Presentation Python for data analysis

Subject:
Diabetes 130 US hospitals for years 1999-2008

Ringeissen Léo - Heckmann Marin - DIA6 2023/2024

Introduction of the dataset

- Shape
  - 50 columns
  - 101766 rows
  - 0 null value

- The data set represents 10 years (1999-2008) of clinical care at 130 US hospitals for diabetic encounters.
- The 50 columns will describe the type of patient, type of stay, the laboratory tests performed, the medications admninistered
- The data have different dtypes: int64 or object.
- The null-values are replaced by « ? »

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_i
0	2278392	8222157	Caucasian	Female	[0-10)	?	
1	149190	55629189	Caucasian	Female	[10-20)	?	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	
3	500364	82442376	Caucasian	Male	[30-40)	?	
4	16680	42519267	Caucasian	Male	[40-50)	?	

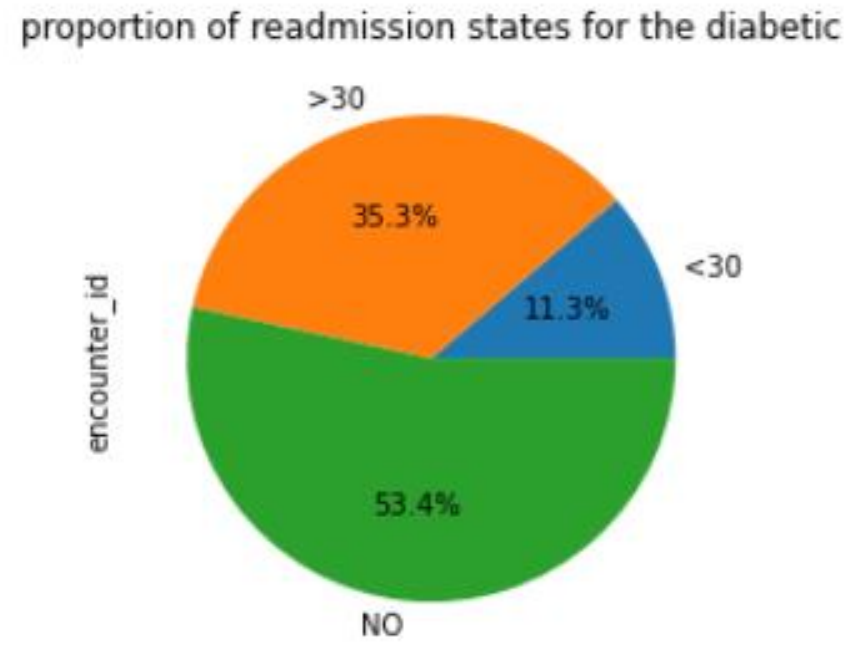
```
Data columns (total 50 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   encounter_id              101766 non-null  int64
 1   patient_nbr               101766 non-null  int64
 2   race                      99493 non-null   object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    3197 non-null    object
 6   admission_type_id         101766 non-null  int64
 7   discharge_disposition_id  101766 non-null  int64
 8   admission_source_id       101766 non-null  int64
 9   time_in_hospital          101766 non-null  int64
 10  payer_code                61510 non-null   object
 11  medical_specialty         51817 non-null   object
 12  num_lab_procedures        101766 non-null  int64
```

Target value: Readmitted

The last column of the dataset is readmitted. It is important to know if a patient will be readmitted in some hospital.

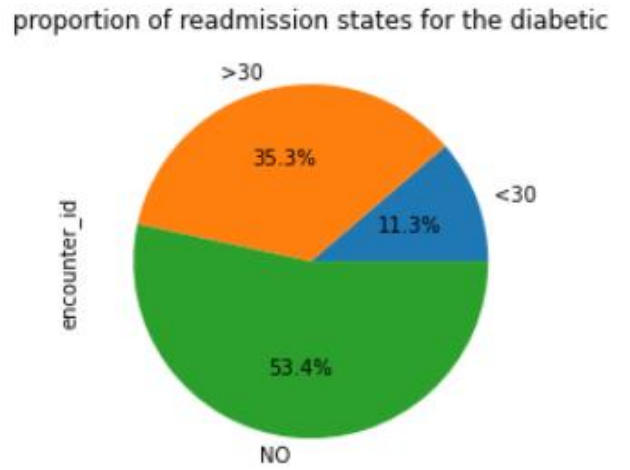
In this database, you have 3 different outputs:

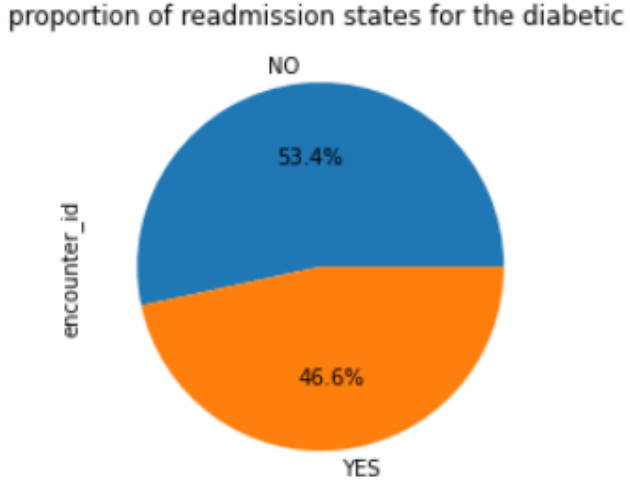
- No readmission;
- A readmission in less than 30 days (this situation is not good, because maybe your treatment was not appropriate);
- A readmission in more than 30 days (this one is not so good as well the last one, however, the reason can be the state of the patient.



Variable changement

- To simplify our analyze, we will gather >30 and <30 together in the category "YES". Our comparison will just be if the patient has been readmitted our not.
- 53,4% of encounter with no readmitting
- 46.6% with readmitting





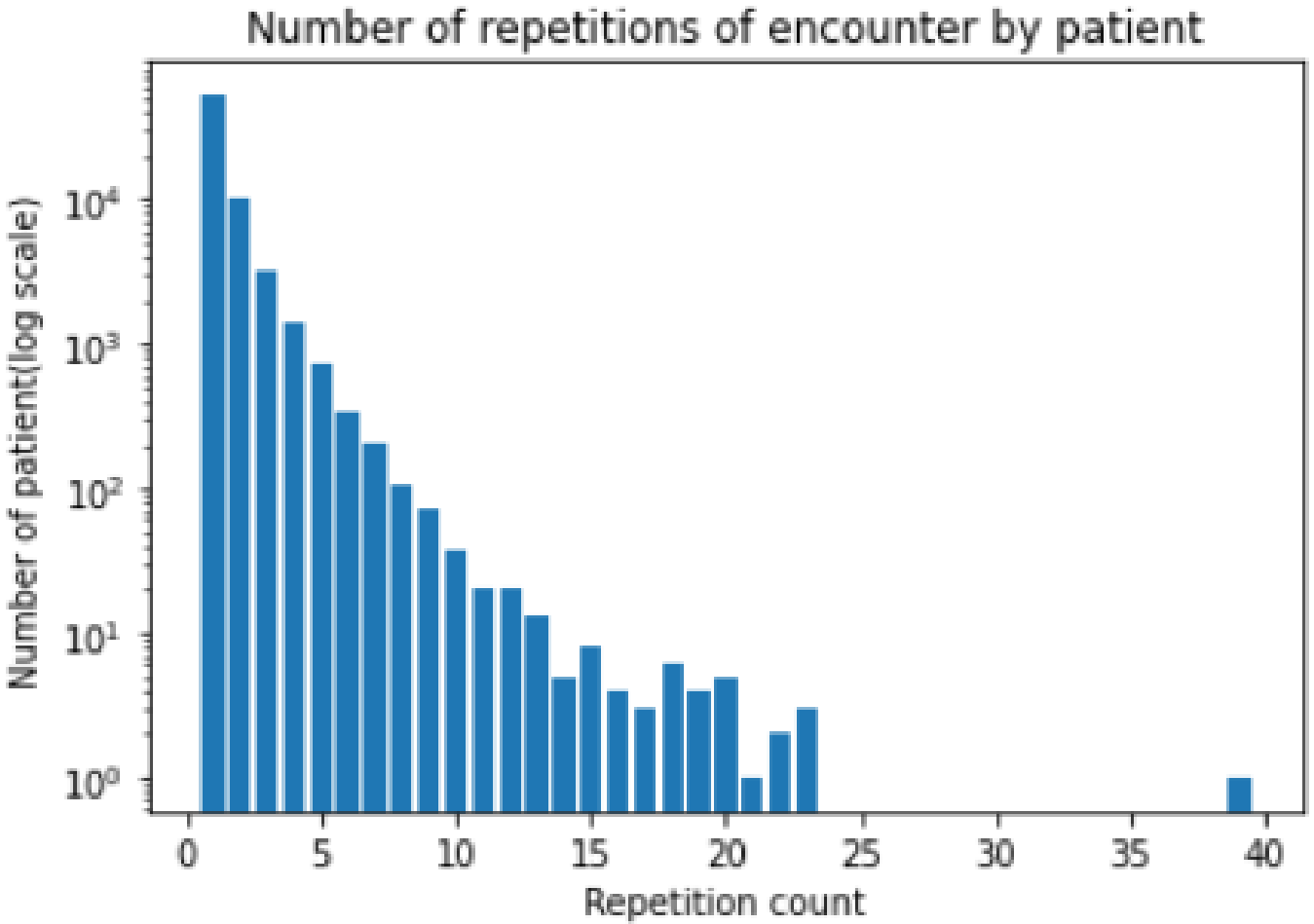
Plan of our study

- Analyse of each variable
- Cleaning the data
- Encoding
- Training the data features to predict the target
- Conclude

→This studies could be really useful because with this training, we could predict if an encounter (thanks all the inputs) will have a readmission

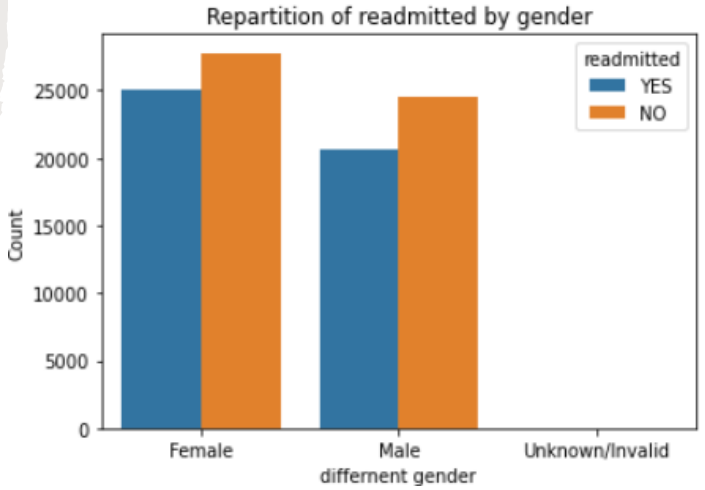
Analyze of ID value

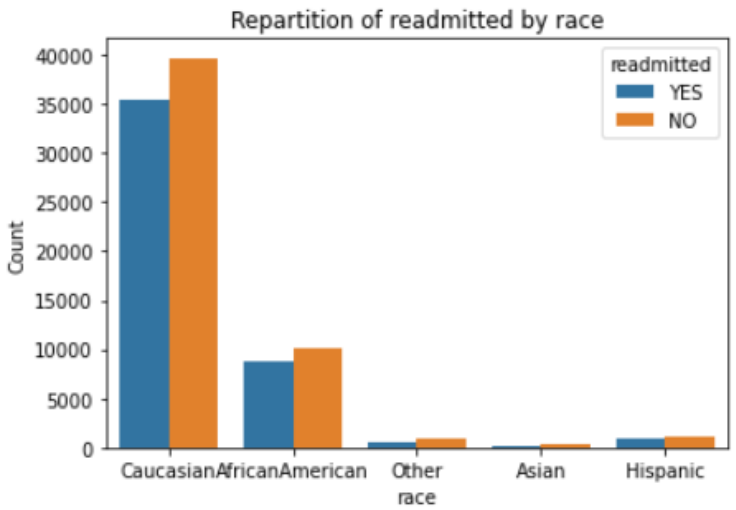
- Encounter_ID,
  - Identifier of the database
  - Value unique
- patient_nbr
  - Identifier of the patient
- The distribution of encounter by patient is particular.(log squale)
- Mostly patient with 1 encounter

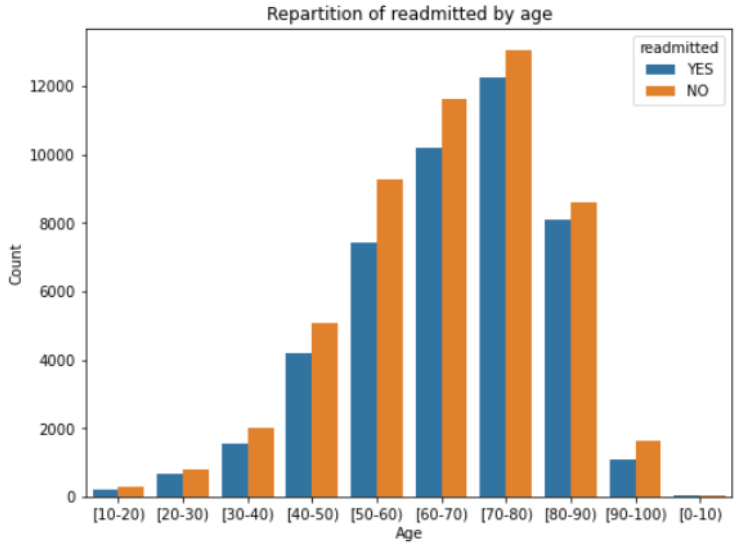


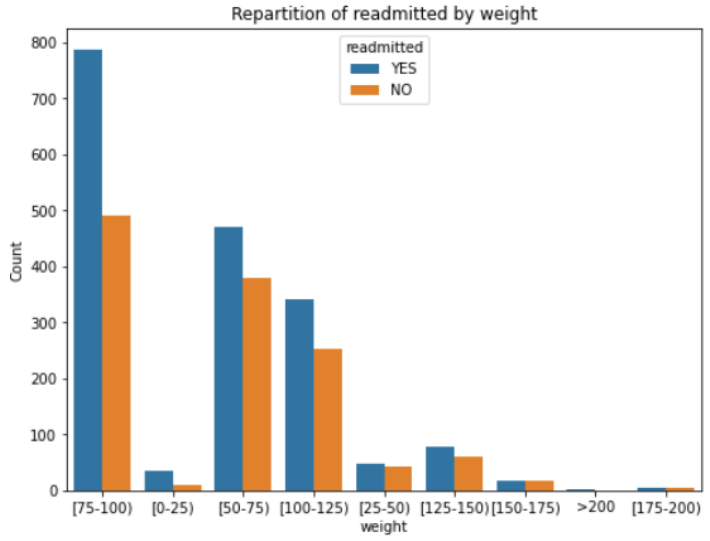
Analyze of Categorical value

- Race,Gender,
  - Same repartition of readmitted
- Age
  - Little correlation
  - Changement in numerical value
- Weight
  - Big correlation
  - Changement in numerical value (kg)
  - Just 3% of the dataset



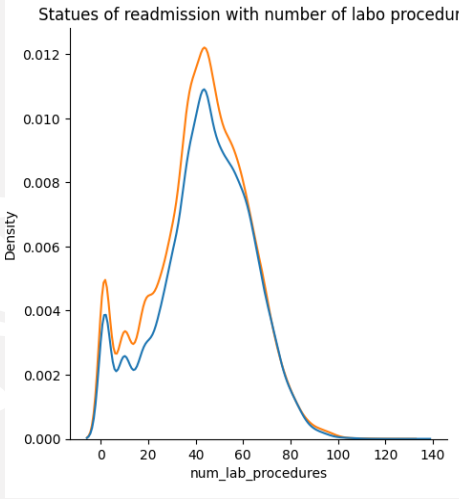


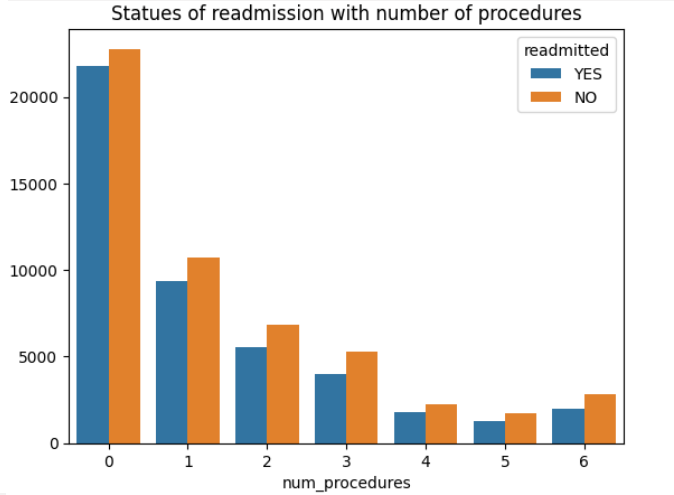


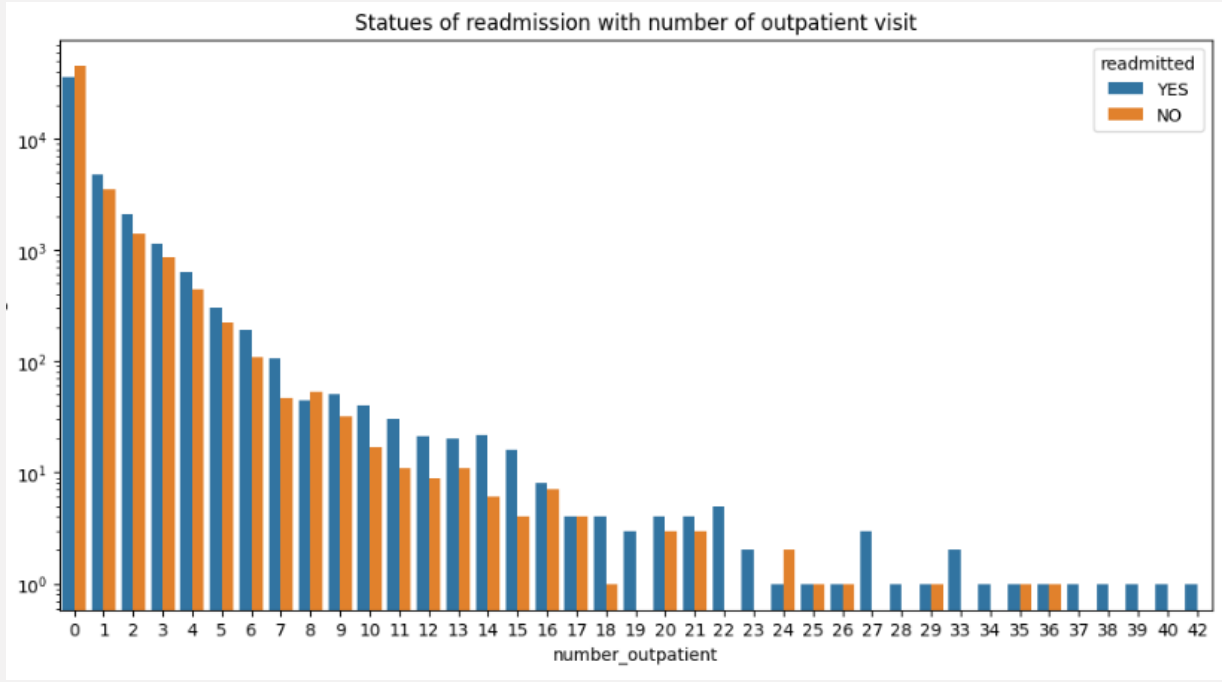


Analyze of numerical value

- multiple columns of numeric value:
- We use to see their distribution according to the target value 3 types of plot:
  - displot with normal scale (Num_lab_procedures and num_procedures)
  - Countplot with normal scale (num_procedures,number diagnoses)
  - Countplot with log scale (number_outpatient, number_emergency,number_inpatient)

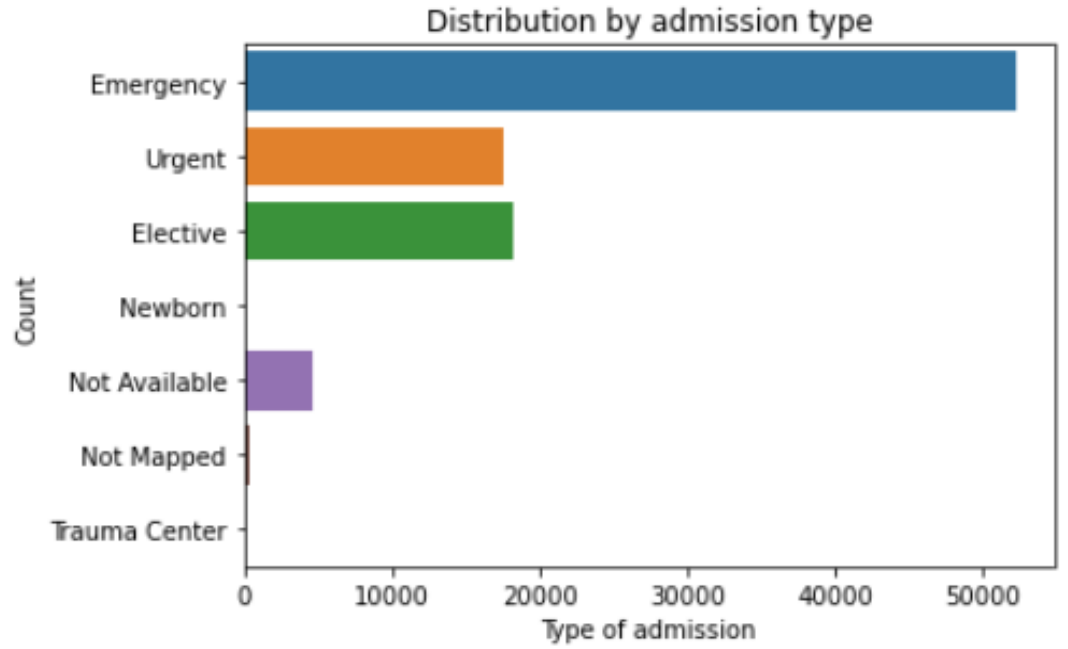


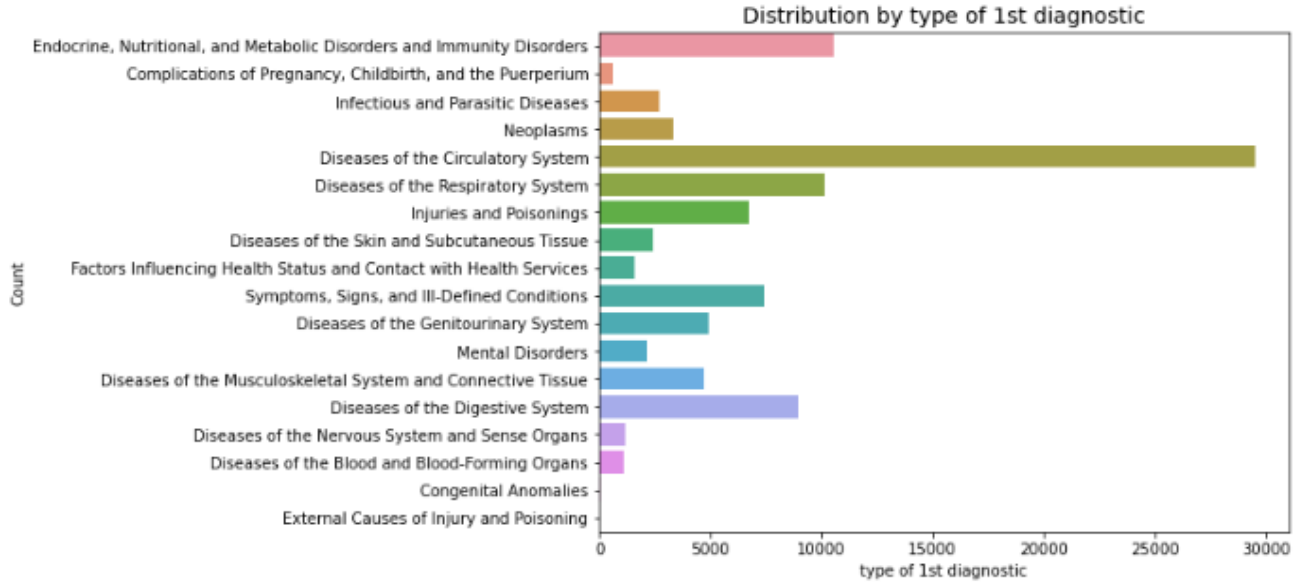




Mapping of Categorical value

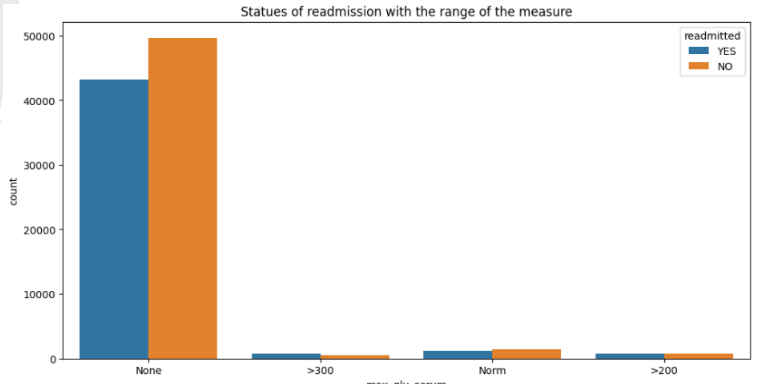
- Admission_type_id,discharge_disposition_id,admission_source_id
- Diag_1,Diag_2,Diag_3

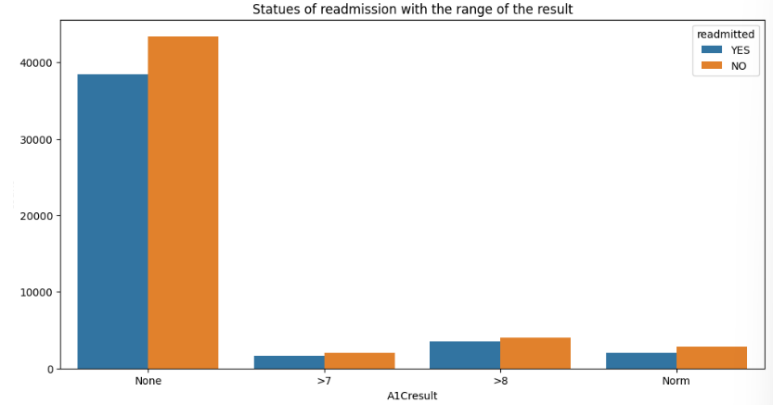


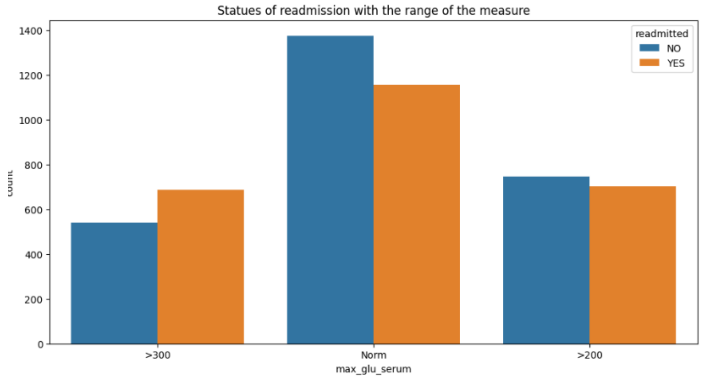


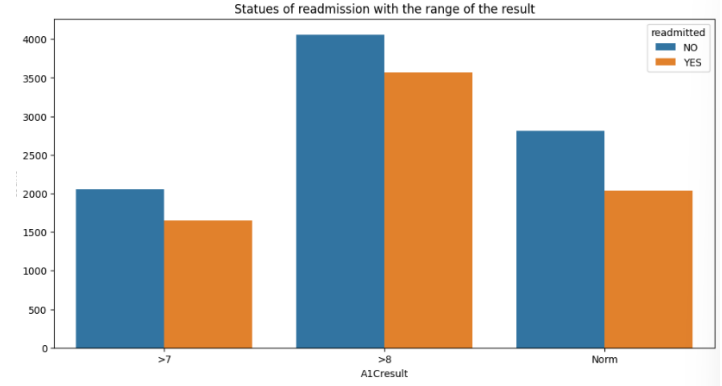
Analyze of the measure

- Have to see the data without the None value (where the parameter is not measured)
- Max_Glu_Serum
  - max_glu_serum greater than 300 there is high chance of Readmission
- A1CResult
  - No big correlation between the result of the measure and the readmission



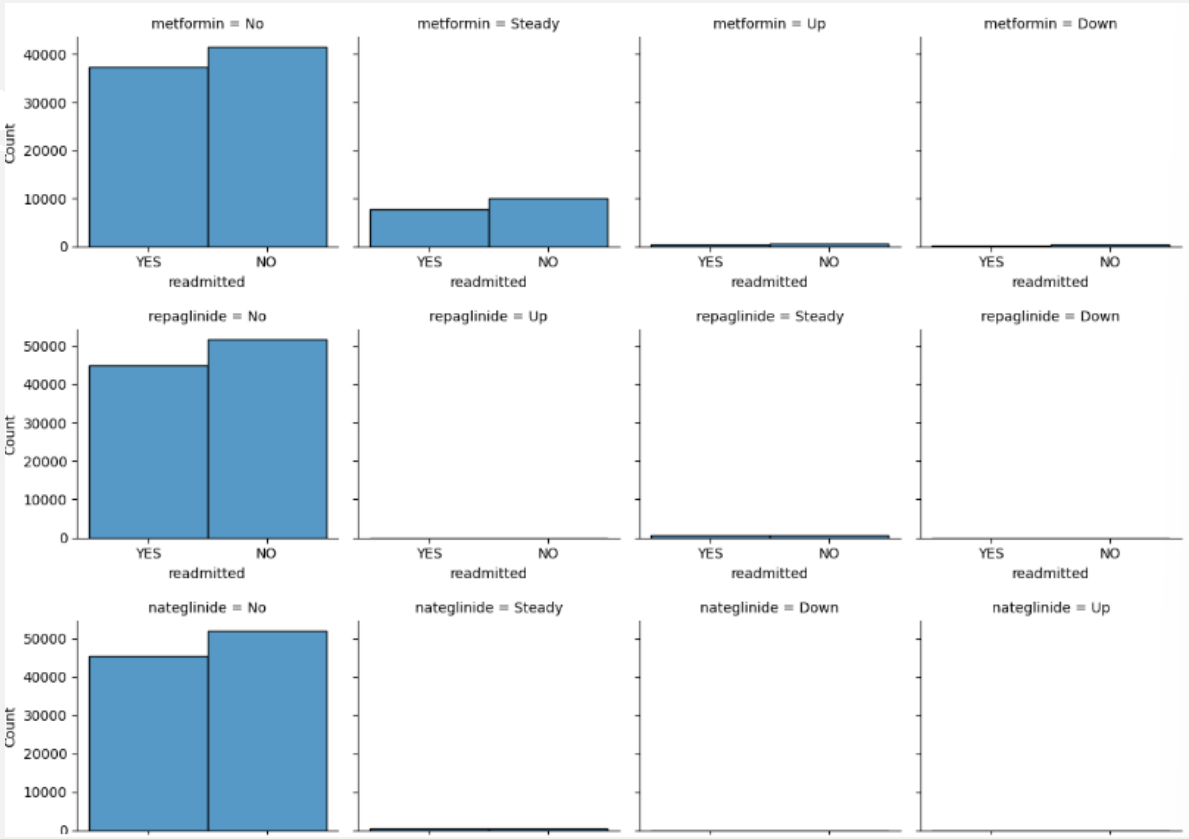






Analyze of the generic names of medication

- 24 columns for each medication
- 4 different values (No, steady, up and down)
- We useto see their distribution a facetgrid that show the proportion of readmission by value in each column

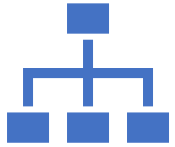


Machine learning objectives

We chose the variable ‘readmitted’ as the target feature

We chose all the remaining variables as the explanatory features

Our goal is to predict with the best accuracy if a patient is going to be readmitted to the hospital afterwards.



Since the readmitted column has 3 values : ‘no’, ’<30 ’ and ‘>30’, we chose to use only ‘yes’ and ‘no’ for the machine learning, to gain efficiency.

Encoding

- Manual encoding

```
#We replace the No with 0 and Yes with 1
data=data.replace({'No': 0, 'Yes': 1, 'Ch':1, 'Down': 1, 'Steady': 2, 'Up':3})
```

- Ordinal encoding

```
#We use ordinal encoder for these 3 columns because there is a hierarchy between its values
#encoding max_glu_serum
max_glu_serum_order = ['None', 'Norm', '>200', '>300']
data['max_glu_serum'] = pd.Categorical(data['max_glu_serum'], categories=max_glu_serum_order, ordered=True)
data = data.sort_values(by='max_glu_serum',ascending=True)
ordinal_encoder = OrdinalEncoder(categories=[max_glu_serum_order])
data['max_glu_serum'] = ordinal_encoder.fit_transform(data[['max_glu_serum']])
```

- One hot encoding

```
diagnosis_columns = ['admission_type_id','discharge_disposition_id','admission_source_id','race','gender','diag_1', 'diag_2', 'diag_3']

onehot_encoder = OneHotEncoder(sparse=False, drop='first')

# Loop through each diagnosis column
for column in diagnosis_columns:
    # Fit and transform the diagnosis column
    diagnosis_encoded = onehot_encoder.fit_transform(data[[column]])

    # Create a DataFrame with the encoded columns
    encoded_df = pd.DataFrame(diagnosis_encoded, columns=onehot_encoder.get_feature_names_out([column]))

    # Reset the index of the original data
    data.reset_index(drop=True, inplace=True)

    # Concatenate the original data with the encoded columns
    data = pd.concat([data, encoded_df], axis=1)

# Drop the original diagnosis columns
data = data.drop(diagnosis_columns, axis=1)
```

Handling missing values

- Lots of missing values on weight column so we chose to create 2 datasets with 2 different strategies :
  - data_weight : less value but weight column
  - data : more values but no weight column

```
data_weight = data.dropna()
data_weight.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3033 entries, 57 to 98009
Columns: 145 entries, encounter_id to diag_3_Symptoms, Signs, and Ill-Defined Conditions
dtypes: float64(109), int32(1), int64(35)
memory usage: 3.4 MB
```

```
data.drop(columns=["weight"],inplace=True)
data.dropna(inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98053 entries, 0 to 98052
Columns: 144 entries, encounter_id to diag_3_Symptoms, Signs, and Ill-Defined Conditions
dtypes: float64(108), int32(1), int64(35)
memory usage: 107.4 MB
```

Machine learning - Algorithms

- Used 5 different algorithms :
  - Support Vector Classification (SVC)
  - Logistic Regression (LR)
  - Random Forest (RF)
  - XGBoost (XGB)
  - Naive-Bayes (NB)

```
# train test split on data_weight
X = data_weight.drop('readmitted', axis=1)
y = data_weight['readmitted']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

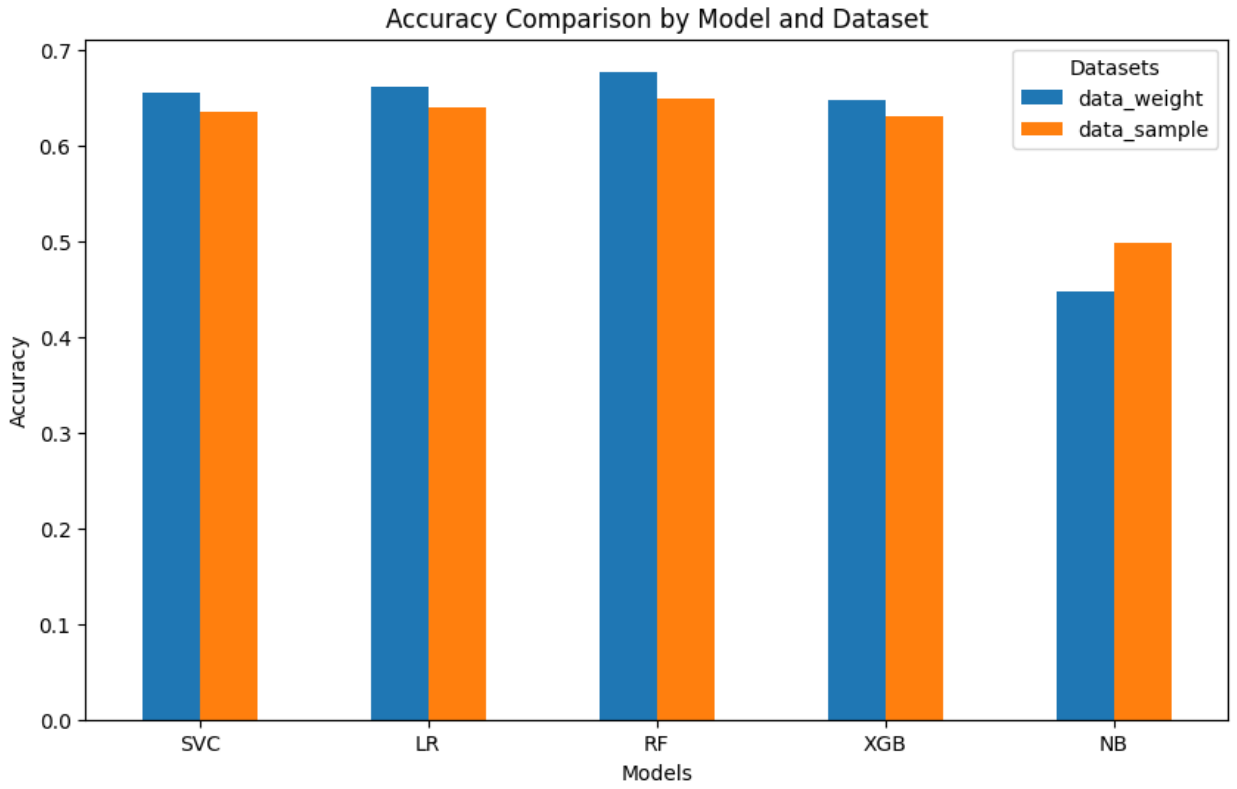
```
#SVC
svc_model = SVC(kernel='linear')

#Train the model on the training data
svc_model.fit(X_train, y_train)

#Make predictions on the testing data
svc_pred = svc_model.predict(X_test)

#Evaluate the model
SVC_w_accuracy = accuracy_score(y_test, svc_pred)
print(f"Accuracy: {SVC_w_accuracy:.2f}")
```

```
Accuracy: 0.66
```

Example of SVC algorithm implementation

Machine learning – 1st results

- Better prediction results on the data_weigth sample, so it's better to have less instances but to keep the weight information.



Machine Learning – Grid search

- We run grid search on the 5 previous algorithms, on the data_weight sample
- Lots of iteration time depending on the algorithm

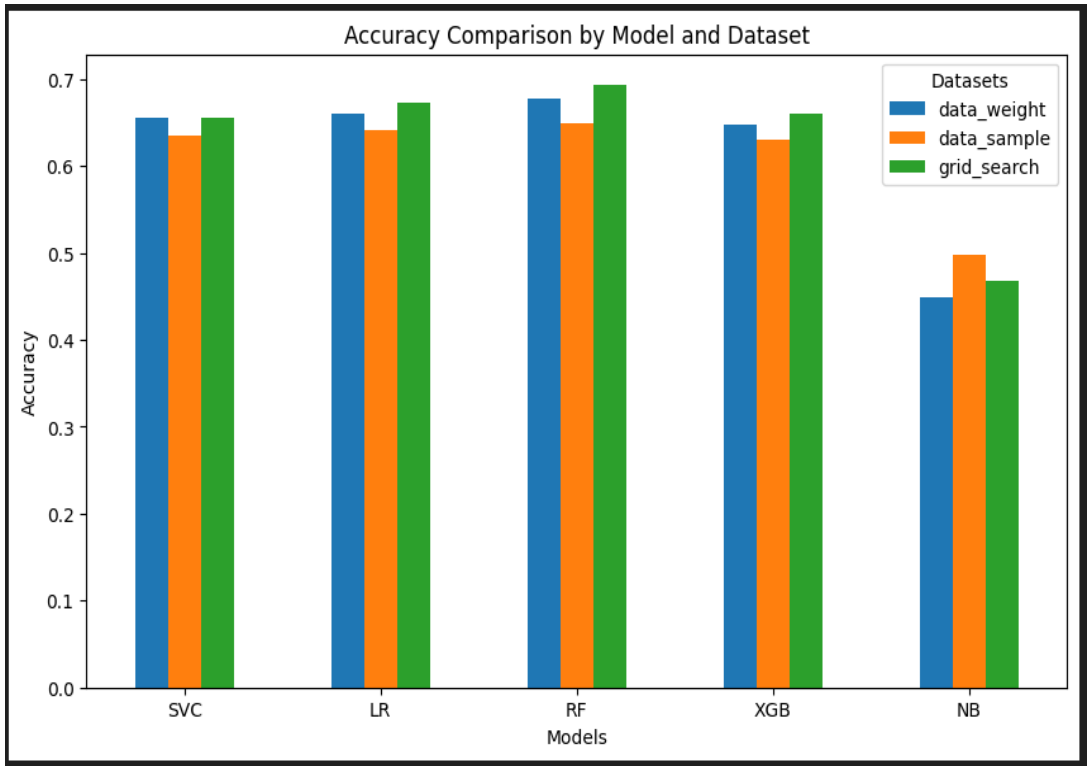
```
# Grid Search SVC
# Define the parameter grid
param_grid = {'C': [0.1, 1, 10], 'gamma': [0.01, 0.1, 1], 'kernel': ['linear', 'rbf']}

# Create the SVC model
svc_model = SVC()

# Instantiate the GridSearchCV object
grid_search = GridSearchCV(svc_model, param_grid, cv=5, scoring='accuracy')

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters found by the grid search
print("Best Hyperparameters:", grid_search.best_params_)

# Make predictions on the testing data using the best model
svc_pred = grid_search.predict(X_test)

# Evaluate the model
SVC_w_gs_accuracy = accuracy_score(y_test, svc_pred)
print(f"Accuracy: {SVC_w_gs_accuracy:.2f}")

# Print additional evaluation metrics
#print("\nClassification Report:")
#print(classification_report(Y_test, SVC_pred))
```

```
Best Hyperparameters: {'C': 0.1, 'gamma': 0.01, 'kernel': 'linear'}
Accuracy: 0.66
```

Example of SVC grid search implementation

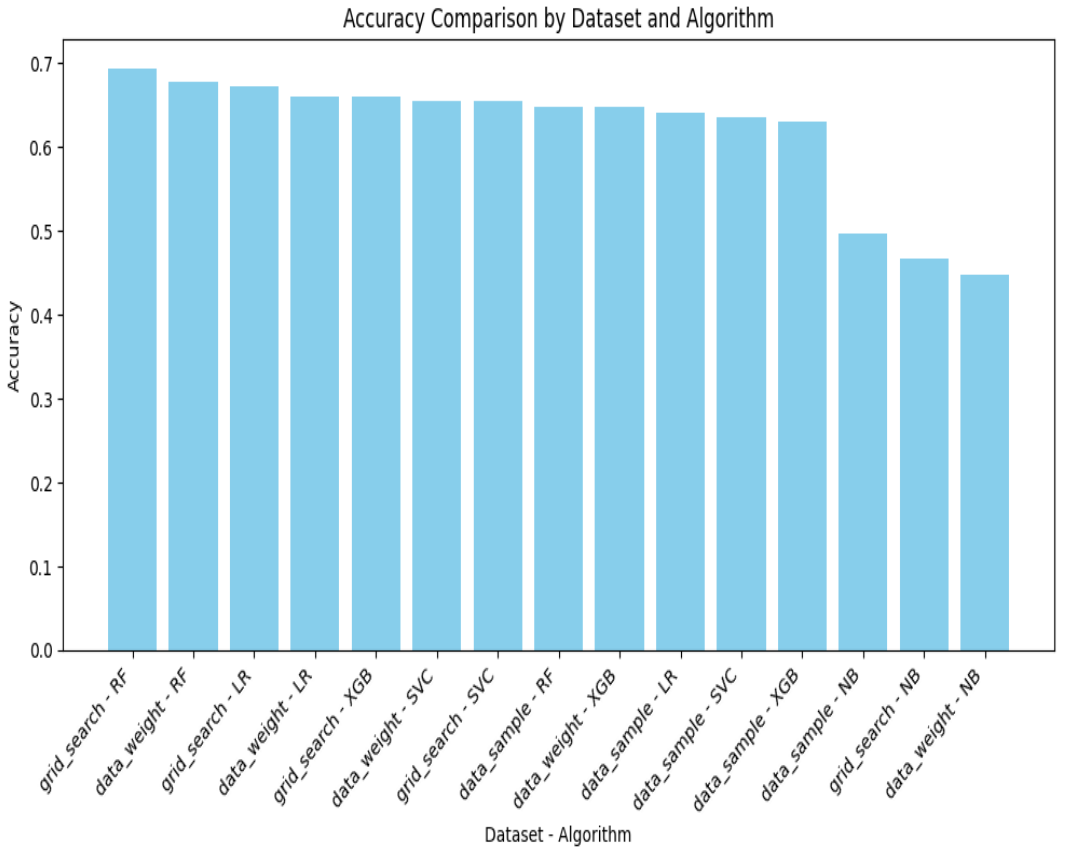
Machine learning – 2nd results

- Overall, grid search helped us improve our results



Machine Learning – final results

- Best algorithm in this situation is Random Forest, using grid search on the data_weight sample.
- Best accuracy = 0.693575



Conclusion

- Overall, we got pretty good results, and we also confirmed the importance of the weight in detecting diabetes
- To see further, we could try predicting which treatment is more efficient for each case