**Fiche TP 2 :**

**Prétraitement des données (CSV)**

1. Importer les bibliothèques Nécessaires

import pandas as pd

from typing\_extensions import dataclass\_transform

import pandas as pd

from sklearn.feature\_selection import SelectKBest

from sklearn.preprocessing import LabelEncoder

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

from glob import glob

import cv2

1. Chargement et Affichage des données du dataset : **weatherHistory.CSV**

**# Charger les données from drive**

drive.mount("/content/drive")

data = pd.read\_csv('/content/drive/MyDrive/weatherHistory.csv')

print("data : ",data.head())

1. **Analyse de données CSV**

**# Afficher le nombre des lignes et collones**

num\_rows, num\_columns = data.shape

print(f"Number of Rows: {num\_rows}")

print(f"Number of Columns: {num\_columns}")

**# Afficher les attribut/information sur la base de données**

print(data.columns)

print(data.info)

**# Calculer le nombre de valeurs de chaque attribut**

data.count()

**# calculer le nombre d'éléments uniques pour chaque attribut**

data.nunique()

1. **Traitement des valeurs manquantes**

**#afficher les attributs avec des valeurs manquantes**

missing\_attributes = data.columns[data.isna().any()]

print(missing\_attributes)

**# Trouver les valeurs manquantes dans la colonne "'Precip Type'"**

missing\_values = data[data['Precip Type'].isna()].index

**# Afficher les lignes avec des valeurs manquantes**

print((missing\_values).value\_counts)

**#afficher les valeurs de Precip Type avec le nombre de typles de chaque valeur**

data['Precip Type'].value\_counts()

1. **Suppression des attributs**

**# Suppression Attributes (e.g., 'Formatted Date', Loud Cover)**

Date\_column1 = 'Formatted Date'

**data\_Cleaning**=data.drop(columns=[Date\_column1],axis=1)

Date\_column2 = 'Loud Cover'

**data\_Cleaning**=data\_Cleaning.drop(columns=[Date\_column2],axis=1)

data\_Cleaning.columns

1. **Remplir les valeurs manquantes par la valeur la plus fréquente**

for index in data.index:

if data.loc[index, 'Precip Type'] == 2:

data.loc[index, 'Precip Type'] = data['Precip Type'].mode()[0]

data.head()

1. **Transformation des attributs catégoriel en numérique**

**# Transformation attribut catégoriel en numérique**

**def label\_encode\_categorical(df):**

label\_encoder = LabelEncoder()

categorical\_features = df.select\_dtypes(include=['object']).columns

for feature in categorical\_features:

df[feature] = label\_encoder.fit\_transform(df[feature])

**return df**

**data\_Cleaning** =label\_encode\_categorical(**data\_Cleaning**)

print("data after", **data\_Cleaning**.head())

1. **Selection des attributs Pertinents**

**# En supposant que vous avez une variable cible (remplacez 'target\_column' par le nom réel de la colonne)**

target\_column = 'Precip type'

data\_Cleaning\_X =data\_Cleaning.drop(columns=[target\_column])

data\_Cleaning\_Y = data[target\_column]

**# Etape de selection des attributs avec la méthode ANOVA**

# Feature selection using ANOVA F-test

anova\_selector = SelectKBest(score\_func=f\_classif, k=5) **#**

**k** est un paramètre qui détermine combien de caractéristiques seront sélectionnées.

X\_anova = anova\_selector.fit\_transform(data\_Cleaning\_X, data\_Cleaning\_Y)

selected\_features = anova\_selector.get\_support() # Un vecteur booléen qui indique quelles caractéristiques ont été sélectionnées

print("Selected feature using ANOVA:")

print(selected\_features)

selected\_feature\_names = data\_Cleaning\_X.columns[selected\_features]

print("Selected feature names using ANOVA:")

print(selected\_feature\_names)

**# Selection des attributs a base de la matrice de corrélation**

correlation\_matrix = data\_Cleaning.corr()

**# Créez un graphique de matrice de corrélation**

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Matrice de Corrélation")

plt.show()

1. **Normalisation des attributs**

from sklearn.preprocessing import MinMaxScaler

import pandas as pd

# 1. Séparer la variable cible déjà encodée

target\_variable = 'Precip Type'

features = data.drop(columns=[target\_variable])

target = data[target\_variable]  # Déjà encodé

# 2. Normaliser les autres variables

scaler = MinMaxScaler()

features\_normalized = scaler.fit\_transform(features)

# 3. Recréer le DataFrame normalisé

data\_normalized = pd.DataFrame(features\_normalized, columns=features.columns)

# 4. Réintégrer la colonne 'Precip Type' encodée

data\_normalized[target\_variable] = target.values

# 5. Affichage

print("Données normalisées avec 'Precip Type' encodé ajouté :")

print(data\_normalized.head())

**Questions :**

1. Modifiez le type de codage des attributs en utilisant le codage "One-Hot Encoding", puis répétez l'étape de sélection d'attributs.
2. Conservez l'attribut 'Precip Type' et supprimez les lignes correspondant aux valeurs manquantes.
3. Après avoir calculé la matrice de corrélation, affichez les attributs sélectionnés en utilisant un seuil minimal.
4. Refaire l’étape de sélection d’attribut avec la méthode : **chi-squared , Random Forest Classifier.**
5. Appliquer la standardisation
6. Dans deux fichiers csv séparé sauvegardé la base normalisée et la base standardisé.