Python for Data Analysis

Projet

RAKAI Mohamed, RANDRIA Ntsoa, REINSBACH Eliot

Introduction

Notre dataset est un jeu de données de 130 hôpitaux aux USA. Ce dataset réunit les données de 1999 à 2008.

Présentation du dataset

Cette étude s’est axée sur la problématique suivante :

Est-ce que la prise de médicaments permet de réduire les risques de réadmission des patients ?

US hôpitals (1999-2008)

Santé et médecine

Classification, Clustering

Les étapes principales

Loading

- Séparation du fichier en features et en targets
- Nettoyage
- Mapping

Visualization

- Sélection des colonnes
- Groupement entre plusieurs colonnes
- Création de dataframes temporaires

Machine Learning

- Encodage
- Pipeline
- Entrainement

Nettoyage

```
Rate of Missing Values by Columns:
race                        2.233555
gender                      0.000000
age                         0.000000
weight                     96.858479
admission_type_id           0.000000
discharge_disposition_id    0.000000
admission_source_id         0.000000
time_in_hospital            0.000000
payer_code                 39.557416
medical_specialty          49.082208
```

```
df['age']

0          [0-10)
1         [10-20)
2         [20-30)
3         [30-40)
4         [40-50)
             ...
101761    [70-80)
101762    [80-90)
101763    [70-80)
101764    [80-90)
101765    [70-80)
```

```
[58] def make_dict_columns_unique(X):
       dict_columns={}
       for i in X:
         dict_columns[i] = X[i].sort_values().unique()
       return dict_columns
```

```
dict_columns = make_dict_columns_unique(df)
for i in dict_columns.keys():
  print(i,'\n',dict_columns[i],'\n')
```

```
race
 ['AfricanAmerican' 'Asian' 'Caucasian' 'Hispanic' 'Other' nan]

gender
 ['Female' 'Male' 'Unknown/Invalid']

age
 ['[0-10)' '[10-20)' '[20-30)' '[30-40)' '[40-50)' '[50-60)' '[60-70)'
 '[70-80)' '[80-90)' '[90-100)']

admission_type_id
 [1 2 3 4 5 6 7 8]
```

```
def tranche_age(texte):
    nombres = re.findall(r'\d+', texte)
    nombres = list(map(int, nombres))
    return (sum(nombres)/2)
```

Mapping 1

Entrée [286]:

```
premapping_diag = {}

with open("file1.csv", 'r', encoding='ISO-8859-1') as file:
    next(file)  # pour ignorer l'en-tête
    for line in file:
        icd9code, long_description, _ = line.strip().split(',')[:3]
        premapping_diag[icd9code] = long_description


for key, value in premapping_diag.items():
    print (key,value)
```

```
003.9 Salmonella infection - unspecified
004.0 Shigella dysenteriae
004.1 Shigella flexneri
004.2 Shigella boydii
004.3 Shigella sonnei
004.8 Other specified shigella infections
004.9 Shigellosis - unspecified
005.0 Staphylococcal food poisoning
005.1 Botulism food poisoning
005.2 Food poisoning due to Clostridium perfringens (C. welchii)
005.3 Food poisoning due to other Clostridia
```

Mapping 1 (suite)

```python
mapping_diag = {}

for old_key, value in premapping_diag.items():
    new_key = old_key.lstrip("0")  # Supprime les zéros inutiles au début du nombre. exemple : 006 vaudra 6
    nk_v2=re.sub(r'\.0$', '', new_key)# Supprime les '.0' inutiles en fin de chaîne. exemple : 120.0 vaudra 120
    nk_v2=re.sub(r'\.00$', '', nk_v2)# Supprime les '.00' inutiles en fin de chaîne. exemple : 120.00 vaudra 120
    nk_v2=re.sub(r'\.$', '', nk_v2)# Supprime les '.' en fin de chaîne.
    if '.' in nk_v2:
        nk_v2= nk_v2.rstrip("0") #supprime le dernier chiffre quand c'est un zéro si et seulement si le nombre est un nombre à v
    mapping_diag[nk_v2] = value  # Ajoute la nouvelle clé et sa valeur correspondante dans le nouveau dictionnaireif '.' in nk_v


for key, value in mapping_diag.items():
    print (key,value)
```

```
1 Cholera due to vibrio cholerae
1.1 Cholera due to vibrio cholerae el tor
1.9 Cholera - unspecified
2 Typhoid fever
2.1 Paratyphoid fever A
2.2 Paratyphoid fever B
2.3 Paratyphoid fever C
2.9 Paratyphoid fever - unspecified
3 Salmonella gastroenteritis
3.1 Salmonella septicemia
3.2 Localized salmonella infection - unspecified
```

```python
diags_lit = []
mauvaises_valeurs=[]
for value in df['diag_1']:
    if str(value) in mapping_diag.keys():
        diags_lit.append(mapping_diag[value])
    else:
        diags_lit.append(0)
        mauvaises_valeurs.append(value)



df['diag_lit_1'] = diags_lit

print((df['diag_lit_1'] == 0).sum())
valeurs_uniques = set(mauvaises_valeurs) #on convertit en set car il n'y a pas 2 fois la même valeur dans un set
print(valeurs_uniques)
```

```
4908
{'275', '558', '445', '584', '690', '444', '787', 'V71', 'V54', '350', '959', '58', 'V53', 'V25', '405', '?', '187', '323', '64
5', '799', '780', '362', '585', '790', '284'}
```

Mapping 2

Entrée [301]:
```
data = pd.read_csv('IDS_mapping.csv')
```

Entrée [302]:
```
data.columns
```

Out[302]:
```
Index(['admission_type_id', 'description'], dtype='object')
```

on va d'abord récupérer les données pour l'admission (9 premières lignes)

Entrée [303]:
```
donnees = data.iloc[0:8]

admission = {}
for index, row in donnees.iterrows():
    admission[row['admission_type_id']] = row['description']

print(admission)
```

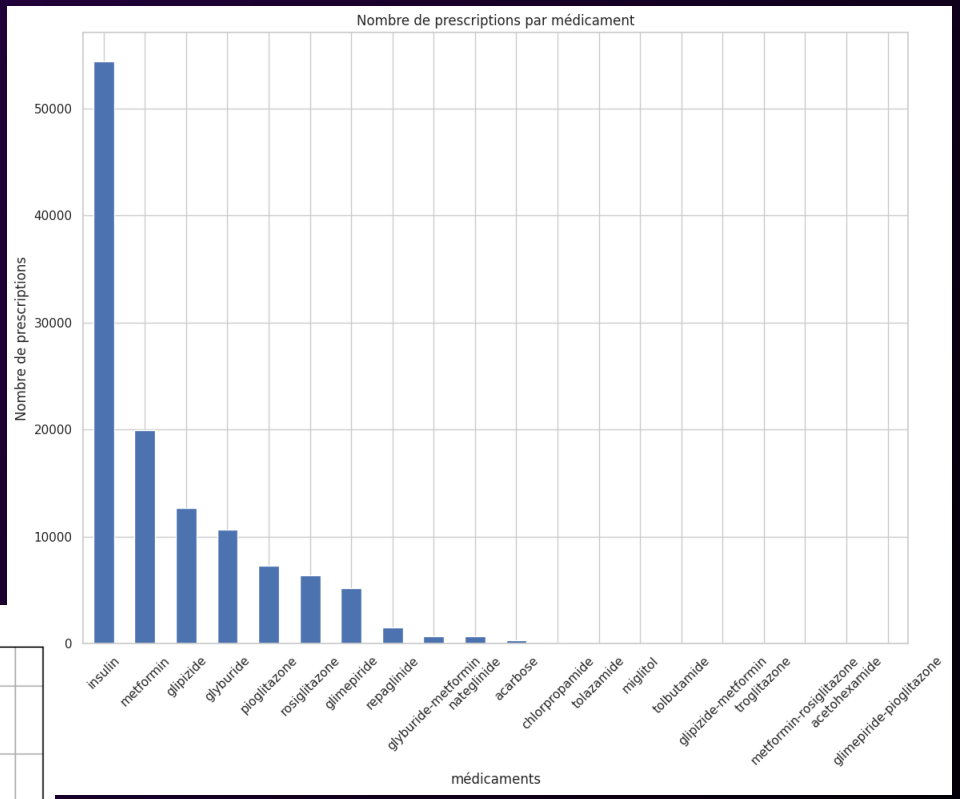
```
{'1': 'Emergency', '2': 'Urgent', '3': 'Elective', '4': 'Newborn', '5': 'Not Available', '6': nan, '7': 'Trauma Center', '8': '
Not Mapped'}
```

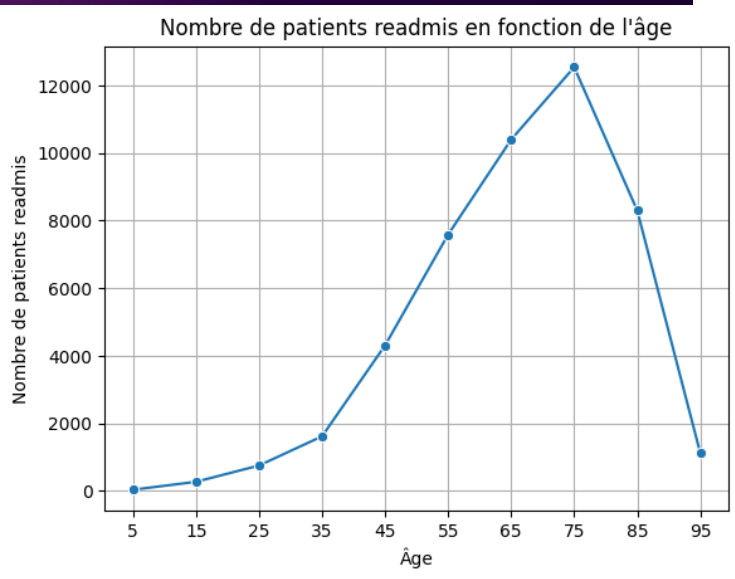
Mapping 2 (suite)

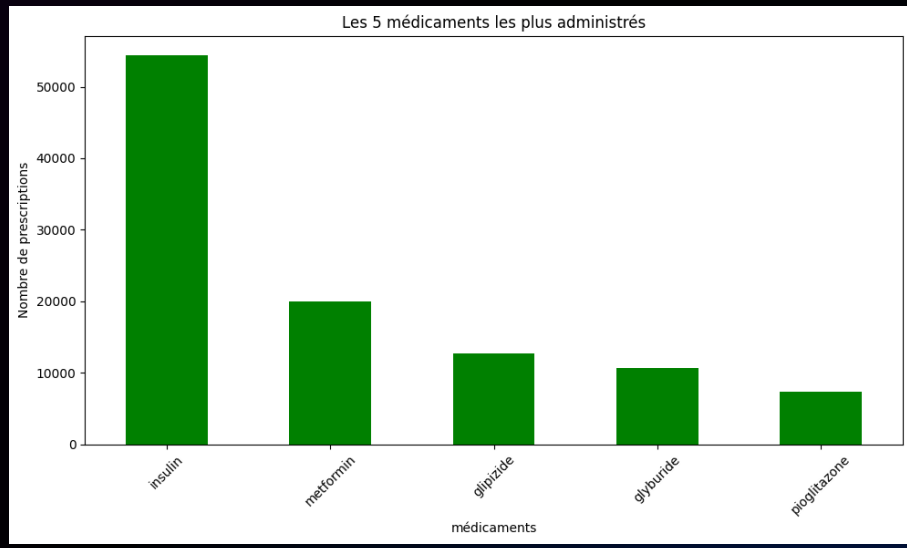
```
admissions_lit = []
mauvaises_valeurs=[]
for value in df['admission_type_id']:
    if str(value) in admission.keys():
        admissions_lit.append(admission[str(value)])
    else:
        admissions_lit.append(0)
        mauvaises_valeurs.append(value)



df['admission_type'] = admissions_lit

# on vérifie les valeurs manquantes
valeurs_uniques = set(mauvaises_valeurs) #on convertit en set car il n'y a pas 2 fois la même valeur dans un set
print(valeurs_uniques)
```

```
set()
```

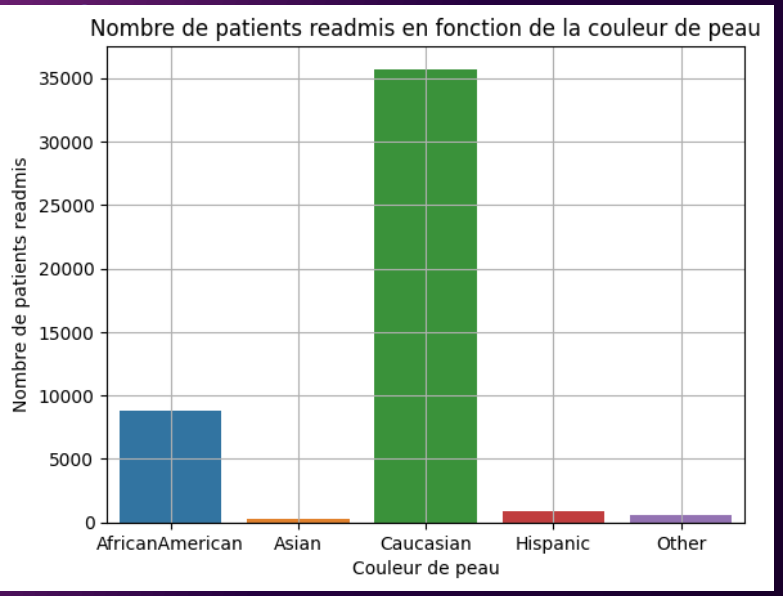
Graphiques 1

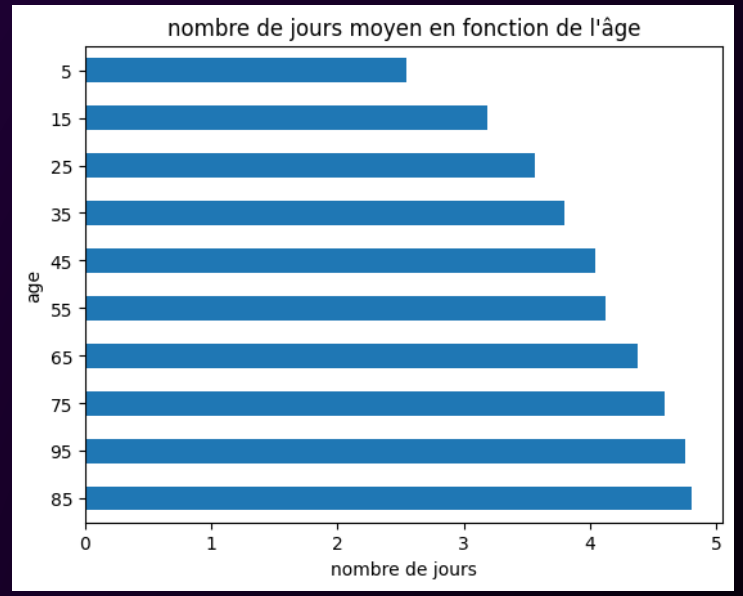


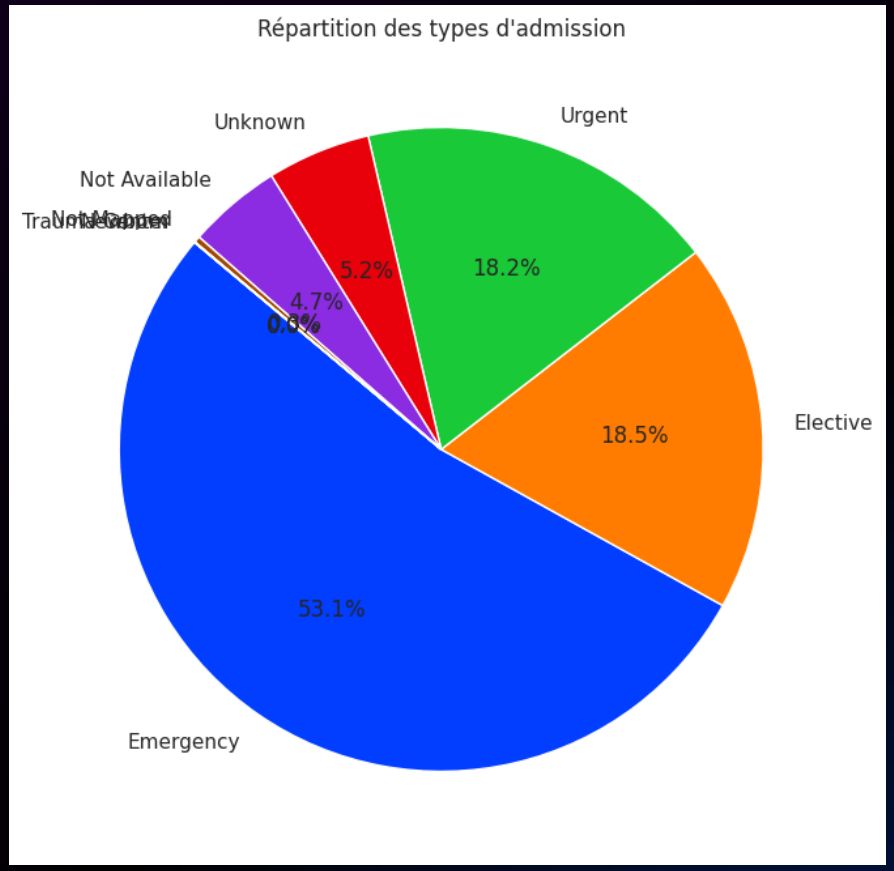


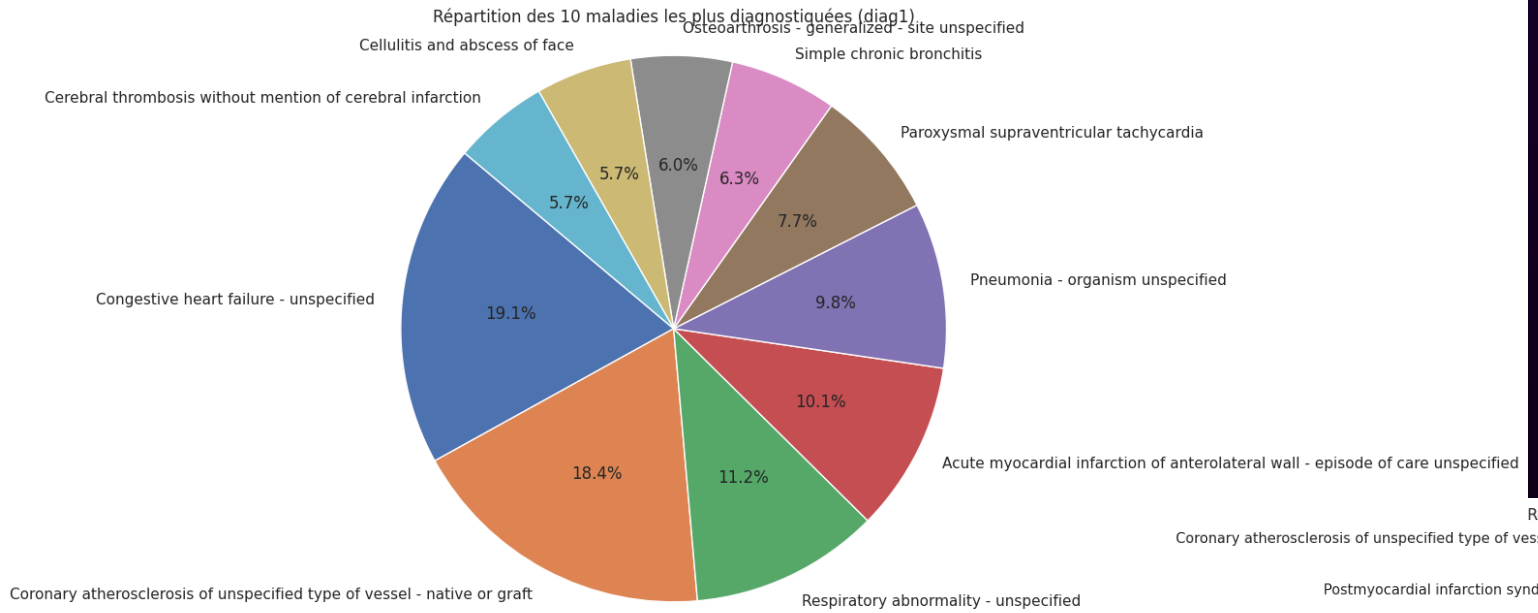


Graphiques 2

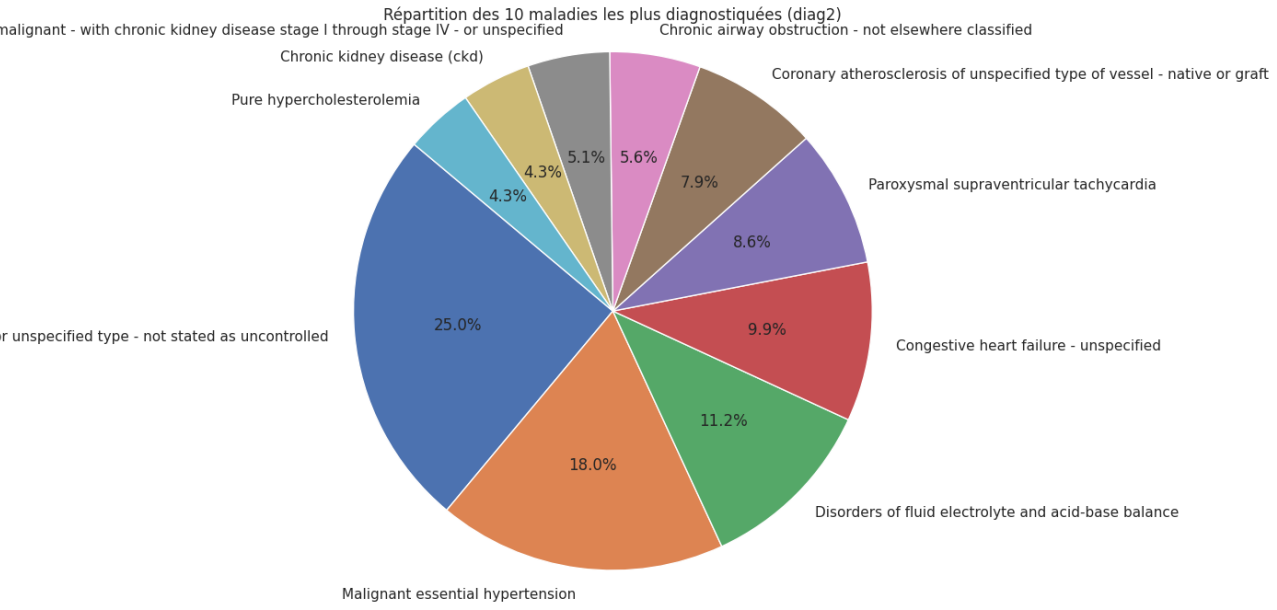


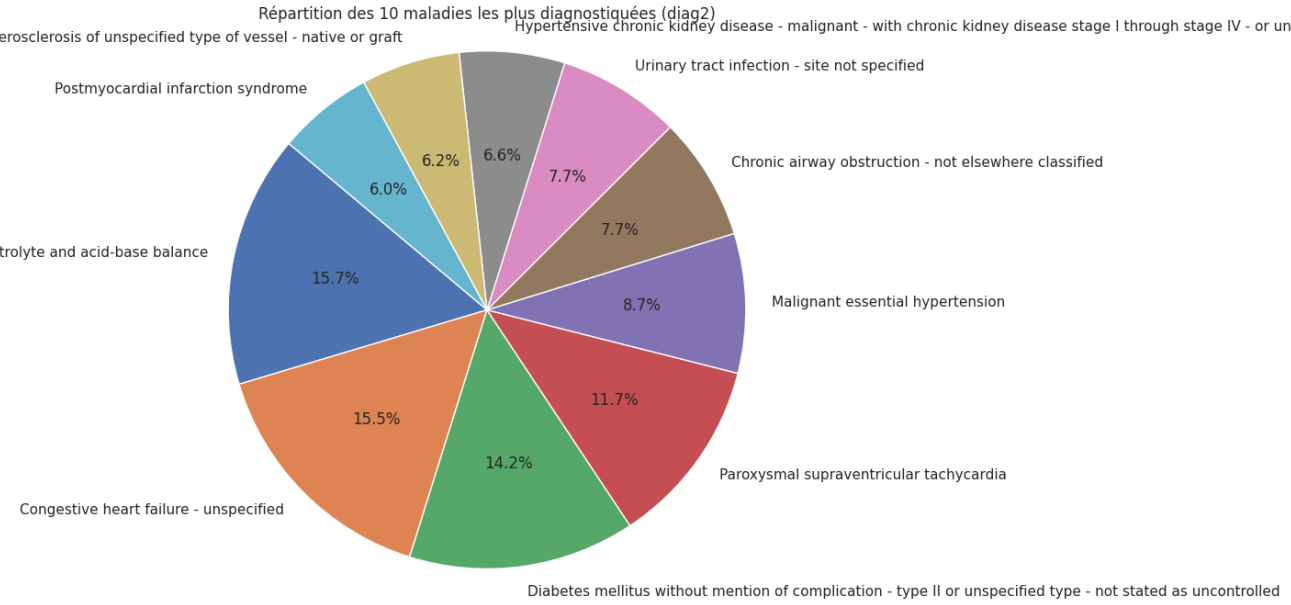




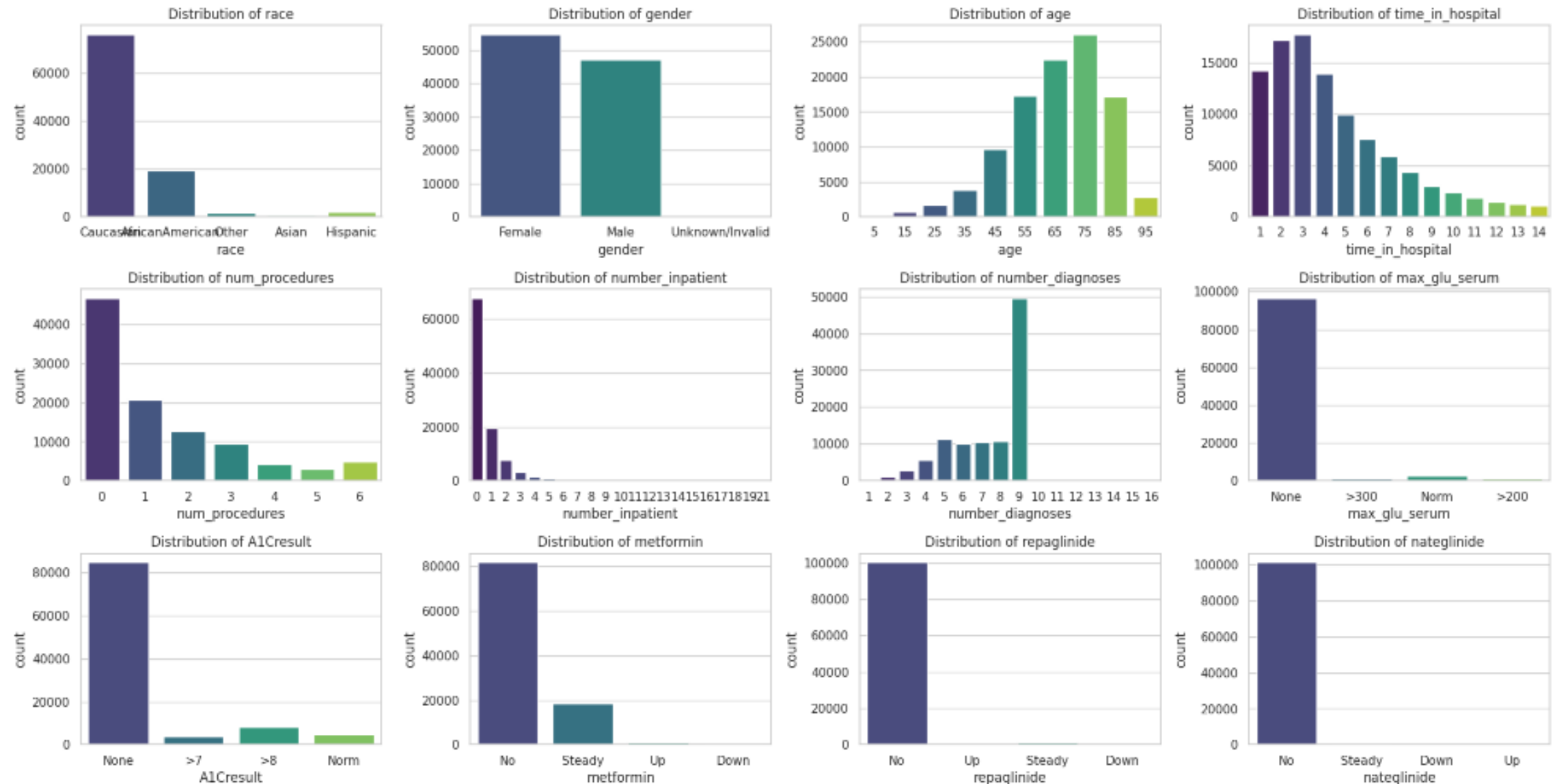

Répartition des 10 maladies les plus diagnostiquées (diag1)
Osteoarthrosis - generalized - site unspecified
Cellulitis and abscess of face
Simple chronic bronchitis
Cerebral thrombosis without mention of cerebral infarction
Paroxysmal supraventricular tachycardia
5.7%
5.7%
6.0%
6.3%
7.7%
Congestive heart failure - unspecified
Pneumonia - organism unspecified
19.1%
9.8%
10.1%
Acute myocardial infarction of anterolateral wall - episode of care unspecified
18.4%
11.2%
Coronary atherosclerosis of unspecified type of vessel - native or graft
Respiratory abnormality - unspecified


Disorders of fluid electrolyte and acid-base balance

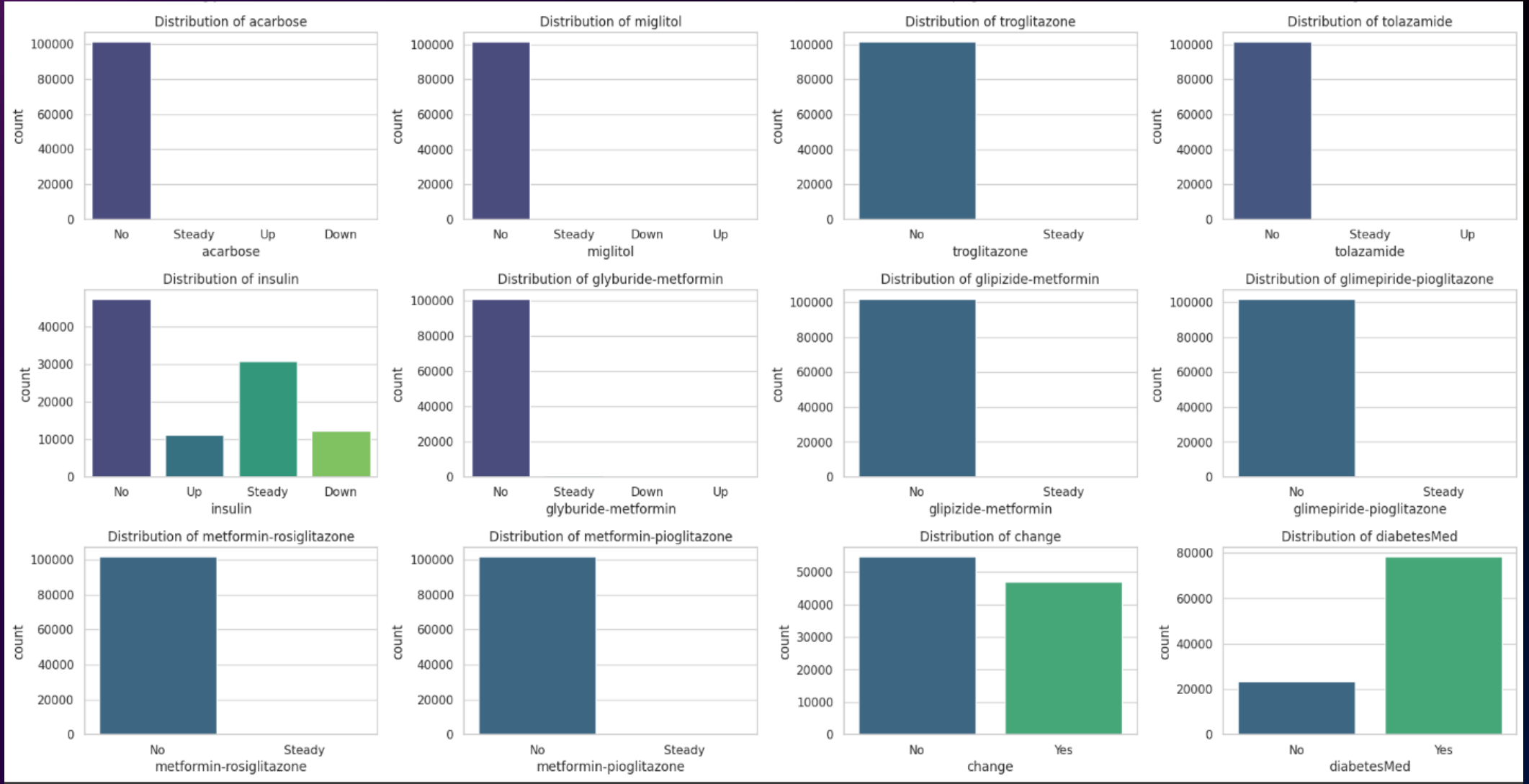

Répartition des 10 maladies les plus diagnostiquées (diag2)
malignant - with chronic kidney disease stage I through stage IV - or unspecified
Chronic airway obstruction - not elsewhere classified
Chronic kidney disease (ckd)
Coronary atherosclerosis of unspecified type of vessel - native or graft
Pure hypercholesterolemia
4.3%
4.3%
5.1%
5.6%
7.9%
Paroxysmal supraventricular tachycardia
8.6%
or unspecified type - not stated as uncontrolled
25.0%
9.9%
Congestive heart failure - unspecified
11.2%
18.0%
Disorders of fluid electrolyte and acid-base balance
Malignant essential hypertension


Répartition des 10 maladies les plus diagnostiquées (diag2)
Coronary atherosclerosis of unspecified type of vessel - native or graft
Hypertensive chronic kidney disease - malignant - with chronic kidney disease stage I through stage IV - or un
Urinary tract infection - site not specified
Postmyocardial infarction syndrome
6.0%
6.2%
6.6%
7.7%
Chronic airway obstruction - not elsewhere classified
7.7%
15.7%
8.7%
Malignant essential hypertension
11.7%
15.5%
14.2%
Paroxysmal supraventricular tachycardia
Congestive heart failure - unspecified
Diabetes mellitus without mention of complication - type II or unspecified type - not stated as uncontrolled

Résumé graphiques 1



Résumé graphiques 2



Ordinal encoding vs onehot encoding

id	color
1	red
2	blue
3	green
4	blue

One Hot Encoding

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Ordinal Encoding

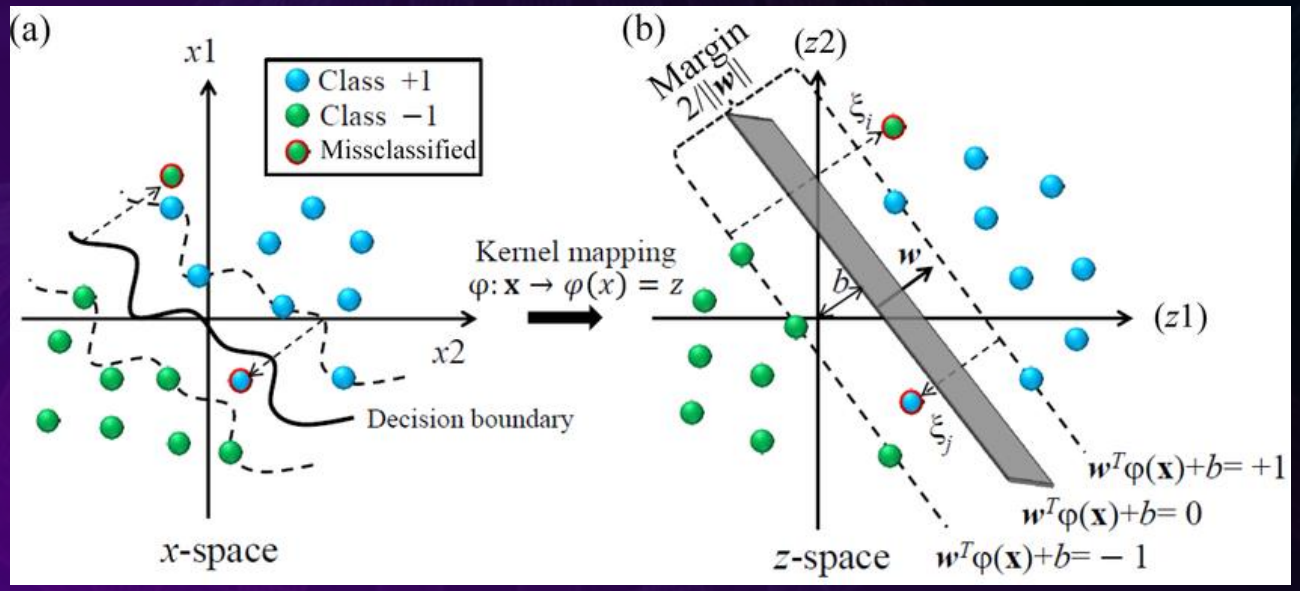
Grades
A
B
C
D
Fail

Grades	Encoded
A	4
B	3
C	2
D	1
Fail	0

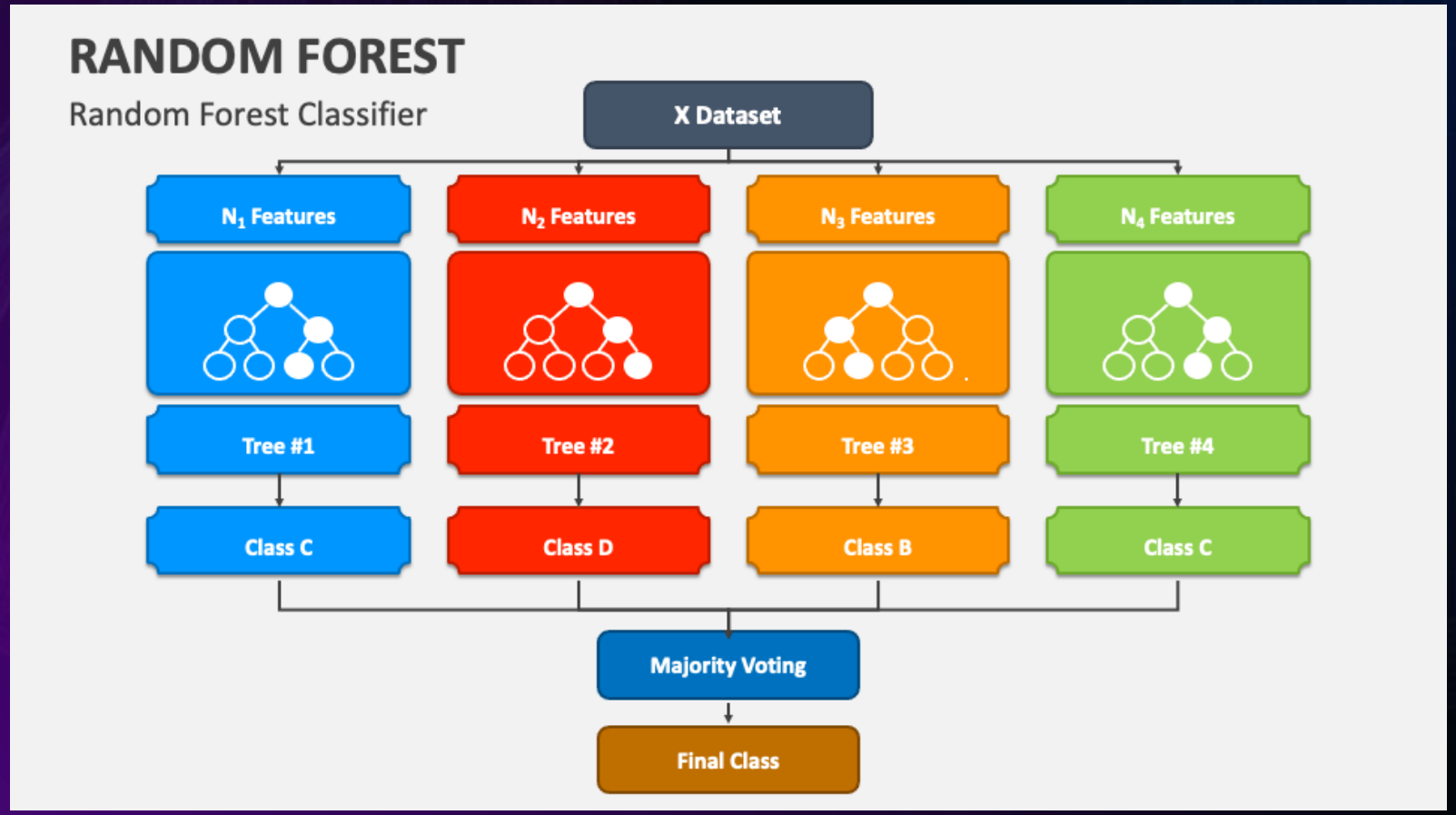
Algorithmes d'apprentissage supervisé

SMV vs Random Forest vs KNN

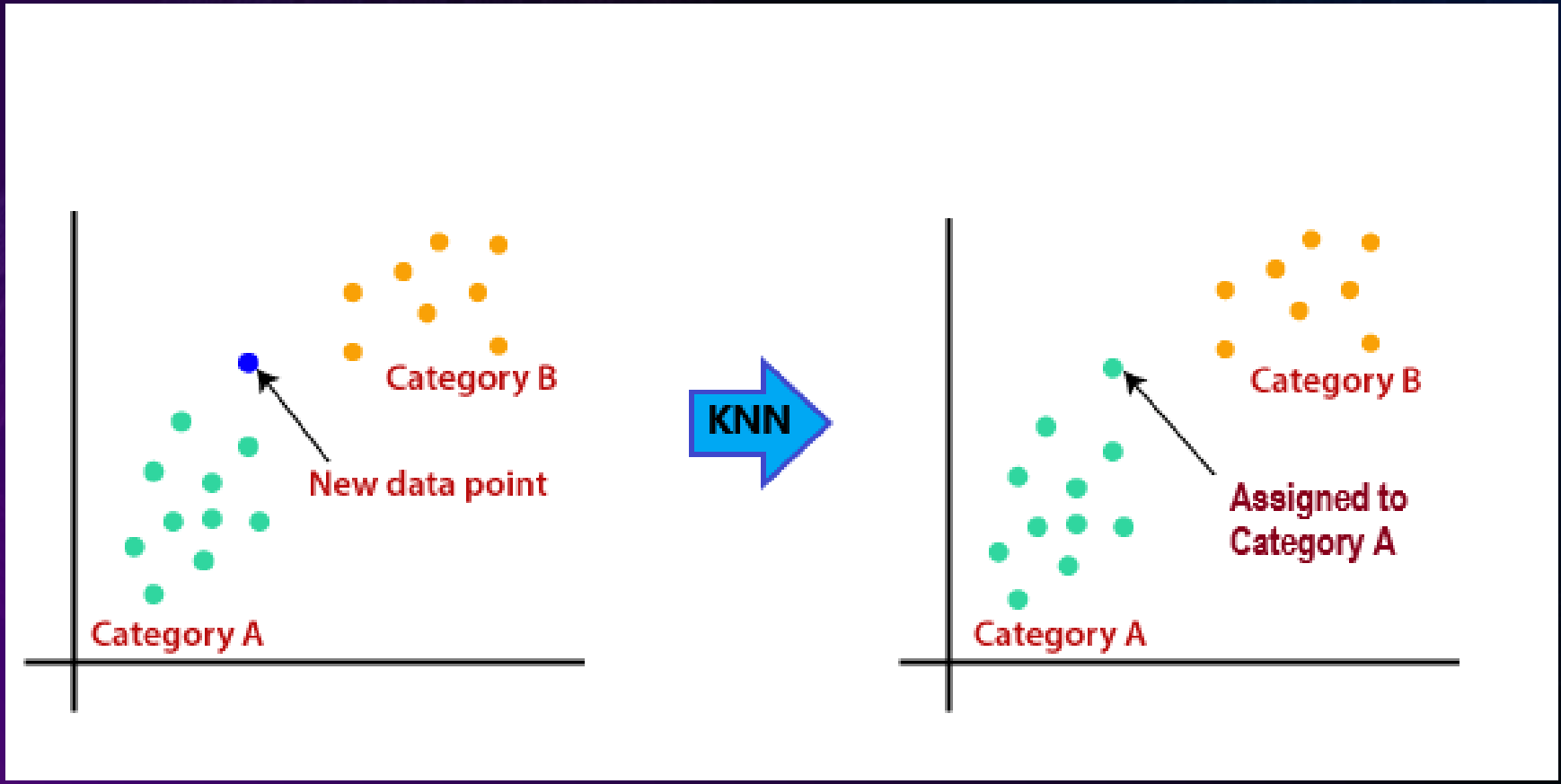
Support Vector Machine (SVM)



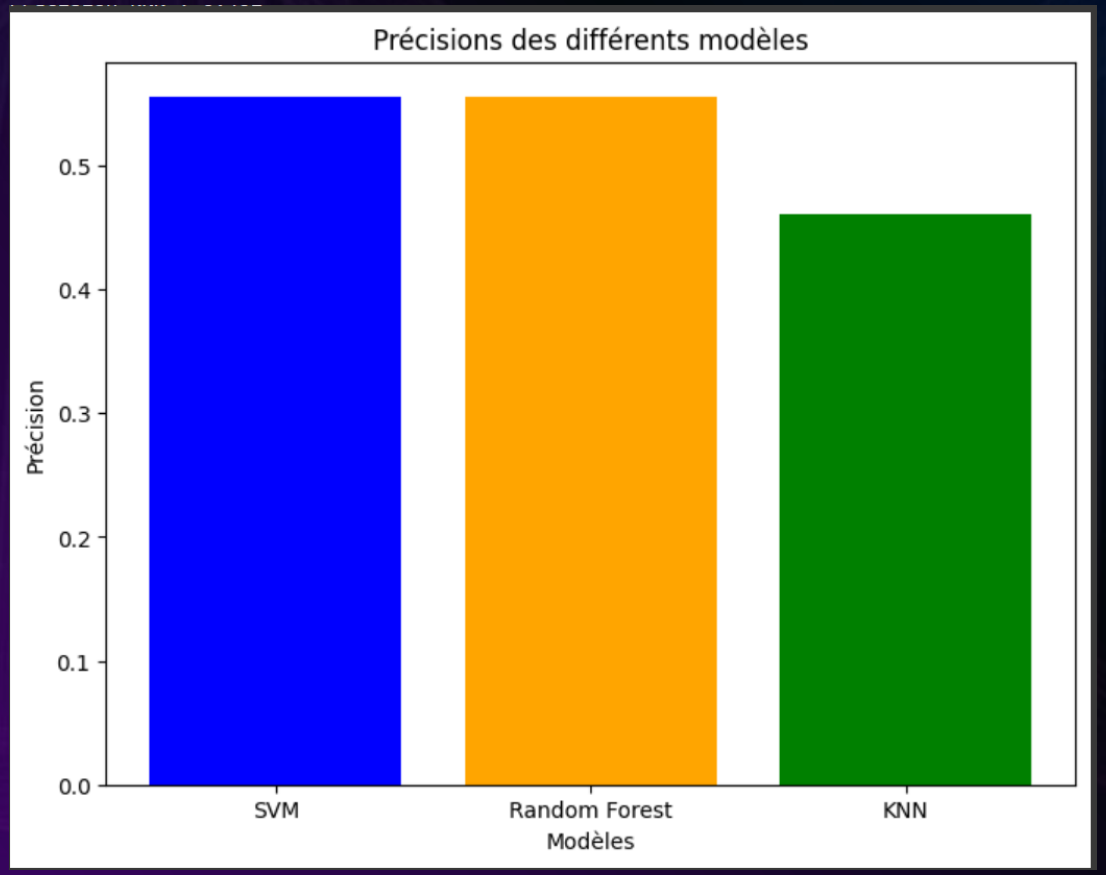
Random Forest



K-Nearest Neighbors (KNN)



Résultats


Précisions des différents modèles
Précision
0.0
0.1
0.2
0.3
0.4
0.5
SVM
Random Forest
KNN
Modèles

Conclusion

Merci