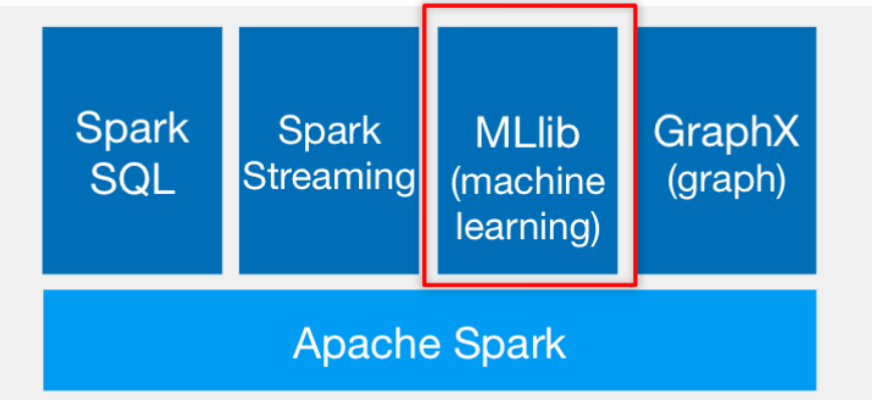
BIG DATA

CURS 7

Apache Spark MLlib
Introducere în Machine Learning pentru Big Data

Apache Spark MLlib



Apache Spark MLlib

- MLlib – biblioteca de *machine learning* (scalabilă) a lui Apache Spark
- **Simplu de folosit** în Java, Scala, Python, R
  - Se încadrează în API-urile Spark şi interoperează cu NumPy
  - Se pot folosi sursele de date Hadoop (HDFS, Hbase, fişiere locale) => este posibilă integrarea în fluxurile de lucru Hadoop

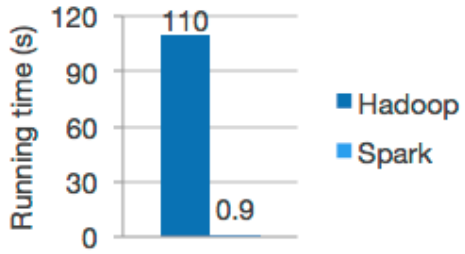
```
data = spark.read.format("libsvm")\
  .load("hdfs://...")

model = KMeans(k=10).fit(data)
```

Apel MLlib în Python

Apache Spark MLlib

- **Performanța** MLlib este determinată de performanța motorului Spark
  - Spark excelează în privința calculului interativ => MLlib rulează rapid
  - Pe de altă parte, MLlib asigură performanța algoritmilor implementați:
    - Algoritmi eficienți care utilizează iterația, ce pot avea rezultate mai bune decât aproximările de tip one-pass ce sunt folosite uneori în MapReduce.



Regresie logistică în Spark vs Hadoop

Apache Spark MLlib

- **Rulează pe diferite** medii şi pe diferite surse de date
  - Medii: Hadoop, Apache Mesos, Kubernetes, *standalone*, *cloud*
  - Sursele de date pe care le poate accesa includ: HDFS, Apache Cassandra, Apache Hbase, Apache Hive etc…
- **MLlib** conţine numeroase funcţionalităţi

Apache Spark MLlib

Funcţionalităţile MLlib:

- Algoritmi ML: algoritmii comuni de învăţare (clasificare, regresie, clustering, filtrare colaborativă)
- Prelucrarea caracteristicilor: extragere, transformare, reducerea dimensiunii, selecţie
- *Pipelines*: *tool*-uri pentru construirea, evaluarea şi optimizarea *pipeline*-urilor ML
- Persistenţă: salvarea şi încărcarea algoritmilor, modelelor şi a *pipeline*-urilor
- Utilitare: algebra liniară, statistică, prelucrarea datelor etc.

Algoritmi MLlib

- Clasificare: regresie logistică, naive Bayes etc.
- Regresie: regresie liniară generalizată, regresie de supraviețuire etc.
- Arbori de decizie, *random forests*, arbori *gradient-boosted*
- Recomandare: ALS (*Alternating Least Squares*)
- Clustering: K-means, GMM (*Gaussian Mixtures*) etc.
- Modelarea topic-urilor: LDA (*Latent Dirichlet Allocation*)
- Reguli de asociere, seturi de *item*-uri frecvente, *mining* de *pattern*-uri

Utilitare MLlib

- Algebră liniară distribuită: SVD, PCA etc.
- Statistică: statistici de sumarizare, testarea ipotezelor etc.

API Apache Spark MLlib

- **API bazat pe DataFrame** (pachetul **spark.ml**) – API-ul principal de ML pentru Spark
- **API bazat de RDD** (pachetul **spark.mllib**) – se află în mod mentenanţă (versiunea >= 2.0)
  - MLlib suportă în continuare acest API
  - Corectează bug-uri, dar nu adaugă funcţionalităţi noi
- De ce API bazat pe DataFrame?
  - API-ul DataFrame – mai *user-friendly* decât RDD
  - Beneficii DataFrame: surse de date, interogări SQL, optimizare (Catalyst, Tungsten), uniformitatea API-ului în diferitele limbaje în care este disponibil
  - Structurile de tip DataFrame facilitează *pipeline*-urile ML, în particular transformarea caracteristicilor

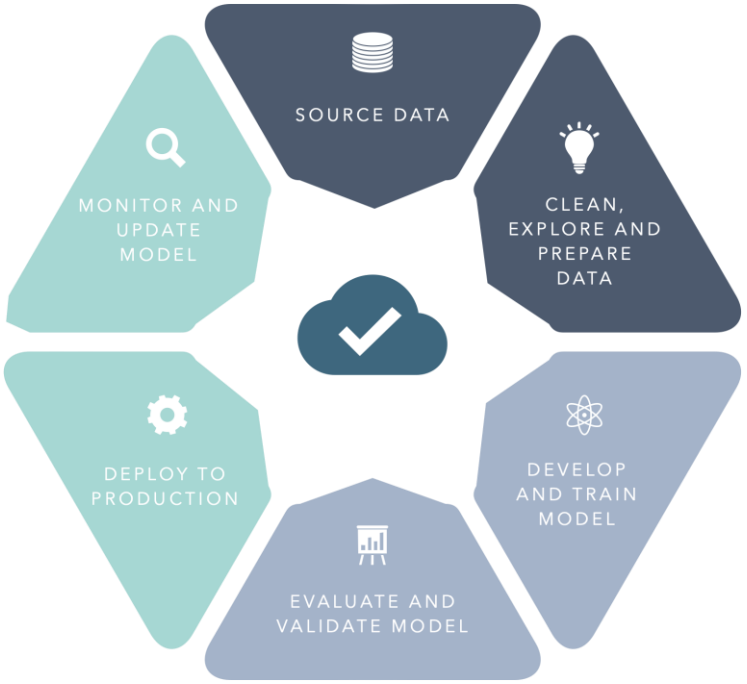
Scikit Learn vs MLlib

- Exemplele din partea a doua a cursului vor folosi biblioteca Scikit-Learn (sklearn)
- Sklearn are performanțe foarte bune atunci cînd datele încap în RAM.
- Python și sklearn realizează procesare in-memory, în mod nedistribuit.
- MLlib este mai adecvată în special atunci când setul de date este mai mare (de ordinul GB).
- Un aspect important - vizualizarea datelor
  - Scikit-learn are avantajul suportului pentru Pandas și Matplotlib
  - Procesul de dezvoltare de modele ML este iterativ și eficient
  - Rezultatele pot fi vizualizate, pot fi verificate presupuneri, pot fi folosite funcții suplimentare pentru testarea tipului distribuțiilor de probabilitate etc.

Noțiuni introductive în ML

Workflow ML

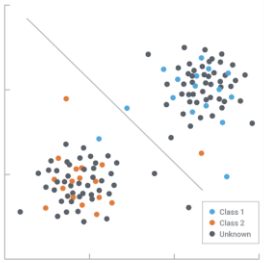
1. **Recuperarea datelor**: Identificarea surselor de date (interne sau externe) pertinente pentru problema dată.
2. **Pregătirea datelor**: curățarea, explorarea și transformarea datelor într-un format convenabil.
3. **Dezvoltarea modelului**: alegerea unui algoritm, construirea și antrenarea modelului pe datele pregătite.
4. **Evaluarea modelului**: evaluarea performanțelor modelullui după o metrică adecvată, ce reflectă obiectivul avut în vedere.
5. **Desfășurarea modelului**: deployment-ul modelului într-un mediu de producție pentru a servi predicțiilor
6. **Monitorizarea modelului**: reantrenarea modelului cu date noi în cazul scăderii performanței acestuia.



ML: Tipuri de învățare

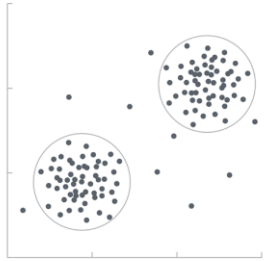
Învățare supervizată

Învățare ghidată de ***task***. Se bazează pe date adnotate. Scopul este cunoscut de la început.
Task-uri: Regresie, Clasificare etc.



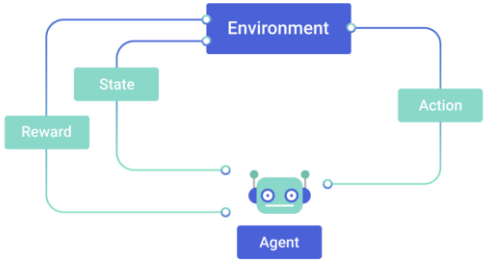
Învățare nesupervizată

Învățare ghidată de **date**. Vizează identificarea pattern-urilor în structura datelor. Nu se specifică niciun scop.
Task-uri: Clustering, Reducerea dimensiunii etc.



Învățare ranforsată

Agent informatic care învață să efectueze un task prin interacțiunea cu mediul, pe baza unui mecanism de recompense și penalități.



Învățarea supervizată: formalism

- Considerăm n obiecte (imagini, texte etc.) descrise prin p caracteristici sau atribute.
- Caracteristicile fiecărui obiect i sunt reprezentate sub forma unui vector $\vec{x}_i = (x_i^1, x_i^2, \dots, x_i^p)$ din $\mathbb{R}^p$
- Fiecărui obiect i îi este asociată o variabilă de ieșire y. Tipul lui y determină natura problemei (clasificare sau regresie).

Obiectiv: predicția ieșirii y a unui obiect nou, caracterizat de vectorul de atribute $\vec{x}$.

Cum? –Învățarea unei reguli de predicție pornind de la un set de date adnotate. Această regulă va fi folosită apoi pentru predicția variabilei target asociată la obiecte noi.

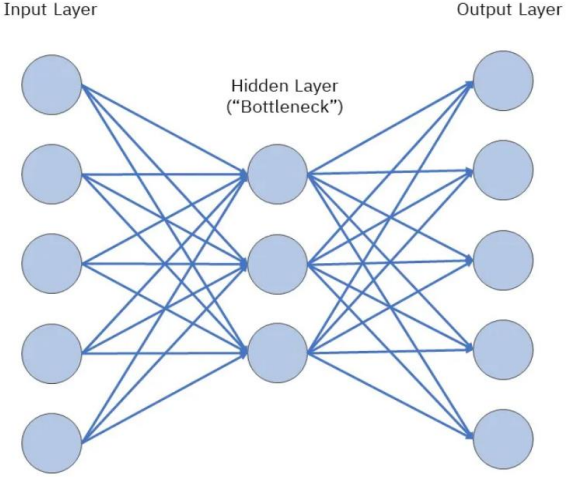
x				y
Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	FALSE	No
Sunny	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Rainy	Mild	High	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Sunny	Mild	High	FALSE	No
Sunny	Cool	Normal	FALSE	Yes

Învățare nesupervizată: Reducerea dimensiunii

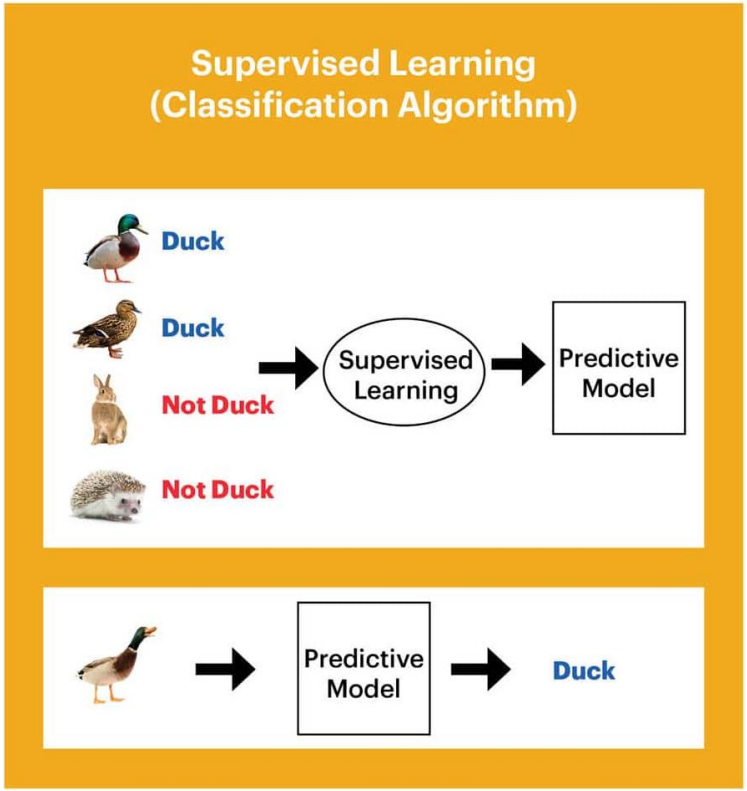
1. O tehnică utilizată atunci când numărul de caracteristici sau de dimensiuni dintr-un set de date este prea mare.
2. Permite reducerea numărului de intrări (de date) la o dimensiune gestionabilă păstrând, pe cât posibil, integritatea setului de date.
3. Este utilizată în mod frecvent în etapa de preprocesare a datelor.

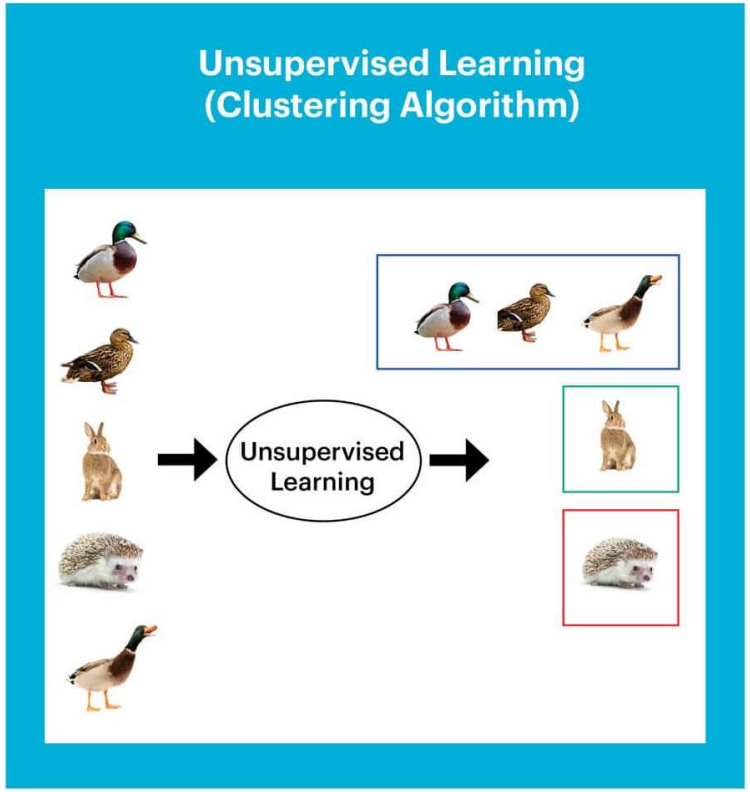
Exemple:

- **Analiza în componente principale (PCA) :** utilizată pentru a reduce redundanţele şi a comprima seturile de date prin extragerea caracteristicilor, păstrând maximum de varianţă.
- **Auto-Encoders :** exploatarea reţelelor neuronale pentru comprimarea datelor dintr-un nivel intermediar şi reconstruirea datelor originale pornind de la această reprezentare intermediară.



Învățare supervizată vs nesupervizată





ML: Algoritmi

Învățare supervizată

- Regresie liniară
- Regresie logistică
- K-NN
- Support Vector Machines (SVM)
- Arbori de decizie
- Random Forest
- Rețele neuronale

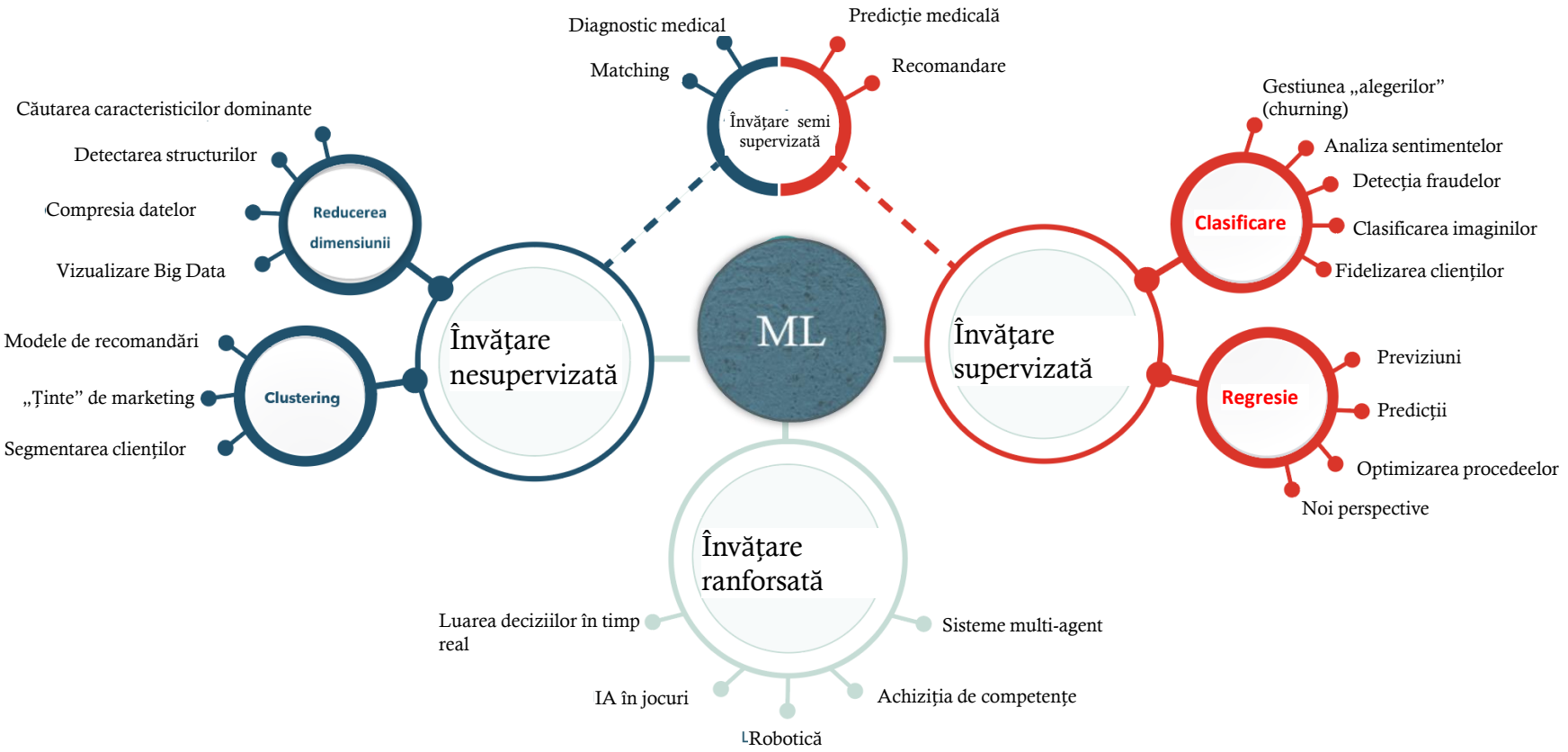
Învățare nesupervizată

- K-Means
- Clustering ierarhic
- Expectation Maximization (EM)
- Principal Component Analysis (PCA)
- Latent Dirichlet Allocation (LDA)
- Non-negative Matrix Factorization (NMF)
- Self-Organizing Maps (SOM)

Învățare ranforsată

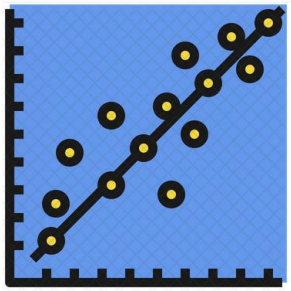
- Q-Learning
- SARSA
- Deep Q-Networks (DQN)
- Deep Deterministic Policy Gradient (DDPG)
- Asynchronous Advantage Actor-Critic (A3C)
- Proximal Policy Optimization (PPO)
- Trust Region Policy Optimization (TRPO)

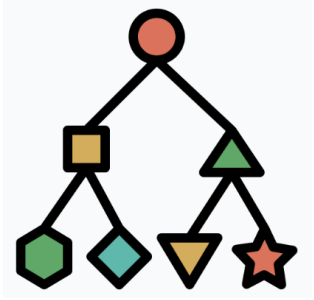
ML: Aplicații



Biblioteca Scikit-learn

- Scikit-learn este o bibliotecă open source în Python ce oferă o interfață unificată pentru antrenarea modelelor de ML.





Regresie
Linear Regression
Random forest
SVM
Nearest Neighbors

Clasificare
Logistic regression
Random forest
Naives Bayes
KNN

Clustering
K-means
Spectral clustering
Mean-shift

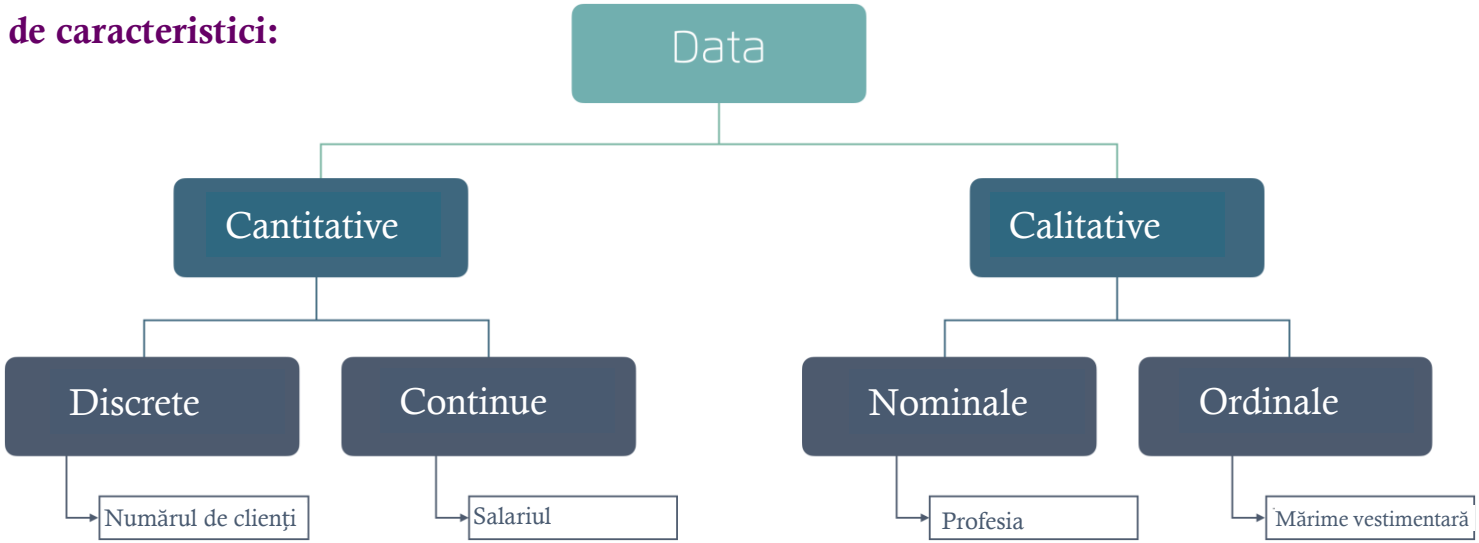
Importanța caracteristicilor

- Există proiecte de învățare automată care eșuează
- Ce poate face diferența?
  - Caracteristicile (*features* – i.e. Atribute, variabile) folosite

Definiție:

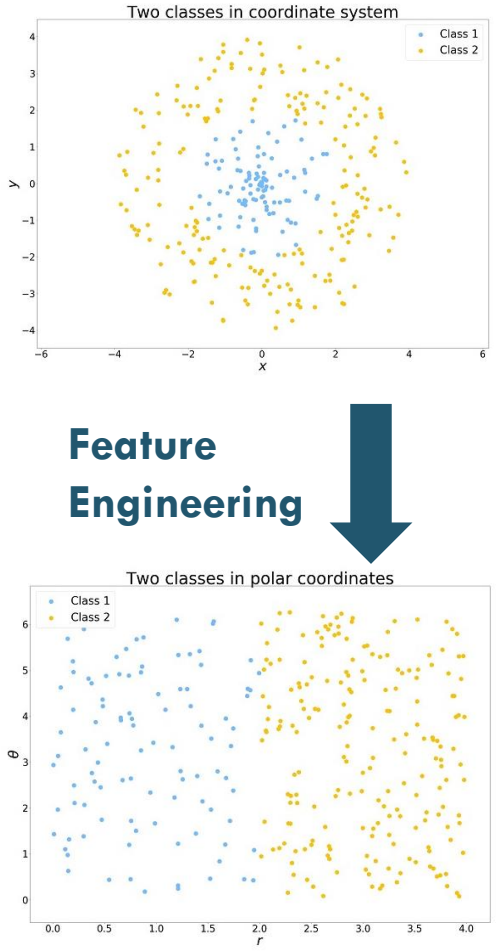
➢ **Caracteristică (*Feature*):** un atribut sau o variabilă ce descrie un aspect al conceptului reprezentat de setul de date (de exemplu: vârstă, greutate, înălțime, gen).

Categorii de caracteristici:



Prelucrarea caracteristicilor

- **Feature engineering:** procesul de transformare a datelor iniţiale într-un format uşor de interpretat.
- Ideea este aceea de a reprezenta datele în aşa fel încât algoritmii de ML să îşi poată atinge cu uşurinţă obiectivul de învăţare.
- *Feature engineering* conţine *task*-urile următoare:
  - Transformarea atributelor (de exemplu, transformarea km în metri)
  - Generarea de atribute (de exemplu, calculul raportului a două atribute existente)
  - Extragerea atributelor (de exemplu, ACP pentru reducerea dimensiunii)
  - Selecţia atributelor (de exemplu, prin suprimarea coloanelor redundante)



Transformări importante

- Scalare
  - Normalizare
  - Standardizare
- Encodare
  - Encodare categorială
  - One hot encoding
- Valori lipsă
  - Înlocuire printr-o valoare constantă
  - Înlocuire prin predicția unui model antrenat
- Separarea Train-Test
  - Separarea setului de date într-o parte pentru antrenament și alta pentru test
  - Scopul: Obținerea unui estimator nedeplasat pentru performanța modelului pe date noi.

Scalarea caracteristicilor

- ***Feature scaling***

Definiţie:

- ***Feature Scaling*: transformare ce constă în adaptarea domeniului de variaţie al atributelor, astfel încât toate variabilele să poată fi comparate la aceeaşi scară.
- Este o etapă importantă care poate îmbunătăţi semnificativ performanţa anumitor algoritmi de ML care depind de această scală.

Metode principale de Feature Scaling

1. **Normalizare** (Min-Max Scaling)
   - Redimensionarea atributului astfel încât valorile să varieze între 0 şi 1.
   - $x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$
2. **Standardizare** (sau Normalizare Z-score)
   - Redimesnionarea atributului astfel încât valorile să aibă media 0 şi deviaţia standard 1.
   - $x_{standard} = \frac{x - x_{mean}}{x_{std}}$

Feature Scaling în sklearn

Normalizare

Metoda 1:

```
from sklearn.preprocessing import
minmax_scale

scaled_data = minmax_scale(data)
```

Metoda 2:

```
from sklearn.preprocessing import
MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(data)
scaled_data = scaler.transform(data)
```

Standardizare

```
from sklearn.preprocessing import
StandardScaler

scaler = StandardScaler()
scaler.fit(data)
scaled_data = scaler.transform(data)
```

Encodarea caracteristicilor

- ***Feature encoding***

Definiție:

- ***Feature Encoding*:** operație ce constă în transformarea (maparea) valorilor unei variabile într-un format numeric ce poate fi interpretat de către algoritmii de ML.
- *Feature encoding* se aplică mai ales pe variabile categoriale pentru encodarea categoriilor (modalități, clase) în numere.

Metode principale de Feature Encoding

1. **Encodare categorială**
   - Fiecare categorie a variabilei este asociată unui număr (de exemplu: high -> 2, medium -> 1, low -> 0)
   - Limitare: introduce o ordine în categorii ce nu este neapărat adecvată (de exemplu, în cazul variabilei categoriale nominale)
2. **One-Hot Encoding**
   - Fiecare categorie a variabilei este asociată unei noi coloane binare
   - Limitare: generează o coloană pentru fiecare categorie, lucru ce poate pune probleme atunci când numărul de categorii este mare.

Feature encoding în sklearn

Categorical Encoding

```
from sklearn.preprocessing
import OrdinalEncoder

encoder = OrdinalEncoder()
encoder.fit(data)
encoded_data =
encoder.transform(data)
```

One-Hot Encoding

```
from sklearn.preprocessing
import OneHotEncoder

encoder = OneHotEncoder()
encoder.fit(data)
encoded_data =
encoder.transform(data)
```

Valori lipsă

- În datele reale pot lipsi unele valori
- Motive posibile:
  - Valoare indisponibilă sau nesalvată
  - Date corupte
- Prelucrarea datelor lipsă este o etapă importantă a curățării datelor deoarece unii algoritmi de învățare automată nu pot fi aplicați pe valori lipsă.
- Metoda de prelucrare a valorilor lipsă depinde foarte mult de date și de context.

Valori lipsă: metode de curățare

Abordarea 1: Strategia cea mai simplă de procesare a datelor lipsă constă în suprimarea observațiilor (liniilor) ce conțin astfel de valori.

Code Snippet:

```
dataset.dropna(inplace=True) # Suprimarea liniilor cu valori lipsă
```

Abordarea 2: Asimilarea valorilor lipsă cu o valorae constantă, o statistică ce va fi calculată (medie, mediană etc.) sau o valoare estimată de către un model predictiv.

Code Snippet:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy = "mean")  # Asimilare cu media
imputer.fit(data)  # Calculul mediilor coloanelor dataset-ului data
clean_data = imputer.transform(data)  # Executarea operației de curățare
```

Cross-Validation

Definiţie:

- ***Cross Validation*:** tehnică ce permite evaluarea modelelor ML şi testarea performanţei acestora. Permite compararea şi selectarea celui mai bun model pentru o problemă de predicţie.
- Există diferite moduri de abordare a cross-validation, cea mai simplă fiind Holdout Cross Validation.
- ***Holdout Cross Validation*:** constă în separarea unui set de date în 2 submulţimi, prima pentru antrenare şi a doua pentru testarea modelului antrenat pe un set de date care încă nu i-a fost expus. Proporţiile clasice de împărţire sunt de 70%-30% sau 80%-20%.

Exemplu:

```
from sklearn.model_selection import train_test_split
data_train, data_test = train(data, test_size=0.2) # Split de 80%-20%
```

Regresia liniară

- Algoritmul de regresie liniară constă în căutarea unei relații între vectorul $\vec{x} = (x_1, x_2, \ldots, x_p)$ al variabilelor independente și variabila *target* numerică y, de forma:

$$y = \sum_{i=1}^{p} a_i * x_i + a_0$$

Obiectiv: găsirea valorilor a_i

Soluție: Metoda celor mai mici pătrate

- Valorile a_i care minimizează suma pătratelor distanțelor între observația y_n și dreapta de regresie $\hat{y}_n = \sum_{i=1}^{p} a_i * x_i + a_0$

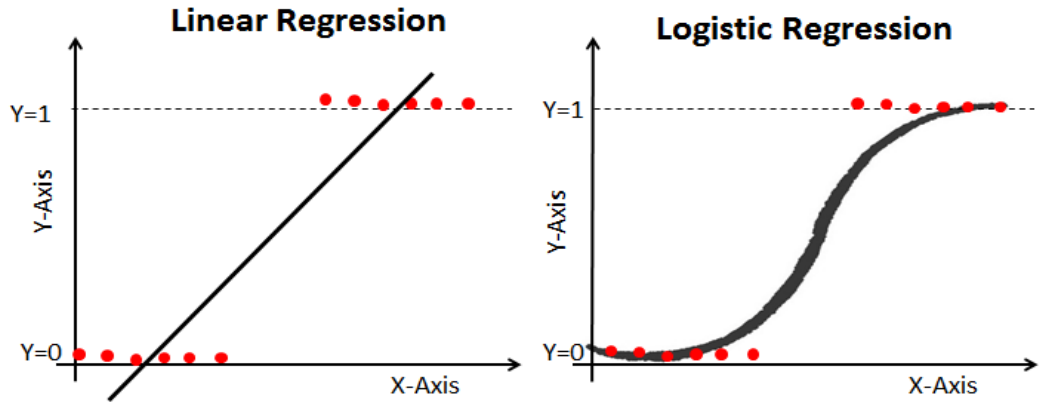
Demo: Regresie

Clasificare: Regresia logistică

- Regresia logistică este unul dintre algoritmii principali de clasificare binară (și unul dintre cei mai simpli).
- Extinde ideea regresiei liniare în cazul în care variabila dependentă y modelează o variabilă cantitativă (codată în 0/1).
- Este posibilă extinderea la cazuri de clasificare multi-clasă.

Exemple de aplicații:

1. Email spam vs email non spam
2. Tumoră malignă sau benignă
3. Aprobarea sau respingerea unui împrumut bancar



Clasificare: Regresia logistică

- Prima etapă: constă în calculul probabilității ca o observație să aparțină clasei 1, utilizând funcția Sigmoid. În acest caz, parametrul va fi ecuația de regresie liniară:

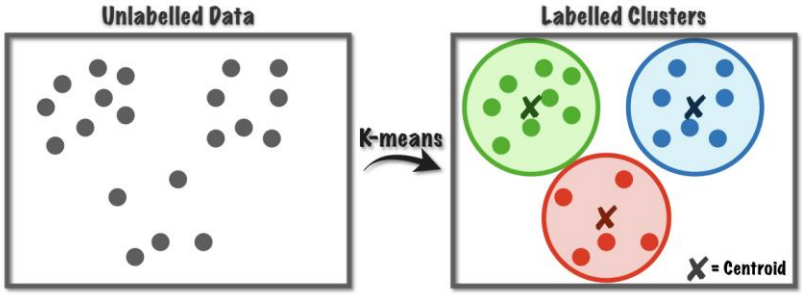
$$P(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}$$

- A doua etapă: o valoare prag este utilizată pentru a clasifica fiecare observație într-una dintre clase. De exemplu: pragul = 0.5
  - Dacă P(y=1)>0.5, observația va fi în clasa 1
  - Dacă P(y=1)≤0.5, observația va fi în clasa 0

Demo: Clasificare

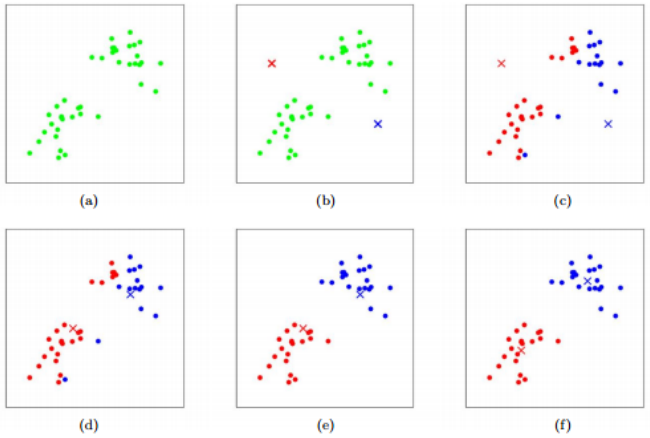
Clustering: K-Means

- Obiectivul este de a diviza observaţiile în K grupuri distincte astfel încât observaţiile din cadrul fiecărui grup să fie similare.
- Observaţie: Există K^N moduri de partiţionare pentru N observaţii în K grupuri.
- Exemple de aplicaţii: Clasificarea clienţilor, Clasificarea pe categorii a produselor, Recomandarea de filme etc.



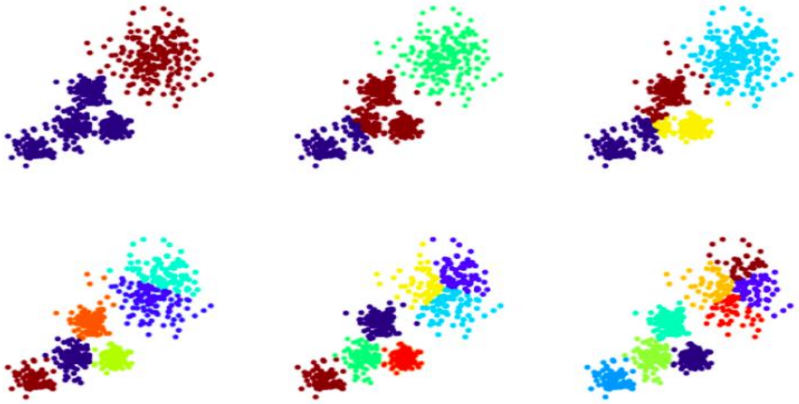
Algoritm :

1. Se aleg (aleator) K puncte (ce reprezintă centroizii).
2. Se asociază fiecare observaţie partiţiei (centroidului) celui mai apropiat.
3. Se repetă etapele următoare până la convergenţă:
   1. Pentru fiecare cluster, se calculează noua poziţie a centroidului (media punctelor clusterului)
   2. Se asociază fiecare observaţie centroidului cel mai apropiat.



Clustering: K-Means

- Valoarea maximă a lui *K* este egală cu *N* (numărul de observații).
- Care este cea mai bună valoare a lui *K*?



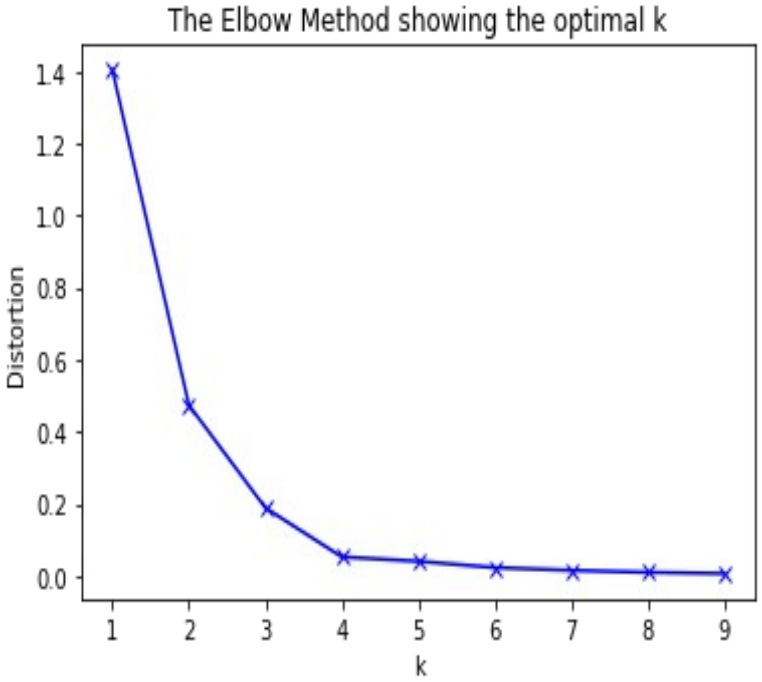
Dezavantaj 1: Valoarea lui *K* trebuie specificată în prealabil.

Dezavantaj 2: K-means este sensibil la valorile aberante.

K-Means: Metoda Elbow

Metoda Elbow:

- Se lansează K-Means cu diferite valori ale lui K și se calculează valoarea funcției de cost la fiecare iterație.
- Funcția de cost a lui K-Means este suma pătratelor distanțelor între fiecare centroid și diferitele observații care îi sunt asociate.
- Numărul optim de clustere K este punctul de inflexiune al curbei.



Demo: Clustering

Bibliografie

1. Apache Spark documentation (https://spark.apache.org/docs/latest).

2. https://spark.apache.org/docs/latest/ml-guide.html