Machine Learning

Metz Numeric School

M2I

Florian AUBRY - 2025

Programme

- **Jour 1:**
  - Fondamentaux du Machine Learning
  - Pratique: Modèles supervisés
- **Jour 2**
  - Modèles avancés et optimisation
  - Pratique: Introduction aux modèles non supervisés

Programme

- **Jour 3**
  - Introduction au Deep Learning
  - Pratique: Application au Traitement des langues
- **Jour 4**
  - Analyses avancées de données
  - Pratique: Combinaison de modèles en cas complexe

Programme

- **Jour 5**
  - Analyse d’image et application pratique

Sommaire

1. Préparation des données
2. Méthodes d’optimisation
3. Modèles non supervisés

Préparation des données

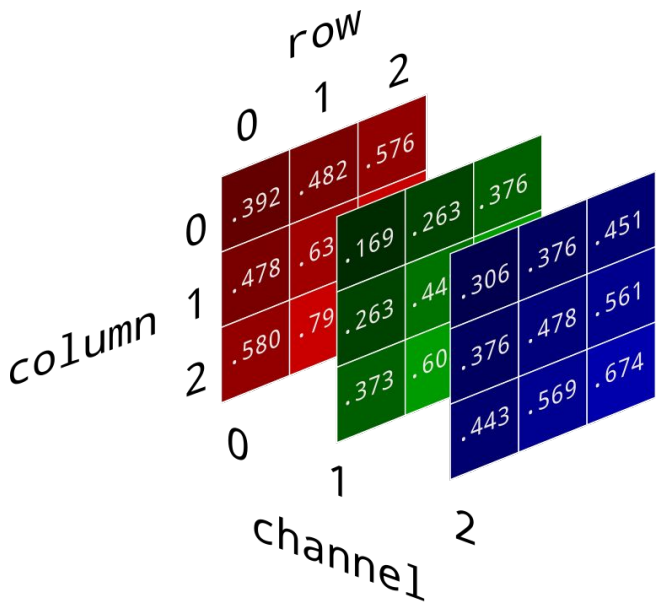
Objectif

- Numériser les données
- Normaliser/standardiser les données
- Gérer les biais dans la donnée d’entraînement
- Obtenir plus de données sans annoter plus

Préparation des données

Images

- Matrice de pixels 3D => Tableau [p_{1Red}, p_{1Green}, $p_{1\ Blue}$, p_{1Alpha}, ...]



Préparation des données

Images

- Transformation en Array Numpy (Pillow, Keras, OpenCV, ...)

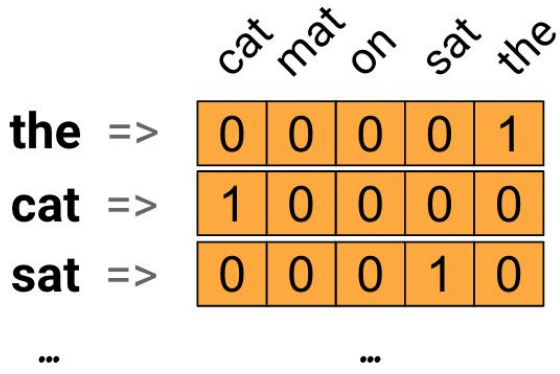
```
# python3 -m pip install Pillow
from PIL import Image
from numpy import asarray


# load the image
img = Image.open('Sample.png')

# Convert PIL images into NumPy arrays
img_array = asarray(img)
```

Préparation des données

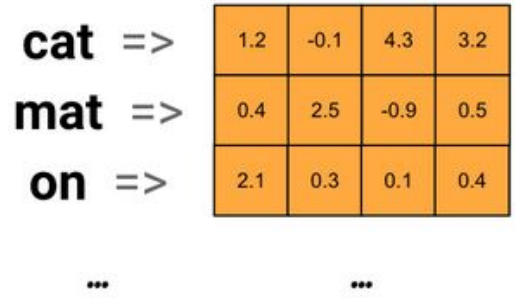
Texte

- Transformer les phrases en vecteurs
- Découper les phrases/mots (tokenization)
- Sparse Vectors:
  - Vecteur de longueur du vocabulaire
  - La plupart du vecteur est vide (Sparse)
- Dense Vectors:
  - Attribuer un vecteur complexe à un mot

One-hot encoding



A 4-dimensional embedding



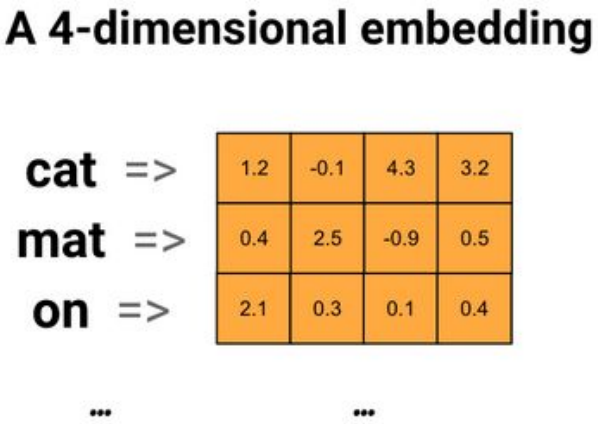
Préparation des données

Word Embeddings

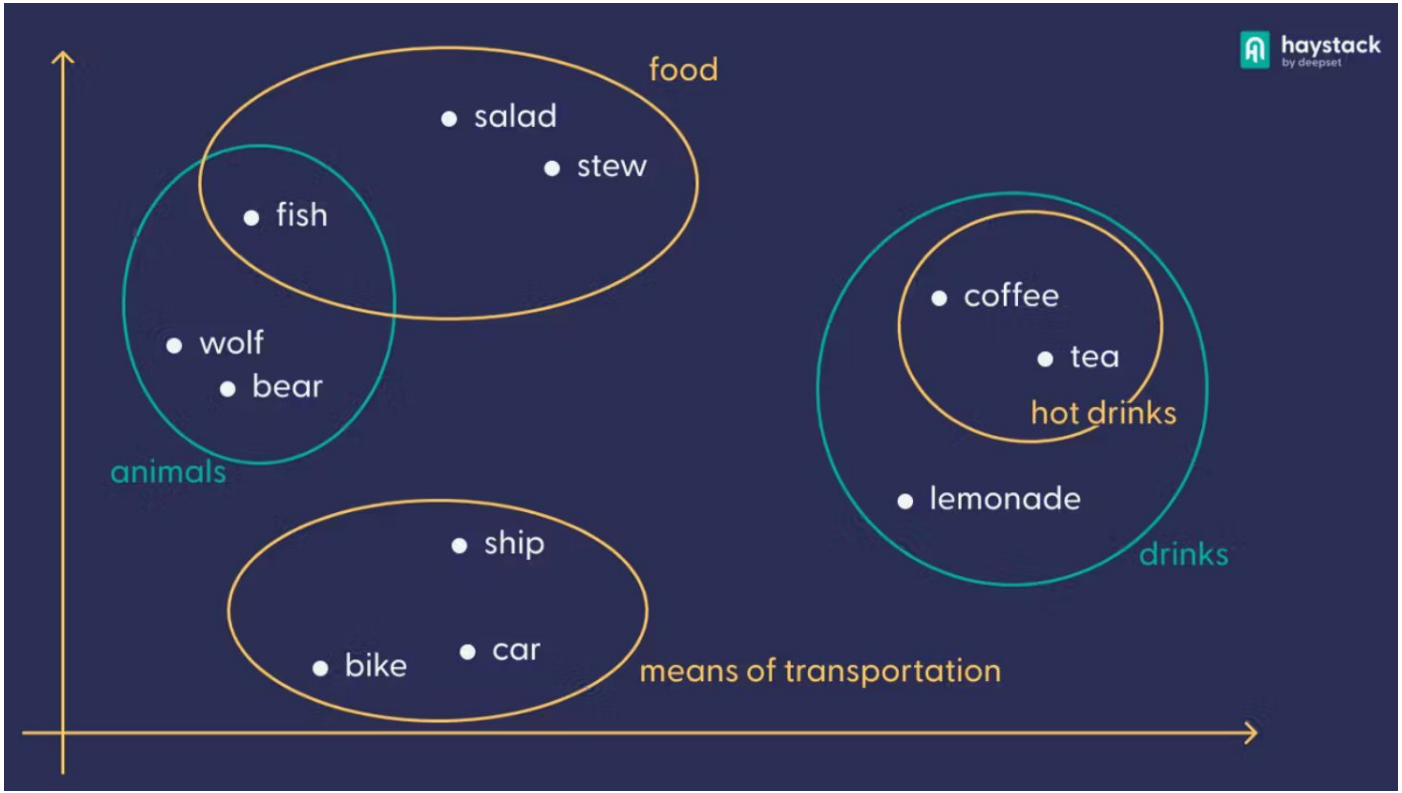
- Chaque mot proche a un vecteur proche
- Une table de correspondance fournie les valeurs

Bibliothèques

- GloVe, SpaCy, FastText



Préparation des données



Préparation des données

SpaCy Word Embeddings for SciKit Learn

```
# python -m spacy download en_core_web_lg
import spacy

word_emb = spacy.load("en_core_web_lg")
sentences = [word_emb("The cat sat on the mat."), word_emb("The dog ate a croissant")]
sentence_list = [s.reshape(1,-1) for s in sentences] # Invert vector shapes in array
input_data = np.concatenate(sentence_list)
```

Préparation des données

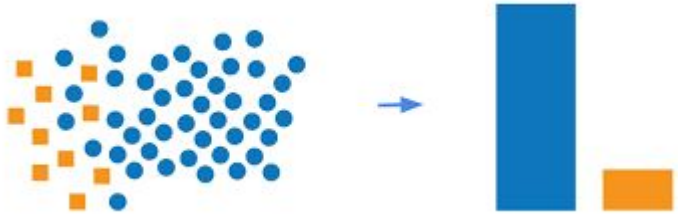
Analyse de sons

- Traitement du signal:
  - Transformée de Fourier
  - Spectrogrammes
  - ...

Préparation des données

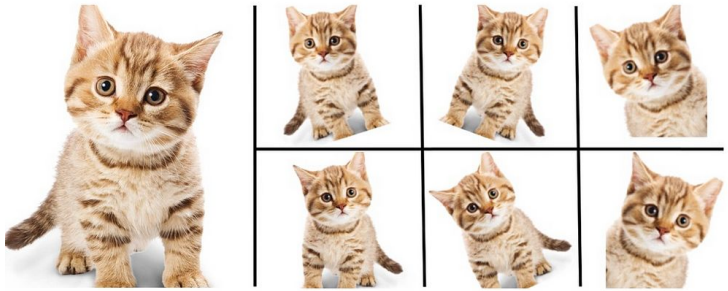
Dataset déséquilibré (Skewed Data):

- Laisser ainsi
- Augmenter/Diminuer la taille
- Adapter le poids des classes



Augmenter la taille du dataset

- Changer l'orientation des images
- Ajouter du bruit
- Extra/Interpoler des points



Préparation des données

Normalisation

- Ajuster les données entre 0 et 1

Standardisation

- Appliquer un Z-Score
  - lissage autour de la moyenne et de l’ écart-type)

Préparation des données

Pipelines SciKit Learn

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import VarianceThreshold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import Normalizer, StandardScaler, MinMaxScaler, PowerTransformer

pipe = Pipeline([
('scaler', StandardScaler()),
('selector', VarianceThreshold()),
('classifier', KNeighborsClassifier())
])

pipe.fit(X_train, y_train)
```

Méthodes d’optimisation

Ajustement des Hyperparamètres

- Sélection des variables externes influençant l’apprentissage
- Éviter le sur/sous-apprentissage

Méthodes d’optimisation

Grid Search

- Tester une grille de paramètres
- Évaluer toutes les combinaisons possibles

Randomized Search

- Tester aléatoirement une grille de paramètres

Halving

- Parcours de grille dichotomique

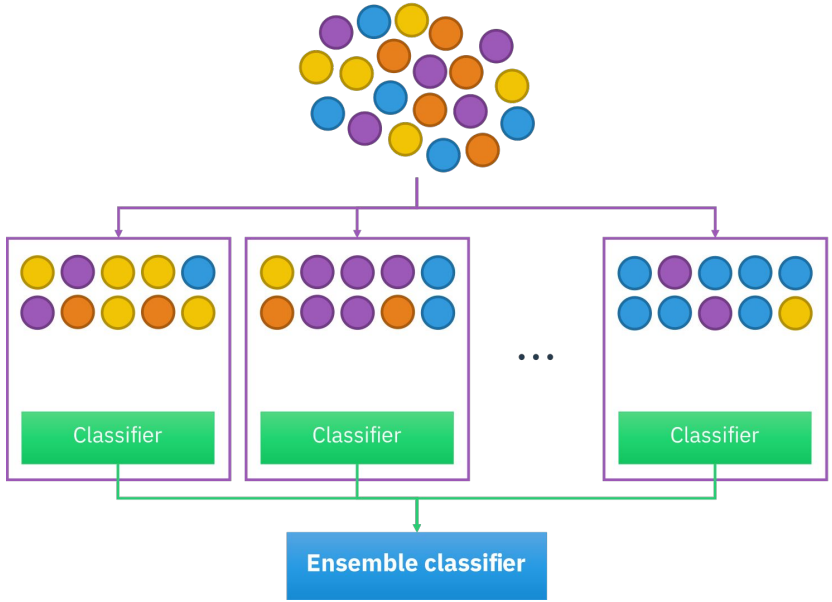
Méthodes d'optimisation

```python
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'gamma':['scale', 'auto'],
              'kernel': ['linear']}

# n_jobs=-1 to use all CPUs
# refit to have the best model at the end
grid = GridSearchCV(SVC(), param_grid, refit = True,n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)
```

Méthodes d’optimisation

Méthodes d’ensembles

- “Bagging Classifier”:
  - Entraîner plusieurs modèles similaires
  - Jeux de données différents
  - Calculer la moyenne des prédictions
  - Réduction de la variance



Méthodes d’optimisation

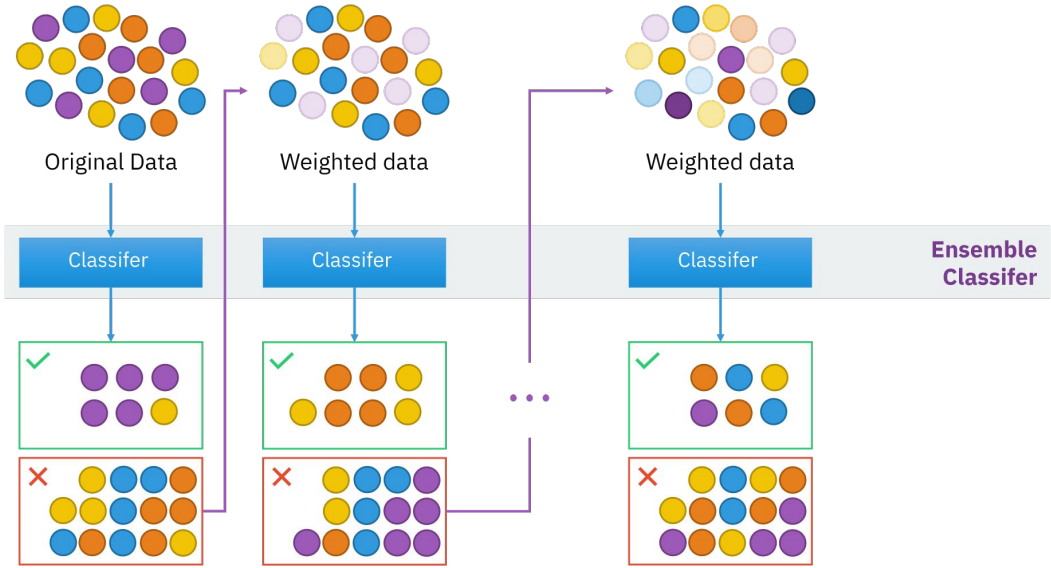
Bagging Classifier

- Bootstrap
  - Tirer aléatoirement des données du set original
  - Out-of-bag: données non tirées aléatoirement
- Inconvénient: très dépendant du bootstrap, lent
- Avantage: plus fiable, résolution non linéaire, moins de risque de sur-apprentissage
- Exemple: RandomForests

Méthodes d’optimisation

Boosting

- Utiliser les erreurs d’un modèle pour entraîner le suivant
- Réduction du biais
- Exemple: AdaBoost, XGBoost, ...



Méthodes d’optimisation

Validation Croisée

- Assurer la capacité de généralisation du modèle
- Éviter l’overfitting
- Toujours conserver un jeu de test indépendant:
  - Le modèle final est entraîné sur l’ensemble des données de ‘train’

Méthodes d’optimisation

Validation Croisée

- Généralement k = 10

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report

y_predict = cross_val_predict(model, X_train, y_train, cv=5)
print(classification_report(y_train, y_predict))
```

Méthodes d’optimisation

Dropout

Pour les réseaux complexes, risque de sur-apprentissage :

- Permet d’éteindre aléatoirement une partie des neurones (poids à 0)
- Changement à chaque itération
- Pour l’inférence : on applique un poid en fonction des neurones éteints à l’entraînement

Méthodes d’optimisation

```
from tensorflow.keras.layers import Dense, Dropout

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))  # Désactive 50% du réseau
model.add(Dense(10, activation='softmax')) [])
}
```

Modèles Non Supervisés

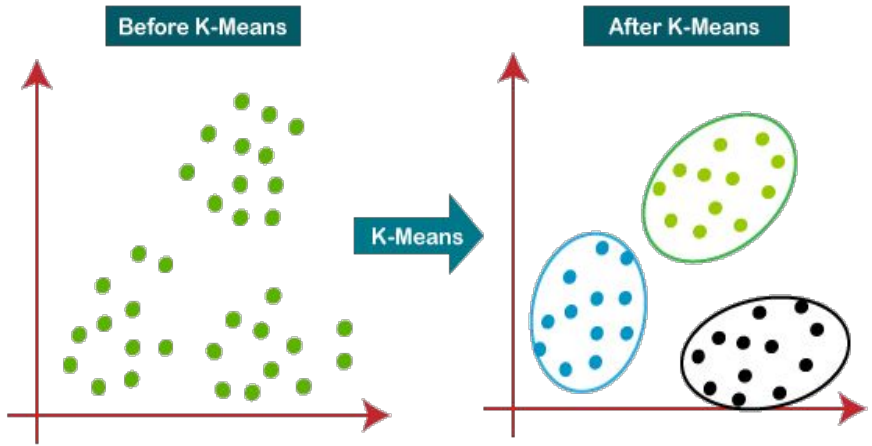
Modèles non supervisés

- Utilisation d’une fonction d’erreur (loss) pour entraîner un modèle
- Découverte des schémas dans la donnée au fur et à mesure
- La mesure de précision devient l’interprétation de la valeur de l’erreur du modèle

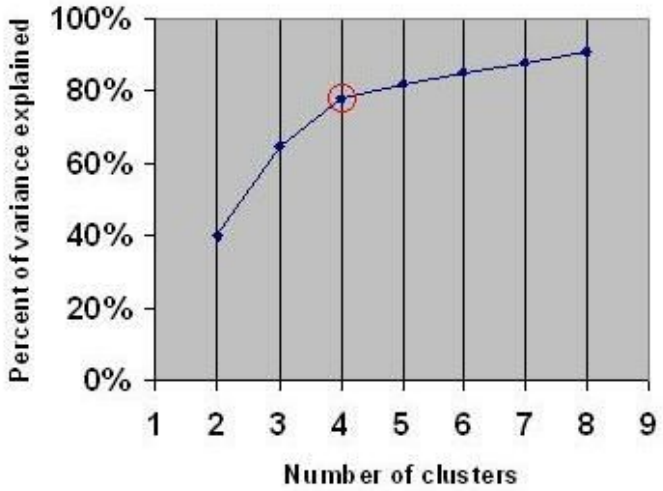
Modèles Non Supervisés

K-Means Clustering

- Grouper les données proches
- Calcul de distance aux centroïdes
- Basé sur la méthode des moindres carrés:
  - Minimiser la distance du centroïde aux points du cluster



Modèles Non Supervisés



K-Means Clustering

Détermination du nombre de clusters :

- Elbow Method (à droite)
- Coefficient de Silhouette :
  - Mesure de distance entre les points du groupe et celle avec ses voisins
- Autres méthodes statistiques

Modèles Non Supervisés

Analyses en Composantes Principales (PCA)

- Réduction de dimensions
- Identification des directions le long desquelles les données varient le plus
- Les Composantes principales conservent alors la variance (information)

Modèles Non Supervisés

Analyses en Composantes Principales (PCA)

```python
from sklearn.decomposition import PCA
import numpy as np

# Start with 2D data
data = np.array([[1, 2], [2, 3], [3, 4], [4, 5]])

pca = PCA(n_components=1)

# Appliquer la PCA aux données
data_transformed = pca.fit_transform(data)
```

Partie Pratique

Vous disposez d'un jeu de données représentant les clients d'une banque. L'objectif est de prédire le départ (churn) des clients avant que cela ne se produise

- Réalisez une étude préalable des données
- Entraînez le modèle vous paraissant le plus approprié et calculez les différentes métriques permettant d'évaluer vos résultats en validation croisée (pas de Bagging, RandomForest ou réseau de neurones)

Partie Pratique

A présent, nous allons tenter d’optimiser notre approche en utilisant

- Un modèle PCA pour réduire le nombre de dimensions
- Une Grid Search pour optimiser les hyperparamètres
- Un modèle fort type Bagging, RandomForest ou Boosté