionamento fattore K

Dott. Antonio Giovanni Lezzi

PANDAS

- Pandas è una libreria software scritta per il linguaggio di programmazione Python per la manipolazione e l'analisi dei dati.
- In particolare, offre strutture dati e operazioni per manipolare tabelle numeriche e serie temporali.
- Il nome deriva dal termine "panel data", termine econometrico per set di dati che include osservazioni su più periodi di tempo per gli stessi individui.

PANDAS

- Il cuore della libreria sono gli oggetti DataFrame, strutture dati 2D indicizzati sia sulle colonne che sulle righe.

- È possibile creare un oggetto DataFrame a partire da zero utilizzando i metodi della libreria numpy:

```
import pandas as pd
import numpy as np

rows = np.arange(3,9)
cols = list('ABCDF')
df = pd.DataFrame(np.random.randn(6,5), index=rows, columns=cols)
print(df)
```

ITSAR

PANDAS

- Dal print dell'oggetto si può notare che è possibile pensare un oggetto della classe DataFrame esattamente come una tabella SQL;
- Inoltre gli indici delle righe e delle colonne possono essere sostanzialmente qualsiasi cosa.
- In questo esempio infatti le righe sono indicizzate con numeri che vanno dal 3 all'8.

ITSAR

PANDAS

- Se si vuole fare riferimento ad una singola colonna basterà utilizzare un'istruzione del tipo:
  column_labelB = df['B']
- se invece si vuole fare riferimento a specifiche righe della tabella l'istruzione diventa:
  rows_label45 = df.loc[4:5]
- con il metodo loc sono utilizzate le etichette delle righe; se si intende estrarre righe a seconda della loro posizione nella tabella, l'istruzione diventa:
  rows_index13 = df.iloc[1:3]
- Con il print di rows_label45 e rows_index13 si nota che il risultato sarà lo stesso.

ITSAR

PANDAS

- La libreria vi permette di caricare un file .csv all'interno di un oggetto DataFrame con la seguente istruzione:
  import pandas as pd
  df = pd.read_csv('table.csv')

- È possibile affettare e filtrare il contenuto del file .csv semplicemente utilizzando i metodi e le istruzioni messe a disposizioni dalla libreria.
  Il metodo mette a disposizione dei parametri opzionali: sep che permette di scegliere il tipo di separatore da utilizzare per il parsing del file, il parametro dtype che vi permette di specificare il tipo delle colonne da caricate nel DataFrame.
  import pandas as pd
  df = pd.read_csv('table.csv', sep=';', dtype={'ID': str})
  # ipotizzando che in 'table.csv' esista un campo 'ID' che deve essere letto come una stringa

ITSAR

PANDAS

- È anche possibile caricare il contenuto di un file Excel all'interno di un oggetto DataFrame con read_excel().

- Si consiglia di visionare le intestazioni degli oggetti DataFrame creati, così da essere certi che il caricamento sia andato a buon fine. Per farlo basterà scrivere:
  print(list(df)) #Il risultato di questa operazione è la lista delle etichette delle colonne.

- Controllare sempre prima il separatore decimale dei floating point utilizzato nei file. Se fosse diverso dal punto, utilizzate il parametro decimal per leggere correttamente il file.

PANDAS

- Una volta creato l'oggetto DataFrame, è possibile modificare una cella della tabella semplicemente facendole riferimento in uno dei modi mostrati nel seguito:

```
import pandas as pd
import numpy as np

rows = np.arange(3,9)
cols = list('ABCDF')
df = pd.DataFrame(np.random.randn(6,5),index=rows,columns=cols)
print(df)
# modifichiamo la cella alla posizione riga '6' colonna 'B'
df.loc[6]['B'] = 9999
# modifichiamo la cella alla posizione riga '3' (prima riga) colonna 'C'
df.iloc[0]['C'] = 9999
print(df)
```

ITSAR

PANDAS

- È possibile impilare due DataFrame con il metodo append:

```
import pandas as pd
import numpy as np
rows = np.arange(3,9)
cols = list('ABCDF')
df1  = pd.DataFrame(np.random.randn(6,5),index=rows,columns=cols)
df2 = pd.DataFrame(np.random.randn(6,5),index=rows,columns=cols)
df = df1.append(df2,ignore_index=True)
print(df)
```

- Il parametro ignore_index impostato a True fa in modo che il nuovo DataFrame abbia dei nuovi indici per le righe e non usi quelli di df1 e df2.
- Se i due DataFrame non sono consistenti con le colonne, il metodo append() aggiungerà NaN nelle colonne non presenti in entrambi i DataFrame.
- Se volete sostituire i NaN con degli 0, vi basterà digitare:

```
df.fillna(0,inplace=True)
```

PANDAS

- Per ottenere informazioni statistiche rapide sul DataFrame appena creato, esistono due funzioni molto interessanti: corr che restituisce la matrice di correlazione di tutte le colonne numeriche del DataFrame, e describe che restituisce informazioni statistiche come media, mediana, minimo, massimo, varianza e altre, per ogni colonna numerica del DataFrame.

```
import pandas as pd
import numpy as np

rows = np.arange(3,9)
cols = list('ABCDF')
print(df.corr())
print(df.describe())
print(df.describe(percentiles=[0.25,0.75]))
# con il parametro 'percentiles' potete aggiungere altre informazioni sui percentili da mostrare
```

ITSAR

PANDAS

- Si può utilizzare la funzione to_csv() per salvare il risultato in un file .csv:
  df.to_csv('table.csv', sep=';', decimal=',', index=False)

- Il parametro decimal serve a specificare il simbolo che sarà utilizzato per dividere i decimali nelle colonne float della tabella. Ricordarsi di aggiungere il parametro quando si deve riaprire la tabella con pandas:
  df = pd.read_csv('table.csv', sep=';', decimal=',')

- Il parametro index se impostato a False fa in modo che non sia aggiunta la colonna degli indici nel file .csv; di default è impostato a True.

ITSAR

ALGORITMI DI MACHINE LEARNING

- NumPy
- Pandas
- **KNN con SciKit-Learn**
- Dimensionamento fattore K

Dott. Antonio Giovanni Lezzi

ITSAR

SCIKIT-LEARN

- Scikit-learn è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python.
- Contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano, ed è progettato per operare con le librerie NumPy e SciPy.

DEFINIRE IL DATASET

- Il primo passo da fare è selezionare il dataset che si vuole analizzare.
- In questo caso utilizziamo un dataset già esistente e famoso nel mondo del machine learning: quello della classificazione dei vini.
- Il dataset lo puoi trovare al seguente link (in pratica è un file csv, separato cioè da virgole).
- Esso è costituito da 13 colonne e 178 istanze.

```
1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050
1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185
1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480
1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735
1,14.2,1.76,2.45,15.2,112,3.27,3.39,.34,1.97,6.75,1.05,2.85,1450
1,14.39,1.87,2.45,14.6,96,2.5,2.52,.3,1.98,5.25,1.02,3.58,1290
1,14.06,2.15,2.61,17.6,121,2.6,2.51,.31,1.25,5.05,1.06,3.58,1295
1,14.83,1.64,2.17,14,97,2.8,2.98,.29,1.98,5.2,1.08,2.85,1045
1,13.86,1.35,2.27,16,98,2.98,3.15,.22,1.85,7.22,1.01,3.55,1045
1,14.1,2.16,2.3,18,105,2.95,3.32,.22,2.38,5.75,1.25,3.17,1510
1,14.12,1.48,2.32,16.8,95,2.2,2.43,.26,1.57,5,1.17,2.82,1280
1,13.75,1.73,2.41,16,89,2.6,2.76,.29,1.81,5.6,1.15,2.9,1320
1,14.75,1.73,2.39,11.4,91,3.1,3.69,.43,2.81,5.4,1.25,2.73,1150
1,14.38,1.87,2.38,12,102,3.3,3.64,.29,2.96,7.5,1.2,3,1547
1,13.63,1.81,2.7,17.2,112,2.85,2.91,.3,1.46,7.3,1.28,2.88,1310
1,14.3,1.92,2.72,20,120,2.8,3.14,.33,1.97,6.2,1.07,2.65,1280
1,13.83,1.57,2.62,20,115,2.95,3.4,.4,1.72,6.6,1.13,2.57,1130
1,14.19,1.59,2.48,16.5,108,3.3,3.93,.32,1.86,8.7,1.23,2.82,1680
1,13.64,3.1,2.56,15.2,116,2.7,3.03,.17,1.66,5.1,.96,3.36,845
1,14.06,1.63,2.28,16,126,3,3.17,.24,2.1,5.65,1.09,3.71,780
1,12.93,3.8,2.65,18.6,102,2.41,2.41,.25,1.98,4.5,1.03,3.52,770
1,13.71,1.86,2.36,16.6,101,2.61,2.88,.27,1.69,3.8,1.11,4,1035
1,12.85,1.6,2.52,17.8,95,2.48,2.37,.26,1.46,3.93,1.09,3.63,1015
1,13.5,1.81,2.61,20,96,2.53,2.61,.28,1.66,3.52,1.12,3.82,845
1,13.05,2.05,3.22,25,124,2.63,2.68,.47,1.92,3.58,1.13,3.2,830
1,13.39,1.77,2.62,16.1,93,2.85,2.94,.34,1.45,4.8,.92,3.22,1195
1,13.3,1.72,2.14,17,94,2.4,2.19,.27,1.35,3.95,1.02,2.77,1285
1,13.87,1.9,2.8,19.4,107,2.95,2.97,.37,1.76,4.5,1.25,3.4,915
1,14.02,1.68,2.21,16,96,2.65,2.33,.26,1.98,4.7,1.04,3.59,1035
1,13.73,1.5,2.7,22.5,101,3,3.25,.29,2.38,5.7,1.19,2.71,1285
1,13.58,1.66,2.36,19.1,106,2.86,3.19,.22,1.95,6.9,1.09,2.88,1515
1,13.68,1.83,2.36,17.2,104,2.42,2.69,.42,1.97,3.84,1.23,2.87,990
1,13.76,1.53,2.7,19.5,132,2.95,2.74,.5,1.35,5.4,1.25,3,1235
1,13.51,1.8,2.65,19,110,2.35,2.53,.29,1.54,4.2,1.1,2.87,1095
1,13.48,1.81,2.41,20.5,100,2.7,2.98,.26,1.86,5.1,1.04,3.47,920
1,13.28,1.64,2.84,15.5,110,2.6,2.68,.34,1.36,4.6,1.09,2.78,880
1,13.05,1.65,2.55,18,98,2.45,2.43,.29,1.44,4.25,1.12,2.51,1105
1,13.07,1.5,2.1,15.5,98,2.4,2.64,.28,1.37,3.7,1.18,2.69,1020
1,14.22,3.99,2.51,13.2,128,3,3.04,.2,2.08,5.1,.89,3.53,760
```

DEFINIRE IL DATASET

- I campi o le intestazioni della tabella sono così definiti:
- **Class**: indica la classe del vino, in particolare nel dataset ne abbiamo 3 (classe 1, classe 2 e classe 3). Rappresenta anche l'output che si vuole ottenere, cioè stabilire a quale di queste classi i nuovi vini appartengono;
- **Alcohol**: indica il grado alcolico del vino (% in volume);
- **Malic acid**: ossia acido malico, uno dei principali acidi organici presenti nelle uve da vino (g / l);
- **Ash**: sono le ceneri, che rappresentano il contenuto delle sostanze minerali presenti in un vino. È un indicatore importante per determinare la qualità del vino (misurato in millisiemens per centimetro, mS/cm);
- **Alcalinity of ash**: ossia l'alcalinità delle ceneri, un parametro che esprime approssimativamente la quantità di acidi organici presenti nel vino sottoforma di sali (pH).

DEFINIRE IL DATASET

- **Magnesium**: indica la quantità di magnesio presente nel vino (g su kg).
- **Total phenols**: indica il numero di fenoli inclusi nel vino, che sono sostanze naturali che danno il colore al vino stesso oltre che a sensazioni gustative (mg/L).
- **Flavanoids**: I flavonoidi sono i polifenoli più abbondanti nel vino. (mg/L);
- **Nonflavanoid phenols**: I composti fenolici conferiscono caratteristiche specifiche al vino e creano anche aromi e sapori specifici quando le interazioni complesse si svolgono durante la fermentazione e la vinificazione (mg / L);
- **Proanthocyanins**: indicano le proantocianidine, un tipo di fenolo antiossidante del vinorosso (mg/L).
- **Color intensity**: ossia una semplice misura di quanto sia scuro il vino;
- **Hue**: è una delle principali proprietà del colore;
- **Proline**: ossia la prolina, un amminoacido (Mg / L).

DEFINIRE IL DATASET

- Le colonne sono 14 perché è stata indicata anche la classe dei vini (1° colonna). Per maggiori informazioni si veda l'analisi eseguita da Github sul vino.

- L'obiettivo che si vuole ottenere è stabilire la precisione che l'algoritmo ci da nel determinare la classe di vino a cui apparterranno nuove istanze, ossia nuovi vini. Più preciso sarà l'algoritmo e maggiormente sarà probabile determinare a che classe futuri vini apparteranno.

IMPOSTIAMO IL PROGRAMMA

- Il metodo Iloc viene definito da due parentesi quadre e due valori separati dalla virgola:

- Il primo termine indica il numero di righe che verranno prese in considerazione (partendo da 0). I due punti indicano tutte le righe del dataset. Sia per X che per y prenderemo tutte le righe del dataset.

- Il secondo termine indica il numero di colonne che verranno prese in considerazione (partendo da 0). Per X consideriamo dalla seconda colonna fino all'ultima. Per y solo la prima colonna.

SUDDIVISIONE DEL DATASET

- Per evitare un adattamento eccessivo, suddivideremo il nostro set di dati in training e test, il che ci dà un'idea migliore di come l'algoritmo è stato eseguito durante la fase di test. In questo modo l'algoritmo viene testato su dati non visti, come sarebbe in un'applicazione di produzione.

- Per creare la suddivisione tra fase di allenamento (training) e test, esegui il seguente script:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

- Prima di tutto importiamo la classe **train_test_split** dalla libreria **sklearn.model_selection**. Poi suddividiamo con la seconda riga di codice il set di dati in 80% di dati di allenamento e 20% di dati di test.

- Ciò significa che su un totale di 178 record, il set di allenamento conterrà 142 record e il set di test contiene 36 di quei record.

ADATTAMENTO DATI

- Prima di effettuare previsioni effettive, è sempre buona norma ridimensionare le caratteristiche in modo che tutte possano essere valutate in modo uniforme.

- Pertanto occorre lanciare il seguente script:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

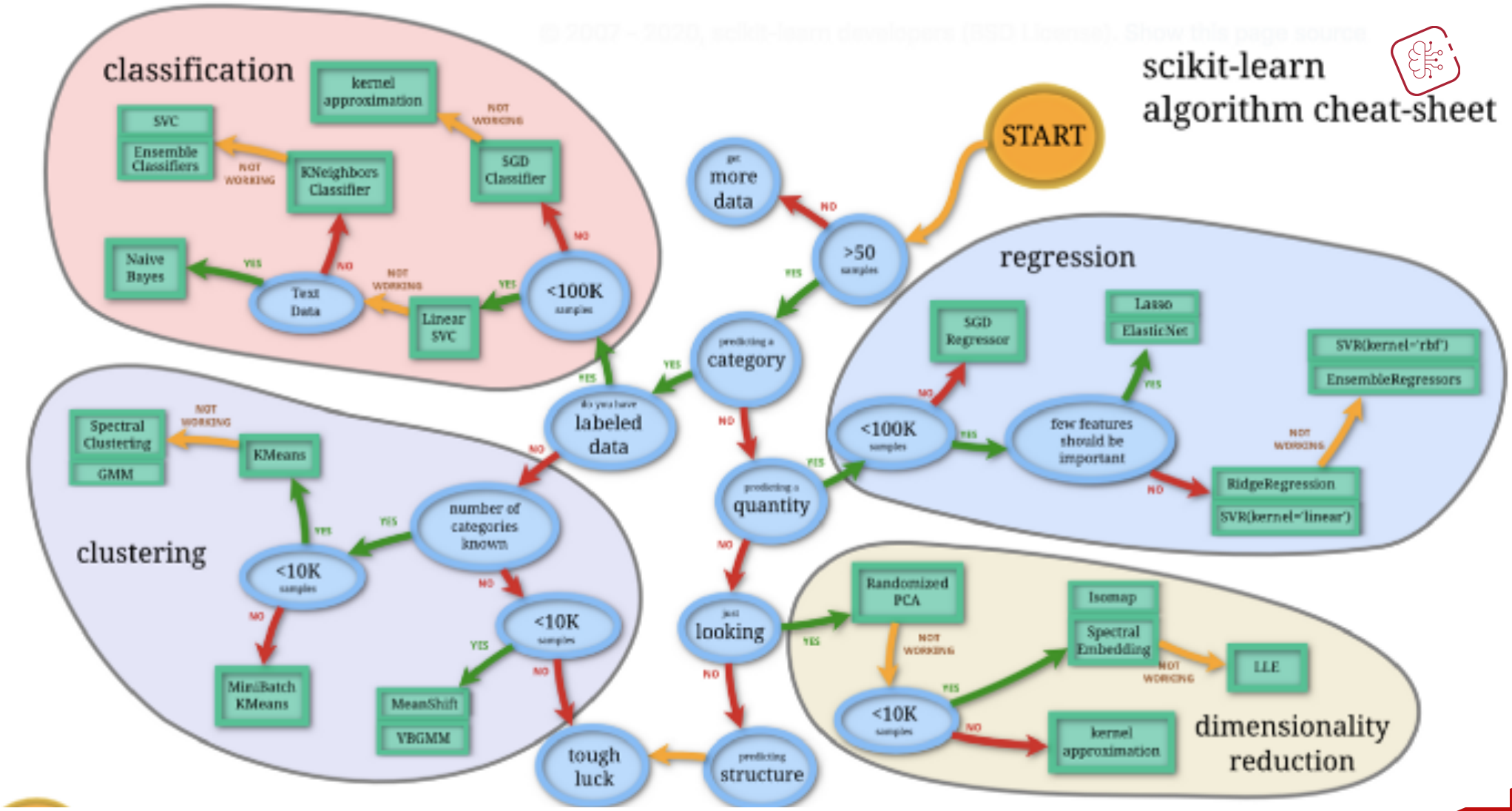
- Viene importata la **classe StandardScaler**, che permette di standardizzare i dati di allenamento e di test per renderli uniformi e confrontabili tra di loro.

ADDESTRAMENTO E PREVISIONE

- È molto semplice addestrare l'algoritmo KNN e fare previsioni con esso, specialmente quando si utilizza Scikit-Learn.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

- Il primo passaggio consiste nell'importare la classe **KNeighborsClassifier** dalla libreria **sklearn.neighbors**.
- Nella seconda riga, questa classe viene inizializzata con un parametro, quello di default che è n_neigbours. Questo è fondamentalmente il **valore di K dell'algoritmo**.
- Non esiste un valore ideale per K e viene selezionato dopo il test e la valutazione, tuttavia per iniziare, 5 sembra essere il valore più comunemente utilizzato per l'algoritmo KNN.


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
NO
YES
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
Naive Bayes
YES
Text Data
NO
NOT WORKING
Linear SVC
YES
<100K samples
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
<100K samples
few features should be important
RidgeRegression
SVR(kernel='linear')
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
number of categories known
<10K samples
MiniBatch KMeans
<10K samples
MeanShift
VBGMM
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
NOT WORKING
<10K samples
kernel approximation