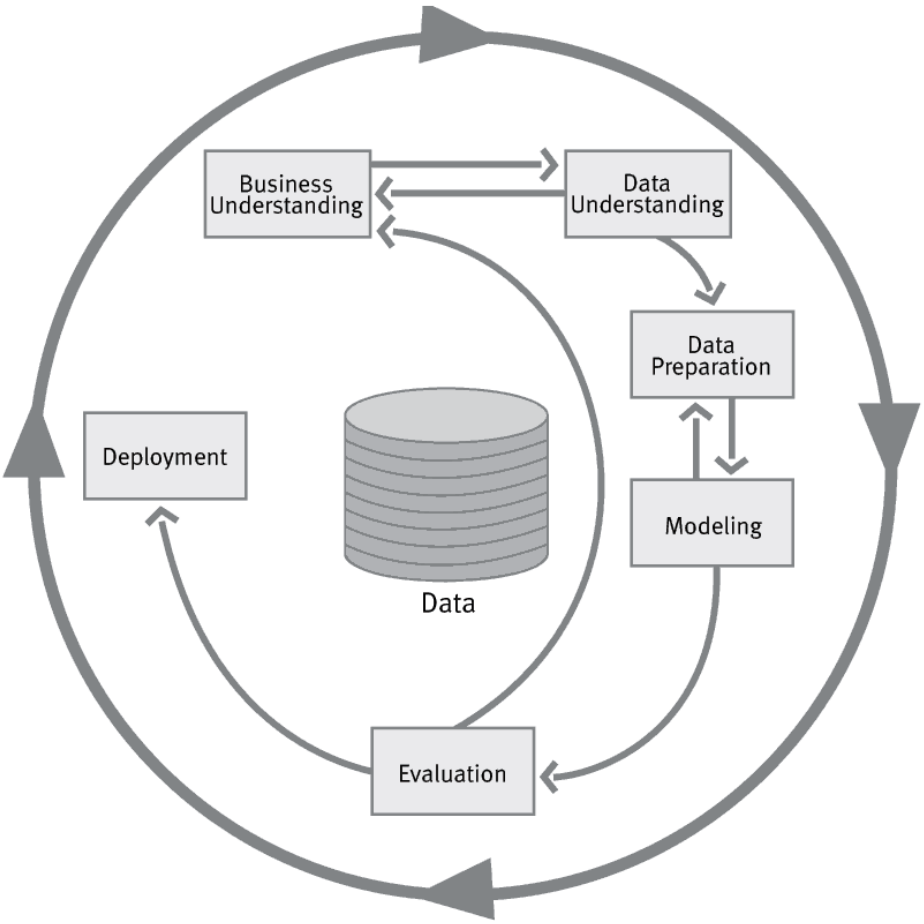
Analyse et prediction de l'attrition des employes

> Attrition : Diminution de la quantite de quelque chose

Avant de commencer, il faut d'abord specifier les etapes standards qu'on doit suivre dans notre projet de Data Mining.



1. Definir la problematique
2. Acquerir et comprendre les donnees
3. Preparer les donnees *(nettoyage & normalisation)*
4. Modeliser *(programmation et entrainement)*
5. Evaluer le modele *(tester sur les donnees qu'on a deja)*
6. Deployer le modele *(tester sur des donnees nouvelles)*

> **Normaliser** : Rendre les donnees du meme types.
> *Exemple : si l'age d'un employe contient "QUINZE" au lieu de "15" on la rend 15 ou bien on enleve l'employe.*

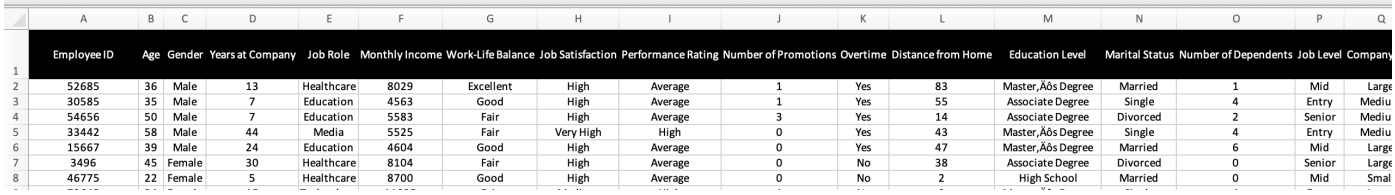
Definir la problematique

L'objectif est de comprendre pourquoi les employes quittent l'entreprise, et pour faire cela il faut construire un modele predictif qui peut predire les prochains employes s'ils vont rester ou quitter l'entreprise.

Acquerir et comprendre les donnees

On telecharge les donnees (*test.csv* & *train.csv*) depuis : https://www.kaggle.com/datasets/stealthtechnologies/employee-attrition-dataset

On importe le CSV a excel pour visualiser les donnees d'une maniere claire.



Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents	Job Level	Company
52685	36	Male	13	Healthcare	8029	Excellent	High	Average	1	Yes	83	Master,Âôs Degree	Married	1	Mid	Large
30585	35	Male	7	Education	4563	Good	High	Average	1	Yes	55	Associate Degree	Single	4	Entry	Mediu
54656	50	Male	7	Education	5583	Fair	High	Average	3	Yes	14	Associate Degree	Divorced	2	Senior	Mediu
33442	58	Male	44	Media	5525	Fair	Very High	High	0	Yes	43	Master,Âôs Degree	Single	4	Entry	Mediu
15667	39	Male	24	Education	4604	Good	High	Average	0	Yes	47	Master,Âôs Degree	Married	6	Mid	Large
3496	45	Female	30	Healthcare	8104	Fair	High	Average	0	No	38	Associate Degree	Divorced	0	Senior	Large
46775	22	Female	5	Healthcare	8700	Good	High	Average	0	No	2	High School	Married	0	Mid	Smal

A partir de cela on peut voir que certaines colonnes on un type fixe *(Nombre entier: Age)* et d'autres variables *(multi-choix (yes, no): Overtime)*, et a partir de cela on peut decrire nos colonnes comme suit :

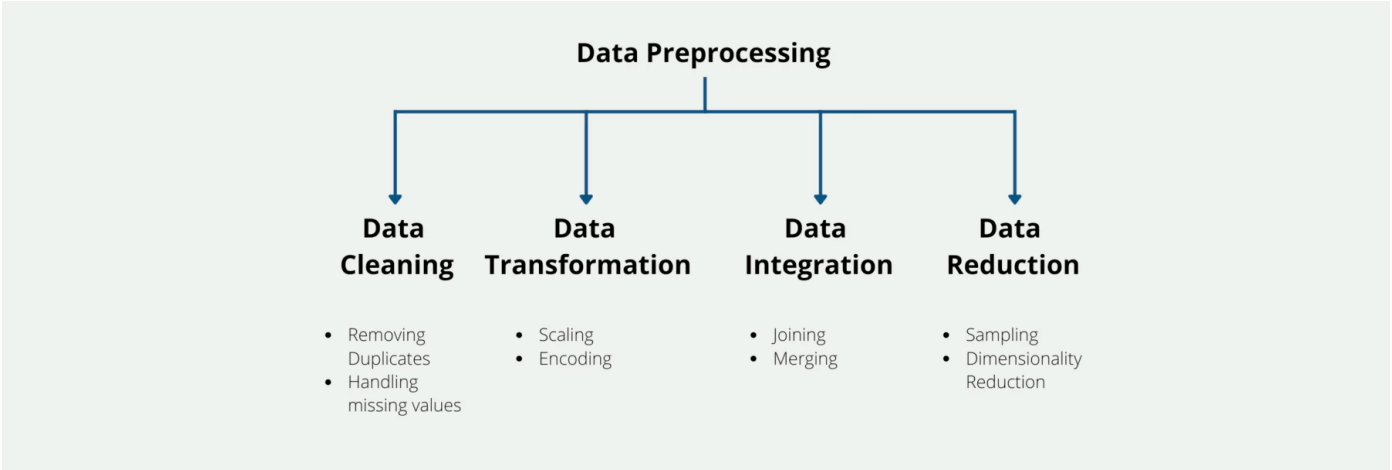
```
Employee ID             : Integer
Age                     : Integer
Gender                  : ['Male', 'Female']
Years at Company        : Integer
Job Role                : ['Healthcare', 'Education', 'Media', 'Technology', 'Finance']
Monthly Income          : Integer ($)
Work-Life Balance       :  ['Excellent', 'Good', 'Fair', 'Poor']
Job Satisfaction        : ['High', 'Very High', 'Medium', 'Low']
Performance Rating      : ['Average', 'High', 'Below Average', 'Low']
Num of Promotions       : [0, 1, 2, 3, 4]
Overtime                : ['Yes', 'No']
Distance from Home      : Integer (miles)
Education Level         : ['Master's Degree', 'Associate Degree', 'High School',
'Bachelor's Degree', 'PhD']
Marital Status          : ['Married', 'Single', 'Divorced']
Num of Dependents       : [0, 1, 2, 3, 4, 5, 6]
Job Level               : ['Mid', 'Entry', 'Senior']
Company Size            : ['Large', 'Medium', 'Small']
Company Tenure          : Integer
Remote Work             : ['No', 'Yes']
Leadership Opportunities: ['No', 'Yes']
Innovation Opportunities: ['No', 'Yes']
Company Reputation      : ['Poor', 'Fair', 'Good', 'Excellent']
Employee Recognition    : ['Medium', 'High', 'Low', 'Very High']
Attrition               : ['Stayed', 'Left']
```

Pour comprendre plus de details sur chaque colonne, il faut revenir a la description dans le lien au dessus.

Preparer les donnees

> **Pretraitement des donnees** (data preprocessing), est un ensemble des operations qu'on applique sur les donnees pour les rendre utilisable et eviter des problèmes.

Cette etape peut etre repeter plusieurs fois jusqu'a ce qu'on aura une Dataset claire et clean.
Le process standard de pretraitement des donnees est decrit comme suit :



> **Note I :** *Il faut garder une ou plusieurs copies de votre Dataset, puisque la corruption de la Dataset lors du pretraitement est une chose recurrente.*
>
> **Note II** : *On ne va pas utiliser la Data Reduction dans ce cas parcequ'elle transforme les donnees pour un besoin précis.*

Le script suivant *(cleaningData.py)* ***detecte les anomalies*** dans la dataset :

```
import pandas as pd
import numpy as np
import re

# Load the dataset
df = pd.read_csv("./test.csv")

# Detect rows with missing values
missing_values = df[df.isnull().any(axis=1)]

# Detect rows with special characters in string columns
def has_special_characters(value):
    if isinstance(value, str):
        return bool(re.search(r"[^a-zA-Z0-9\s]", value))
    return False

special_char_rows = df.applymap(has_special_characters).any(axis=1)
rows_with_special_chars = df[special_char_rows]
```

```
# Detect duplicate rows
duplicates = df[df.duplicated()]

# Output summary
print(f"\nSummary:")
print(f"Total rows with missing values: {len(missing_values)}")
print(f"Total rows with special characters: {len(rows_with_special_chars)}")
print(f"Total duplicate rows: {len(duplicates)}")
```

Le resultat recu est comme suit :

```
youssef@yousbot Attrition % python3 cleaningData.py

Summary:
Total rows with missing values: 0
Total rows with special characters: 7506
Total duplicate rows: 0
```

Ca veut dire que notre dataset contient des characteres speciaux qu'il faut nettoyer.

Le script suivant *(cleanSpecialCharacters.py)* ***nettoye les donnees des characteres speciaux*** en les remplacant par des characteres alpha-numerique.

```
import pandas as pd
import re

# Function to clean special characters
def clean_special_characters(value):
    if isinstance(value, str):
        return re.sub(r"[^a-zA-Z0-9\s]", "", value)  # Keep only alphanumeric and spaces
    return value

# Read the dataset
df = pd.read_csv("test.csv")

# Apply the cleaning function
df_cleaned = df.applymap(clean_special_characters)

# Save the cleaned dataset
df_cleaned.to_csv("test_cleaned.csv", index=False)
print("Special characters cleaned. Cleaned dataset saved as 'test_cleaned.csv'.")
```

Exemple de donnees nettoyées (avant, apres)

Master,Äôs Degree	Masters Degree
Associate Degree	Associate Degree
Associate Degree	Associate Degree
Master,Äôs Degree	Masters Degree
Master,Äôs Degree	Masters Degree
Associate Degree	Associate Degree

Maintenant le *test_cleaned.csv* contient une Dataset nettoyee et prete a etre utilise par nos modeles qu'on va creer par la suite.

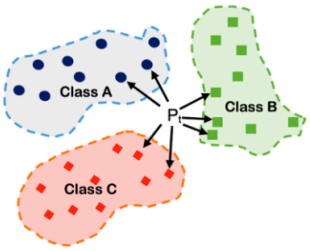
Modéliser *(programmation et entrainement)*

Pour choisir l'algorithme appropries a notre besoin, il faut d'abord les comprendres et étudier chacun selon ses propres caracteristiques, voir meme les appliquer tous si necessaire et comparer les resultats.

On doit choisir entre KNN, K-Means et Arbre de decision.

KNN : K-Nearest Neighbors

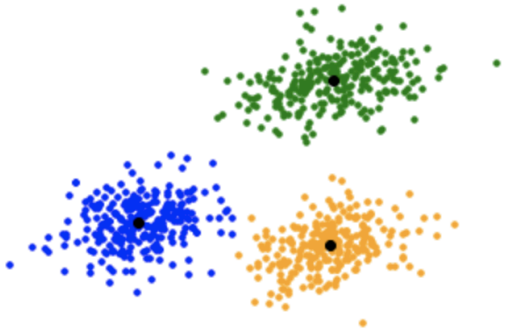
Pour expliquer l'algorithme facilement. Imaginez des elements disperses partout. KNN va essayer de les regroupper dans des groupes selon leur proximité. Et ceux qui sont proches sont regroupes dans un groupe. Chaque nouveau element venu (P), pour le classer le KNN va voir sa proximite de l'une des classes (groupe) existante (Classe A ou B ou C), et va l'ajouter dedans.



> **Note** : *KNN est un algorithme predictif. Il utilise les donnees existantes pour classifier une nouvelle donnee.*

K-Means

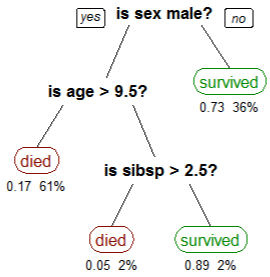
Imaginez des elements disperes partout. KMeans va essayer de les classer dans des classes selon la médiane (our le point centre de chaque classe).



> **Note** : *KNN est un algorithme descriptif. Il utilise les donnees pour trouver des Patterns ou modeles dans les donnees eux memes.*

Arbre de Decision

L'arbre de décision fonctionne comme un diagramme structuré en branches, où chaque nœud représente une question ou un critère basé sur une variable. À chaque niveau, les données sont divisées en groupes plus homogènes en fonction de ce critère. Ce processus se répète jusqu'à atteindre une condition d'arrêt (par exemple, un seuil minimal d'homogénéité ou une profondeur maximale). Le chemin parcouru mène à une feuille qui contient une prédiction ou une classification.



> **Note** : Arbre de decision est *descriptif* et *predictif*.

Decision :

Selon la description de nos donnees, on peut clairement deduire que KNN et K-Means ne sont pas approprie a etre applique. Puisque KNN et K-Means sont des algorithmes appliques sur des donnees numérique, et pour les donnees numerique il faut un decodage et transformation en utilisant des techniques avances comme la distance de Hamming. Ce qui est complique et rend le processus moins precis.

Pour l'arbre de decision, il consiste a construire un arbre sur des donnees qui sont clairement structures (surtout un mixte de donnees numerique et multi-choix).

Le tableau suivant resume la difference entre les 3 algorithmes :

Méthode	Nature des Données	Prédictif ou Non	Meilleure Adaptation (Numérique)	Meilleure Adaptation (Multi-choix)
KNN	Numérique et catégorique (avec encodage)	Prédictif	Oui	Oui (avec distance adaptée)
KMeans	Principalement numérique	Non (Clustering)	Oui	Non (mais variantes comme K-Modes)
Arbre de décision	Numérique et catégorique	Prédictif	Oui	Oui

Dont il est claire que pour notre cas est **Arbre de decision**.

Creer le modele

Maintenant, il faut qu'on cree un modele d'Arbre de decision en se basant sur notre donnees d'entrainement *(train.csv puisqu'elle contient plus de donnees que test.csv)*.

trainModel.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
import joblib

# Load the dataset
df = pd.read_csv("train.csv")

# Preprocessing the data
# Encode categorical columns
categorical_columns = [
    "Gender", "Job Role", "Work-Life Balance", "Job Satisfaction",
    "Performance Rating", "Overtime", "Education Level", "Marital Status",
    "Job Level", "Company Size", "Remote Work", "Leadership Opportunities",
    "Innovation Opportunities", "Company Reputation", "Employee Recognition", "Attrition"
]

label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Save encoders for prediction use

# Define features and target
X = df.drop(columns=["Attrition", "Employee ID"])  # Remove target and identifier
y = df["Attrition"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the Decision Tree model
model = DecisionTreeClassifier(max_depth=5, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Save the model and label encoders
joblib.dump(model, "decision_tree_model.joblib")
```

```
joblib.dump(label_encoders, "label_encoders.joblib")
print("Model and encoders saved successfully!")
```

On executant le script, le modele va etre créer sous forme de 2 fichiers ***decision_tree_model.joblib*** et ***label_encoders.joblib***.

Ces modeles vont etre utiliser pour faire des predictions sur des nouvelles donnees.

Evaluer le modele

L'evaluation du modele, ca veut dire tester le modele sur des donnees d'entrainement (ou de nouvelles donnees dont on sait a priori la valeur d'attrition), et reperer un pourcentage de precision (accuracy).

Dans notre cas, apres l'execution du modele l'evaluation est faite et le pourcentage est de 70%.

```
youssef@yousbot Attrition % python3 trainModel.py
Accuracy: 0.7059563758389261
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.68      0.69      5667
           1       0.71      0.73      0.72      6253

    accuracy                           0.71     11920
   macro avg       0.71      0.70      0.70     11920
weighted avg       0.71      0.71      0.71     11920
```

Deployer le modele:

Maintenent il faut utiliser le modele qu''on a creer (se basant sur l'arbre de decision) sur un nouveau employe et voir si l'employe va **quitter ou rester** dans l'entreprise.

Le script *predictModel.py* va importer le modele qu'on a generer dans l'etape derniere, et l'executer sur un nouveau employe qu'on a generer d'une maniere aleatoire, contenant les donnees suivantes :

```
"Age": 35, "Gender": "Male", "Years at Company": 5, "Job Role": "Technology", "Monthly Income": 7000, "Work-Life Balance": "Good", "Job Satisfaction": "High", "Performance Rating": "High", "Num of Promotions": 1, "Overtime": "Yes", "Distance from Home": 10, "Education Level": "Bachelor's Degree", "Marital Status": "Married", "Num of Dependents": 2, "Job Level": "Mid", "Company Size": "Large", "Company Tenure": 10, "Remote Work": "No","Leadership Opportunities": "Yes", "Innovation Opportunities": "Yes", "Company Reputation": "Good", "Employee Recognition": "High"
```

predictModel.py

```
import pandas as pd
import joblib
from sklearn.metrics import accuracy_score, classification_report
```

```
# Load the saved model and encoders
model = joblib.load("decision_tree_model.joblib")
label_encoders = joblib.load("label_encoders.joblib")

# Load the test dataset
test_df = pd.read_csv("test.csv")

# Preprocess the test data
# Extract features and target
X_test = test_df.drop(columns=["Attrition", "Employee ID"])  # Remove target and
identifier
y_test = test_df["Attrition"]

# Encode categorical columns in the test dataset
for col, le in label_encoders.items():
    if col in X_test.columns:
        X_test[col] = le.transform(X_test[col])
y_test_encoded = label_encoders["Attrition"].transform(y_test)

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test_encoded, y_pred)
print(f"Model Accuracy on test data: {accuracy * 100:.2f}%")

# Generate and display a detailed classification report
print("\nClassification Report:")
print(classification_report(y_test_encoded, y_pred,
target_names=label_encoders["Attrition"].classes_))
```

En executant le script ci-dessus, on recoit la decision de notre modele si l'employee va rester ou quitter l'entreprise, et le % de precision du modele. Dans notre cas on l'essaye sur toute la dataset *test.csv* et on calcule le pourcentage des resultats exactes :

```
youssef@yousbot Attrition % python3 predictModel.py

Model Accuracy on test data: 71.44%
Classification Report:
              precision    recall  f1-score   support

        Left       0.70      0.68      0.69      7032
      Stayed       0.72      0.74      0.73      7868

    accuracy                           0.71     14900
   macro avg       0.71      0.71      0.71     14900
weighted avg       0.71      0.71      0.71     14900
```

Donc, d'une confiande de 71.44% notre modele opere sur de nouvelles donnees.