KNN

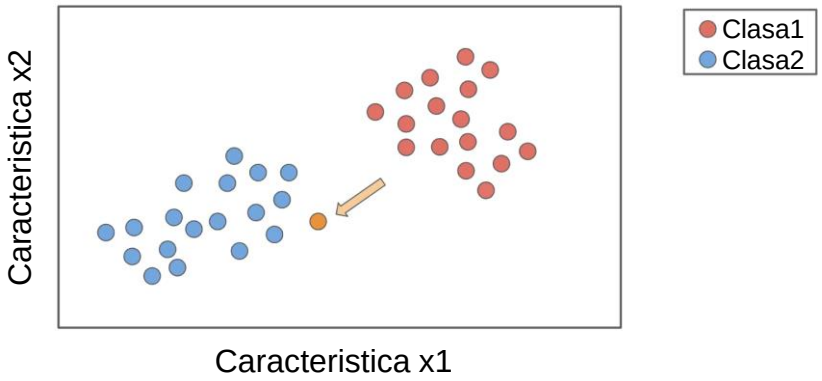
K-Nearest Neighbors

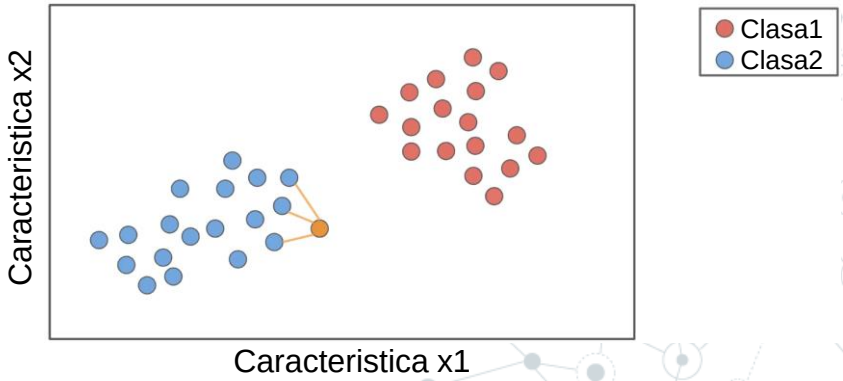
Ce ne așteaptă?

1. Esența algoritmului
2. Selectarea valorii lui K
3. Scalarea datelor
4. Instrumente Scikit-Learn

1. Esența algoritmului

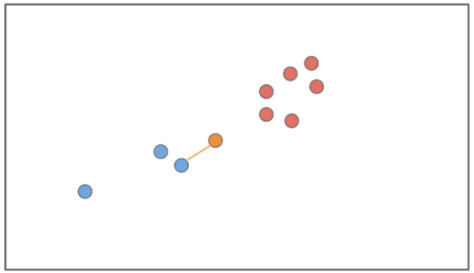
- Clasificarea unui anumit punct al datelor se realizează în funcție de categoria din care fac parte punctele vecine
- Exemplu de aplicare KNN în cazul datelor cu doar 2 caracteristici x1 si x2



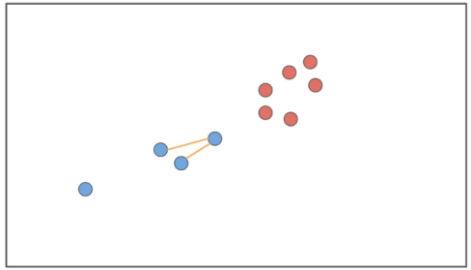


Importanța numărului K de ‘vecini analizați’

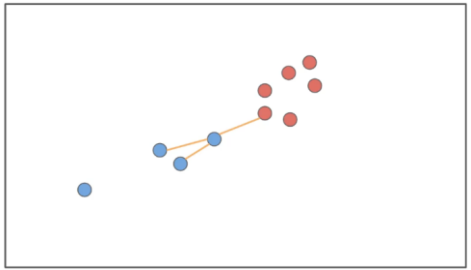
K=1



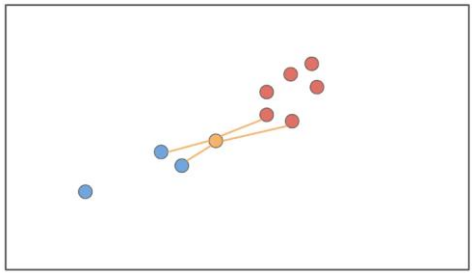
K=2



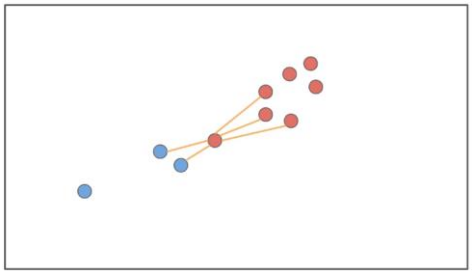
K=3



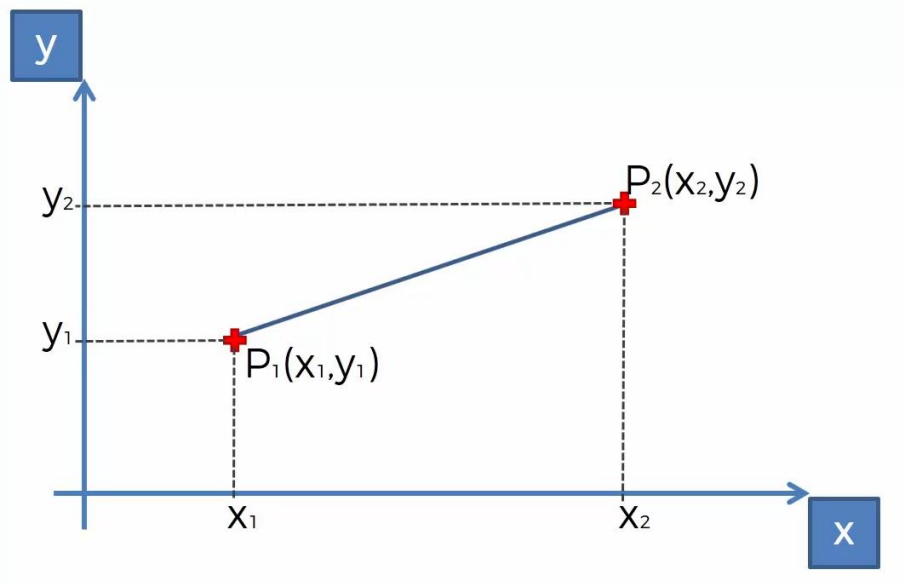
K=4



K=5



Distanța dintre puncte în cazul datelor cu 2 caracteristici



Distanța Euclidean (D) dintre punctele P1 si P2

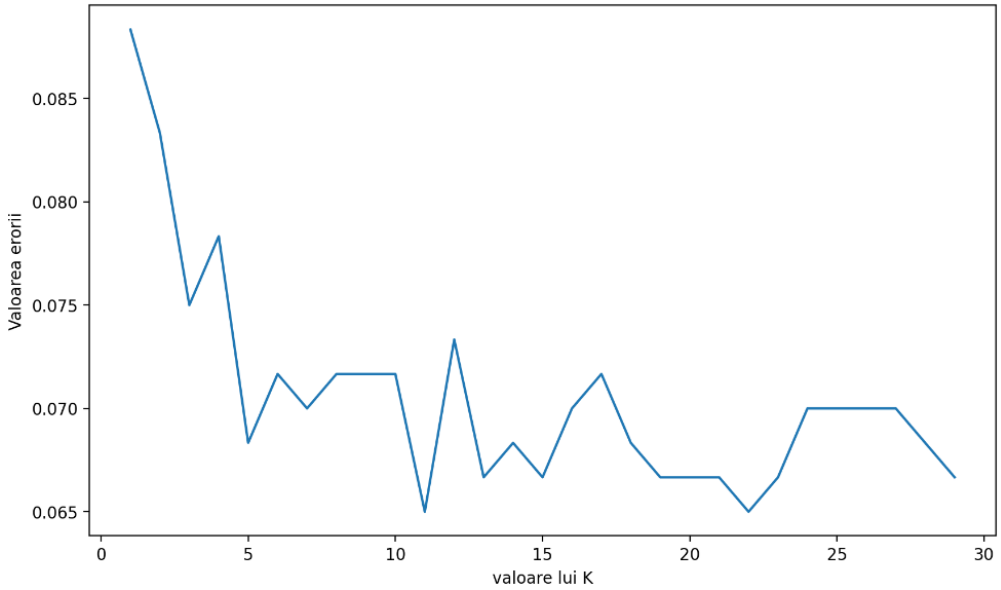
$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Alte metode de determinare a distanței: **Minkowski, Manhattan, Chebyshev**, etc

2. Selectarea valorii lui K

- Selectarea clasei noului punct se realizează după legea majorității numărului de vecini apropiați
- În cazul numărului de vecini egal pentru 2 clase se va selecta clasa celui mai apropiat vecina
- Pentru a evita situația de mai sus, de obicei, se selecteaza un număr K de valoare impară
- Numărul K se selectează din considerentul reducerii erorii dar și a complexității modelului
- Valoarea optima a lui K se poate selecta fie prin metoda ‘cotului’, fie prin procedura Cross Validation la crearea modelului.

Metoda ‘cotului’ - dependența valorii erorii de valoarea lui K

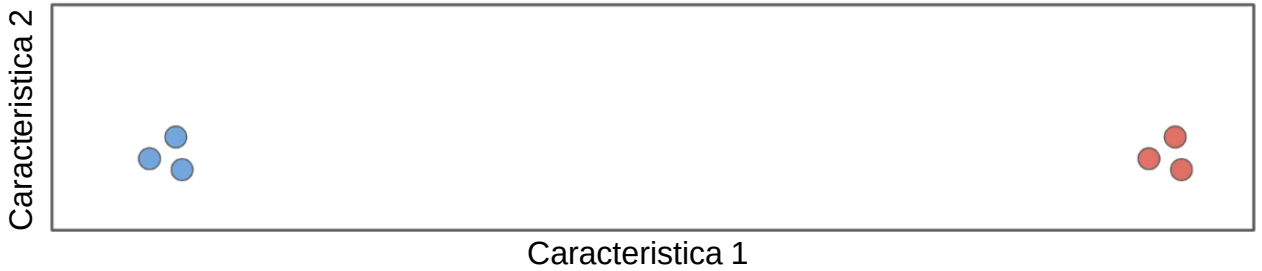


Pentru o precizie înaltă și o complexitate mai redusă se poate alege K=5

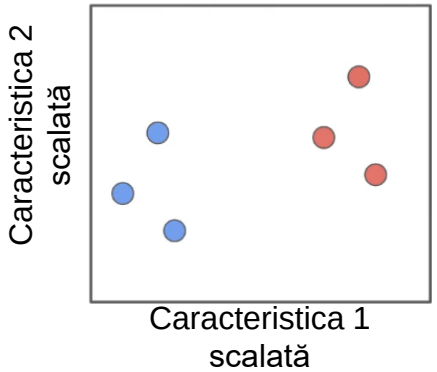
Pentru o precizie mai înaltă și o complexitate mai mare se poate alege K=11

3. Scalarea datelor

- **Algoritmul KNN necesită ca datele tuturor caracteristicilor să fie în aceeași bandă de valori**
- **Până la scalare**



- **După scalare**



4. Instrumente Scikit-Learn

- **Scalarea datelor:**
  - **Se importa clasa de scalare standardă StandardScaler**

```
from sklearn.preprocessing import StandardScaler
```

  - **Se creează obiectul de sclare**

```
scaler = StandardScaler()
```

  - **Se realizează trainingul obiectului de scalare pe datele de training**

```
scaler.fit (X_train)
```

  - **Se aplică scalare pe datele de training și de test**

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

- **Crearea modelului**
  - Se importa clasa algoritmului LogisticRegression

```
from sklearn.neighbors import KNeighborsClassifier
```

  - Se creează modelul cu fixarea valorilor hiper-parametrilor:
    - n_neighbors – numărul de vecini ce se vor considera (valori posibile = int, implicit =5)
    - algorithm – algoritmul utilizat pentru deminarea celor mai apropiați vecini (valori posibile = {'auto', 'ball_tree', 'kd_tree', 'brute'}, implicit = 'auto')
    - metric – metoda utilizată pentru determinarea distanței dintre puncte (valori posibile = {'minkowski', 'euclidean', 'manhattan', 'chebyshev'}, implicit= 'minkowski',)

```
model = KNeighborsClassifier(n_neighbors=5, algorithm = 'auto', metric= 'minkowski')
```

  - Se realizează trainingul modelului pe datele de training

```
model.fit(X_train_scalat, y_train)
```

  - Se realizează predicția pe datele de test

```
y_pred = model.predict(X_test_scalat)
```