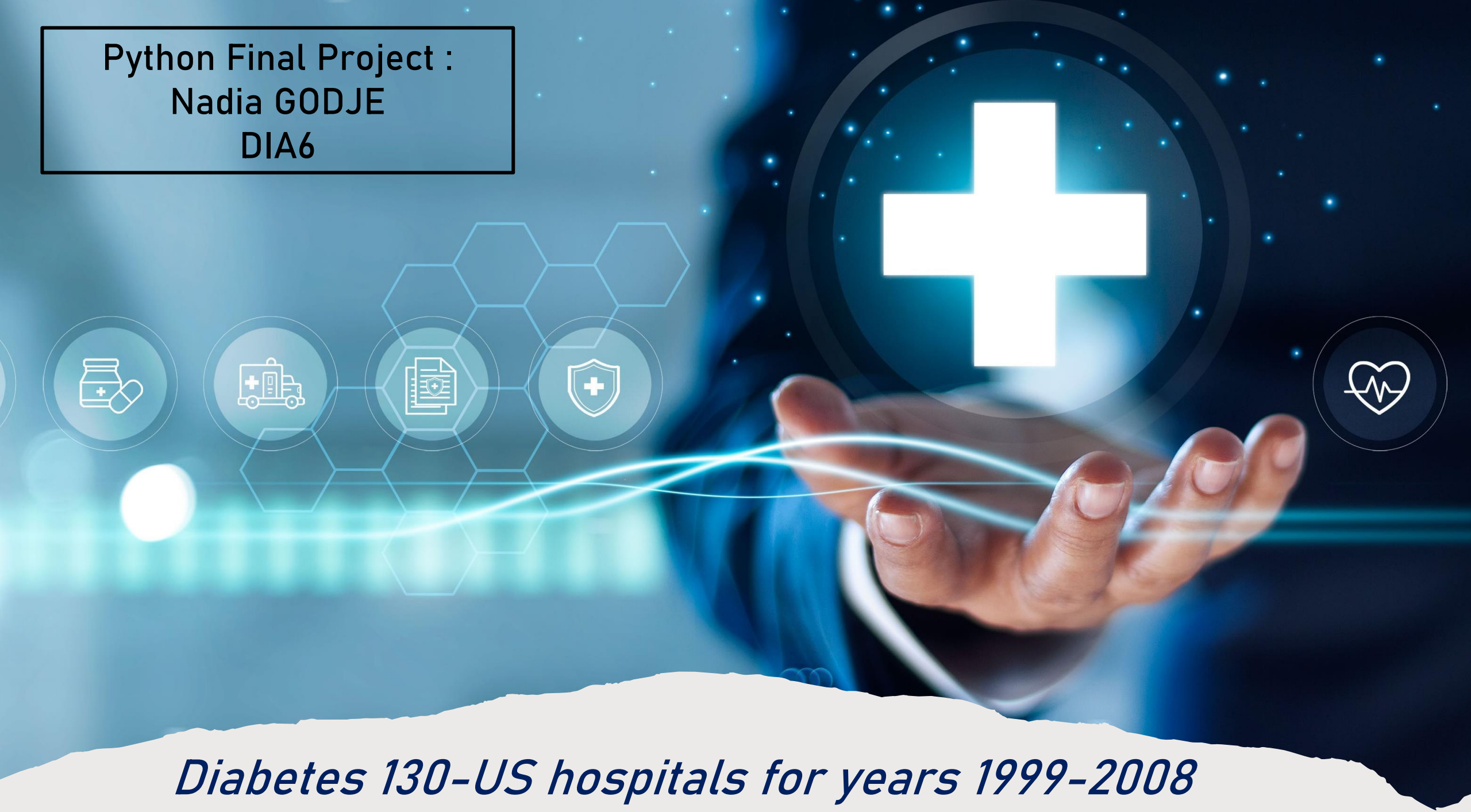
Python Final Project :
Nadia GODJE
DIA6



Diabetes 130-US hospitals for years 1999-2008

The dataset represent hospitalized patient records diagnosed with diabetes.

The goal is to determine the early readmission of the patient within 30 days of discharge.

The dataset contains 101,766 rows and 50 columns.

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	
...	...	...	...	...	...	...	...	...	
101761	443847548	100162476	AfricanAmerican	Male	[70-80)	?	1	3	
101762	443847782	74694222	AfricanAmerican	Female	[80-90)	?	1	4	
101763	443854148	41088789	Caucasian	Male	[70-80)	?	1	1	
101764	443857166	31693671	Caucasian	Female	[80-90)	?	2	3	
101765	443867222	175429310	Caucasian	Male	[70-80)	?	1	1	

101766 rows × 50 columns

In order to make the data more usable, I did the following modifications…

1 - Data Preprocessing

Replace all '?' with NaN in the DataFrame (it aims us to see better the columns missing some data

```
diabetic_data.replace('?', np.nan, inplace=True)
```

Filter on columns containing less than 50000 missing data and delete rows with missing values in columns specified by the index of the minus_from_50_mile series.

```
moins_de_50_mille=diabetic_data.isna().sum()[(diabetic_data.isna().sum() < 50000) &
(diabetic_data.isna().sum()>0)]
diabetic_data = diabetic_data.dropna(subset=moins_de_50_mille.index)
```

Replace the intervals in the age and weight columns with the integer average

```
age_mapping = {'[70-80)': 75, '[60-70)': 65, '[50-60)': 55, '[80-90)': 85, '[40-50)': 45, '[30-40)': 35, '[90-100)': 95,
'[20-30)': 25, '[10-20)': 15, '[0-10)': 5}
diabetic_data['age'] = diabetic_data['age'].replace(age_mapping)
weight_mapping = {'[75-100)': (75+100)//2, '[50-75)': (50+75)//2, '[100-125)': (100 + 125) // 2, '[125-150)': (125 + 150) //
2, '[25-50)': (25 + 50) // 2, '[0-25)': (0 + 25) // 2, '[150-175)': (150 + 175) // 2, '[175-200)': (175 + 200) // 2}
diabetic_data['weight'] = diabetic_data['weight'].replace(weight_mapping)
diabetic_data['weight'] = pd.to_numeric(diabetic_data['weight'], errors='coerce')
```

Replace missing values with the mean

```
diabetic_data['weight'].fillna(diabetic_data['weight'].mean(), inplace=True)
```

Delete the max_glu-result and A1Cresult columns

```
diabetic_data=diabetic_data.drop(['max_glu_serum','A1Cresult'],axis=1)
```

1 - Data Preprocessing

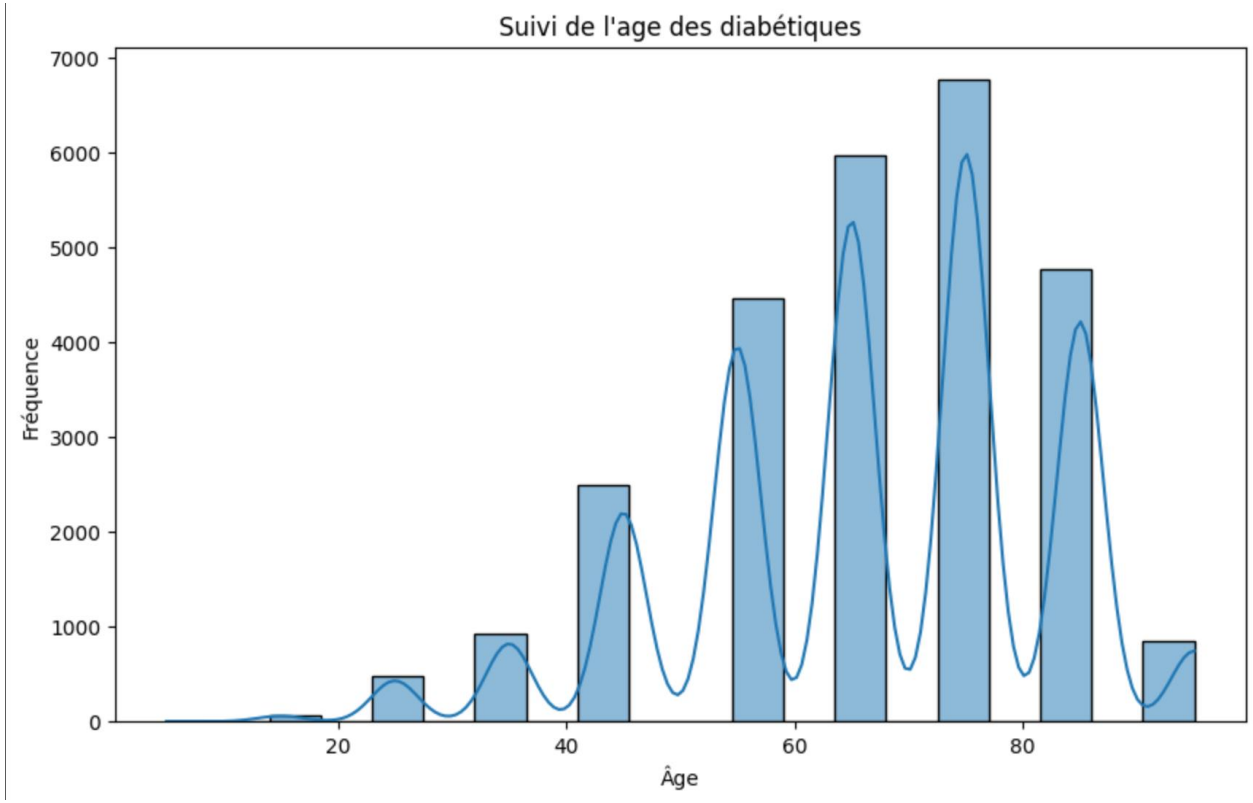
Le dataset nettoyé contient maintenant **26755 lignes** et **48 colonnes**, soit **25 % du dataset initial**. Ce résultat est dû à des données qu'il m'était impossible de prédir avec les données présentes ou trop de données manquantes comme la race du patient, la spécialité du médecin, le code de paiement, max_glu_serum, A1Cresult etc.

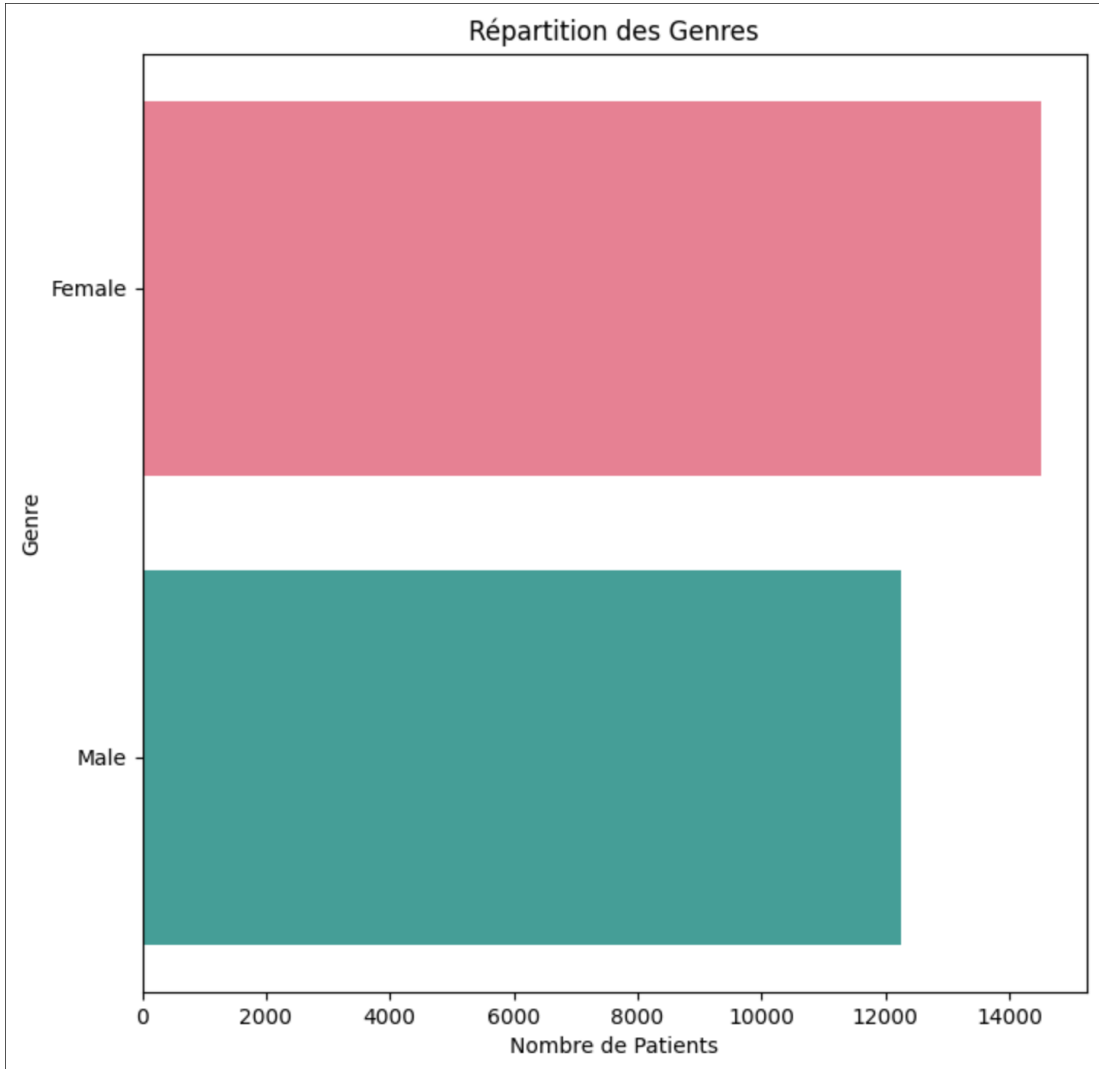
	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admissio
20446	72091308	20123568	Caucasian	Female	75	87.623202	1	22	
20737	72848634	20377854	Caucasian	Female	65	87.623202	2	1	
20824	73062156	20408121	Caucasian	Female	95	87.623202	1	1	
21083	73731852	20542797	Caucasian	Male	75	87.623202	1	2	
23879	81355914	7239654	Caucasian	Female	75	87.623202	1	3	
...	...	...	...	...	...	...	...	...	
101735	443739044	106595208	Caucasian	Male	75	87.623202	2	6	
101743	443793668	47293812	Caucasian	Male	85	87.623202	1	13	
101747	443804570	33230016	Caucasian	Female	75	87.623202	1	22	
101749	443816024	106392411	Caucasian	Female	75	87.623202	3	6	
101764	443857166	31693671	Caucasian	Female	85	87.623202	2	3	

26755 rows × 48 columns

2 - Data Vizualisation

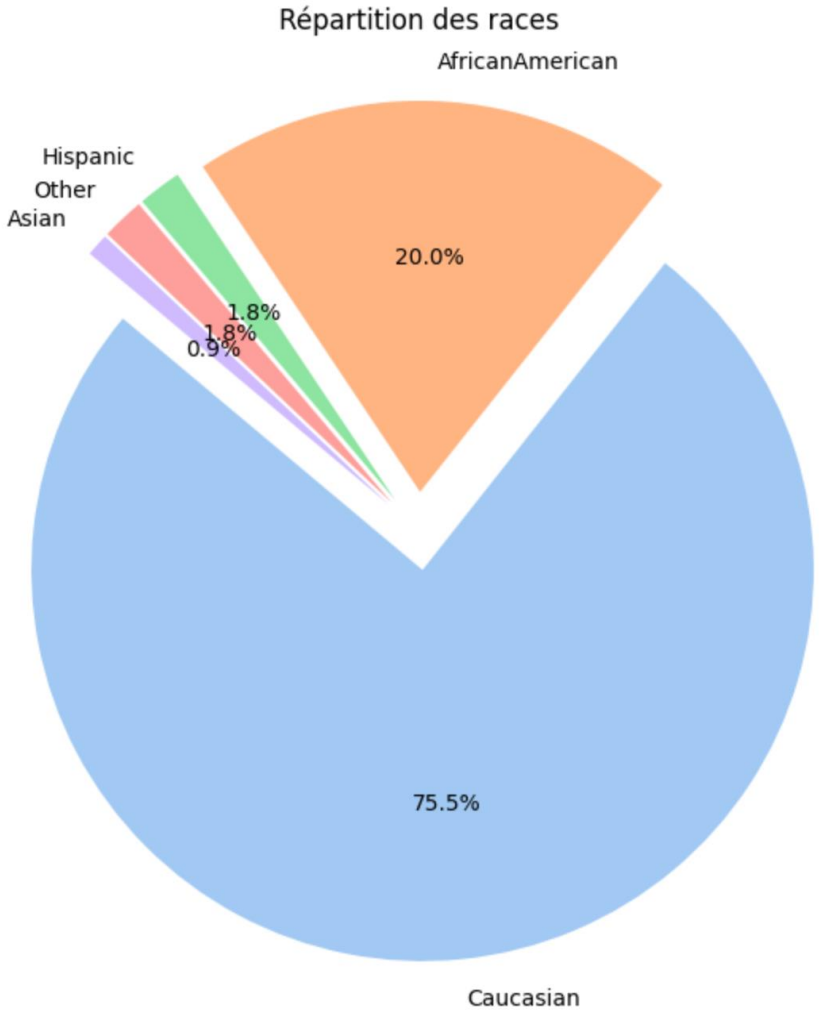
We can now make some visualizations :





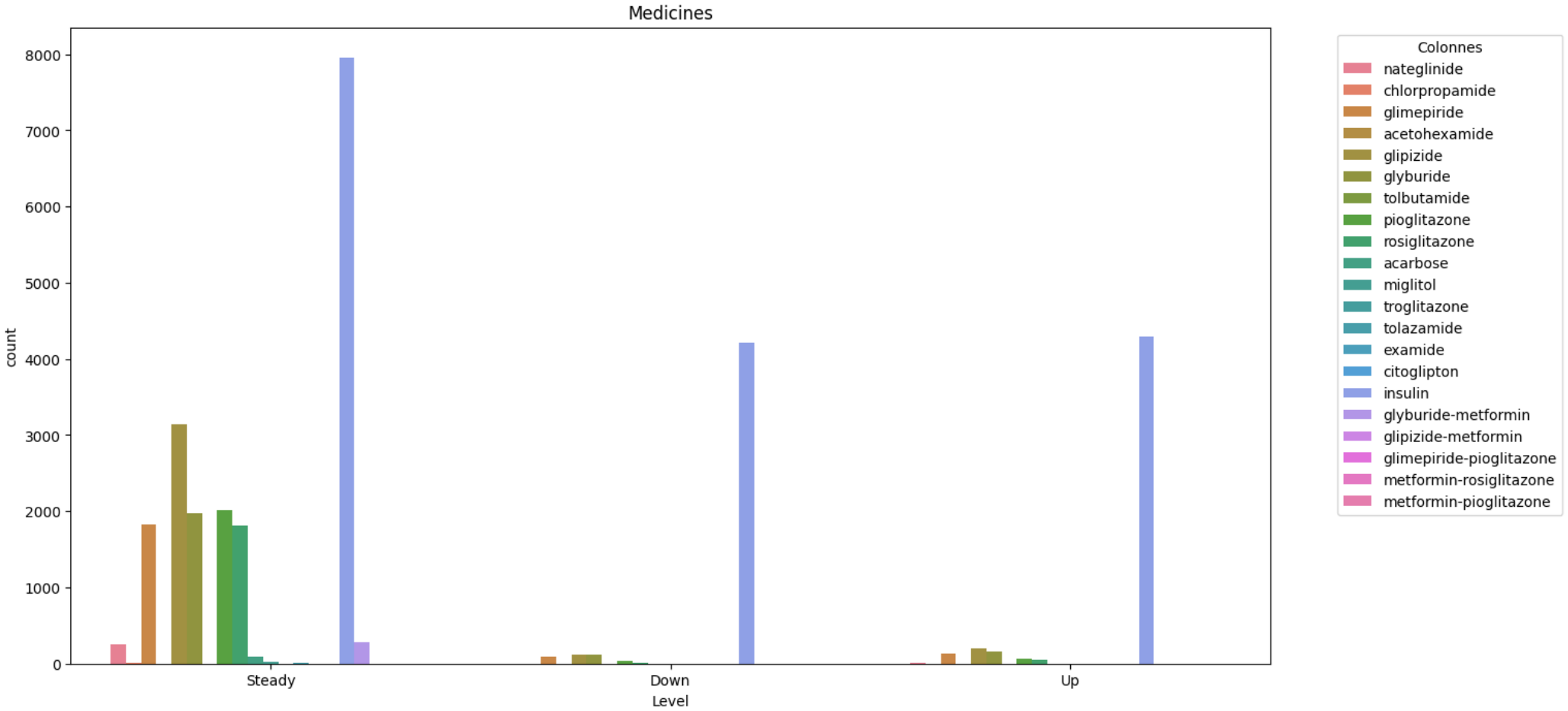
We can note that the elderly and women were the most affected by diabetes between 1999 and 2008 in US hospitals.

2 - Data Vizualisation



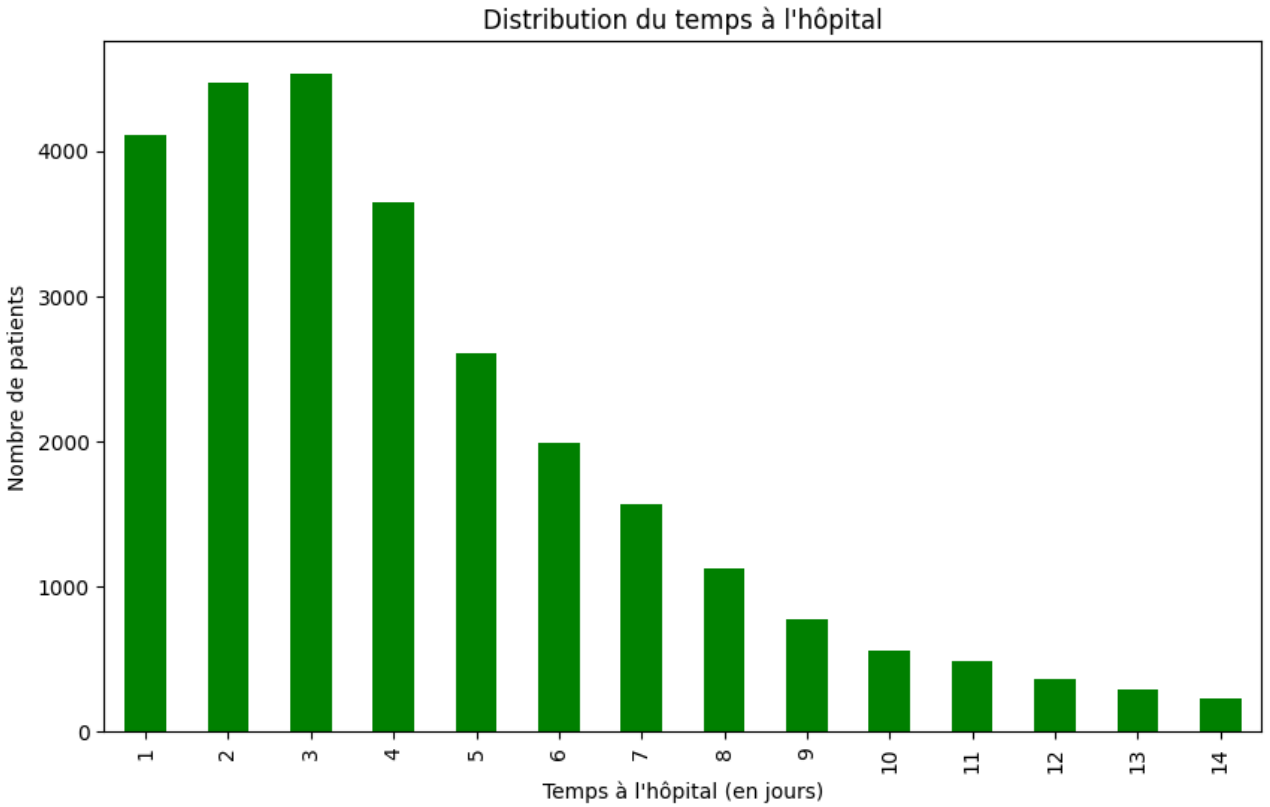
We note that the Caucasian population is largely represented among diabetics treated in the 130 American hospitals between 1999 and 2008. But this distribution must be put into perspective since it illustrates the state of the American population.

2 - Data Vizualisation

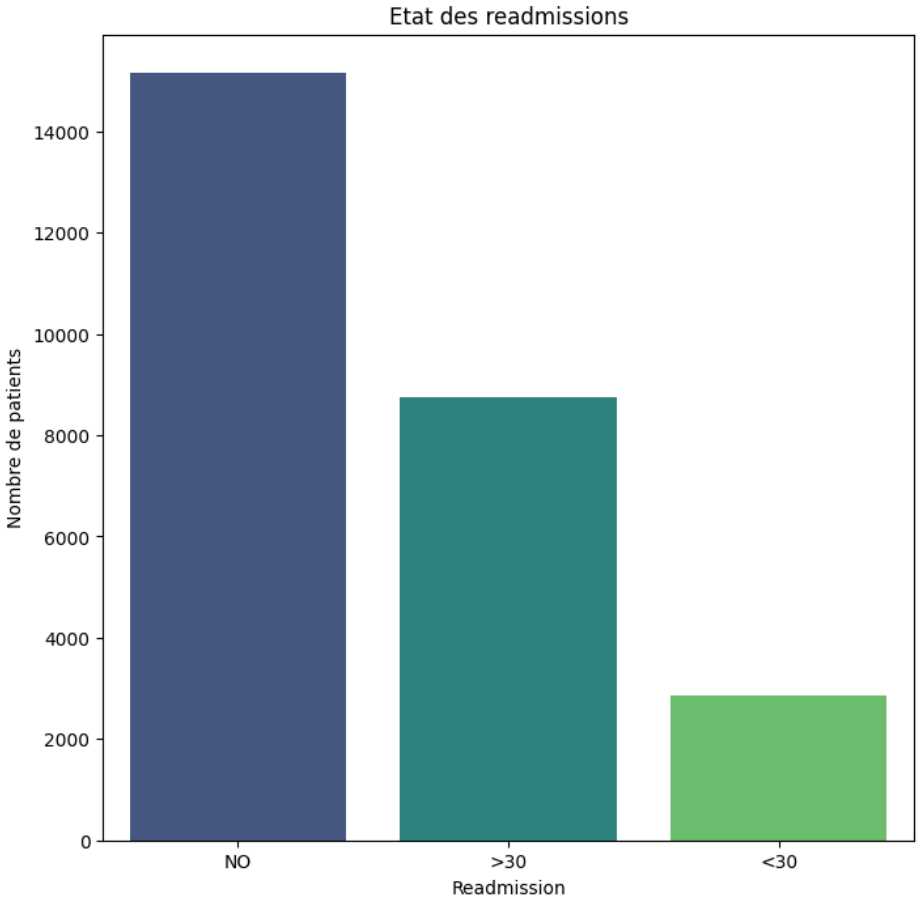


We note that insulin was the most used product in American hospitals between 1999 and 2008 to treat diabetes. This is still the case today.

2 - Data Vizualisation



This graph shows that in general, patients spend very little time in hospital, between 2 and 4 days, and no one spends more than 2 weeks. This shows that this pathology was quite easily treated in American hospitals, without major complications.



As a result, 56.64% of patients are not readmitted, 32.68% are readmitted more than 30 days after discharge, and only 10.68% less than 30 days later. (we'll try to predict next)

1 - Data Preprocessing (the rest and the end)

We divide the dataset into **numerical and categorical** variables

```
categorical_data = diabetic_data.select_dtypes(include=['object'])
numerical_data = diabetic_data.select_dtypes(include=['int64','float64'])
```

Encoding of categorical variables --> ordinal encoding is chosen

```
from sklearn.preprocessing import OrdinalEncoder

encoder = OrdinalEncoder().set_output(transform="pandas")
data_encoded = encoder.fit_transform(categorical_data)
```

We concatenate encoded categorical data with numerical data

```
diabetic_data_enc = pd.concat([numerical_data, data_encoded], axis=1)
```

	encounter_id	patient_nbr	age	weight	admission_type_id	discharge_disposition_id	ad
20446	72091308	20123568	75	87.623202	1	22	
20737	72848634	20377854	65	87.623202	2	1	
20824	73062156	20408121	95	87.623202	1	1	
21083	73731852	20542797	75	87.623202	1	2	
23879	81355914	7239654	75	87.623202	1	3	
...	...	...	...	...	...	...	
101735	443739044	106595208	75	87.623202	2	6	
101743	443793668	47293812	85	87.623202	1	13	
101747	443804570	33230016	75	87.623202	1	22	
101749	443816024	106392411	75	87.623202	3	6	
101764	443857166	31693671	85	87.623202	2	3	

26755 rows × 48 columns

3 - Data Modeling

We define the target column and the features :

```
target_name = "readmitted"
Y = diabetic_data_enc[target_name]
pd.DataFrame(Y).head()
```

✓ 0.0s

	readmitted
20446	2.0
20737	2.0
20824	2.0
21083	2.0
23879	2.0

```
X = diabetic_data_enc.drop(columns=[target_name])
X.head()
```

✓ 0.0s

	encounter_id	patient_nbr	age	weight	admis
20446	72091308	20123568	75	87.623202	
20737	72848634	20377854	65	87.623202	
20824	73062156	20408121	95	87.623202	
21083	73731852	20542797	75	87.623202	
23879	81355914	7239654	75	87.623202	

5 rows × 47 columns

Why is the column ‘readmitted’ our target ?

Predicting readmission is crucial in the medical field. By identifying patients at high risk of readmission, healthcare professionals can take preventive measures, such as post-hospital follow-up, adjusting medications, or scheduling follow-up appointments, to reduce the risk of readmission.

3 - Data Modeling

Next, we separate the encoded data into **test data** and **training data using *train_test_split from sklearn***

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X,Y,
test_size=0.2,random_state=42)
```

The Standardization involves scaling all the characteristics of the dataset to make them comparable. It is useful for improving model performance and stability. (data preprocessing)

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Models type : Classification

The use of classification models seems appripriate here. The ‘readmitted’ column is categorical, indicating whether a patient is readmitted or not. Since we're looking to predict a specific class (readmitted or non-readmitted), classification is the appropriate task.

3 - Data Modeling

We use different training models : RandomForest, KNN, SVM and DecisionTree

We get the best hyperparameters of each model using GridSearch and we use it to train the model on data so that we can have accurencies to compare models performances

❑ Model evaluation : Accurency

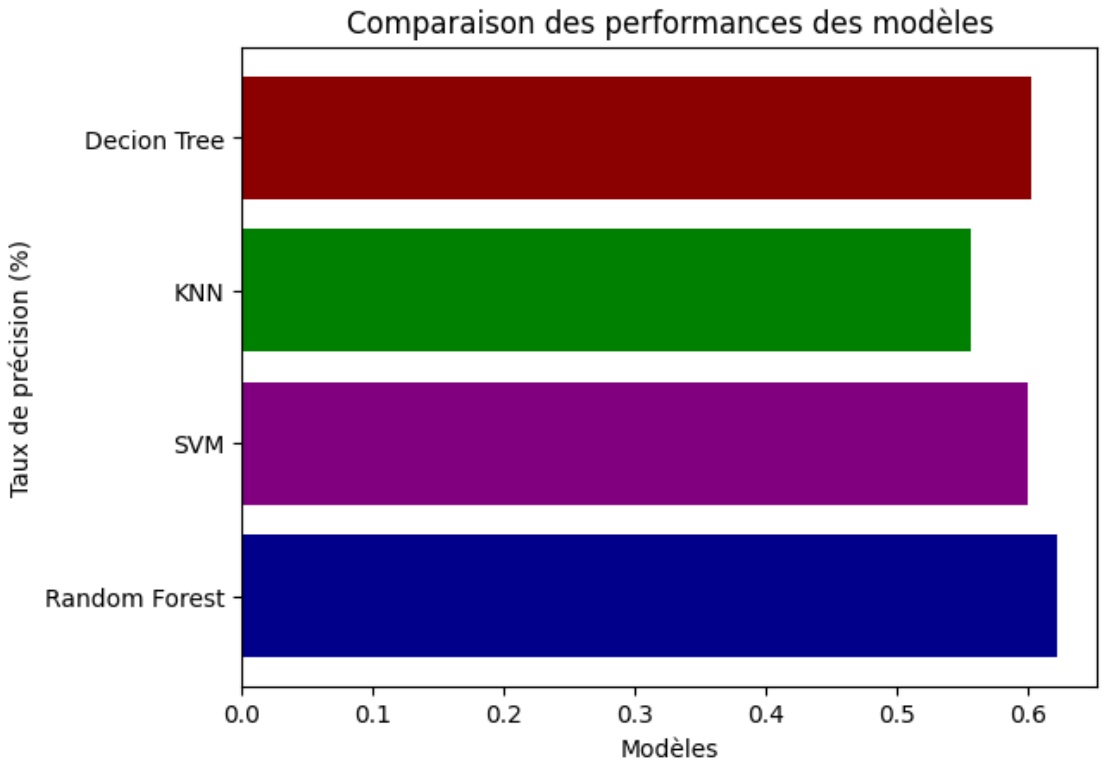
Accuracy measures the proportion of correct predictions among all predictions.

RandomForest → 61.99 %

KNN → 55.69 %

SVM → 60.03 %

Decision Tree → 60.25 %



3 - Data Modeling

❑ Model evaluation : Cross-validation scores

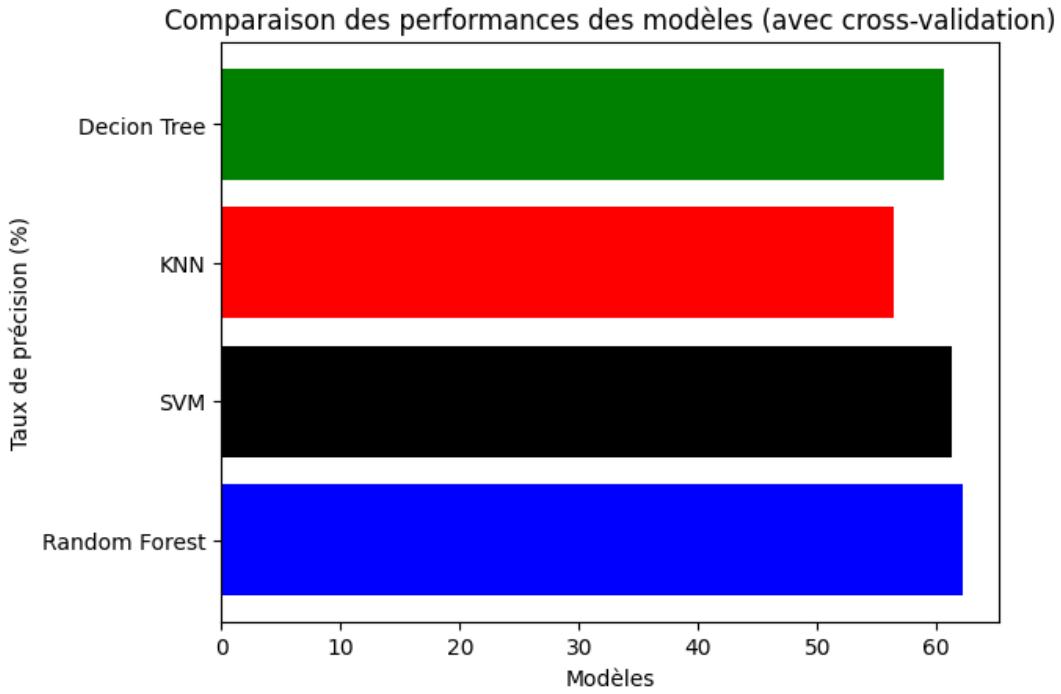
The cross-validation score gives an estimate of a model's performance using a cross-validation technique. It measures model performance on several different training and test data sets.

RandomForest → 62.18 %

KNN → 56.37 %

SVM → 61.24 %

Decision Tree → 60.68 %



Conclusion

Although the RandomForest shows better results than the others, they are still disappointing.

There are several factors that may explain this poor performance, such as the choice of model, hyperparameters, etc. But it's also important to note that readmitted column mainly contains the value 'NO' which can lead to a problem of class imbalance.

As the 'NO' class is dominant, the model may learn to simply predict this majority class to maximize its accuracy, even if it doesn't capture well the cases of the minority classes. This can result in biased model.