# Hospital Readmission Analysis

**Axel Houte, Thomas Hédan, Minji Kim - DIA3**

Python For Data Analysis

# Summary

- Dataset
- Variables
- Visualizations
- Link between readmission and variables
- Decision trees
- Bagging
- Gradient boosting
- Why re-sample ?
- Use again the same models
- Conclusion

# Dataset : Diabetes 130 US hospitals for years 1999-2008

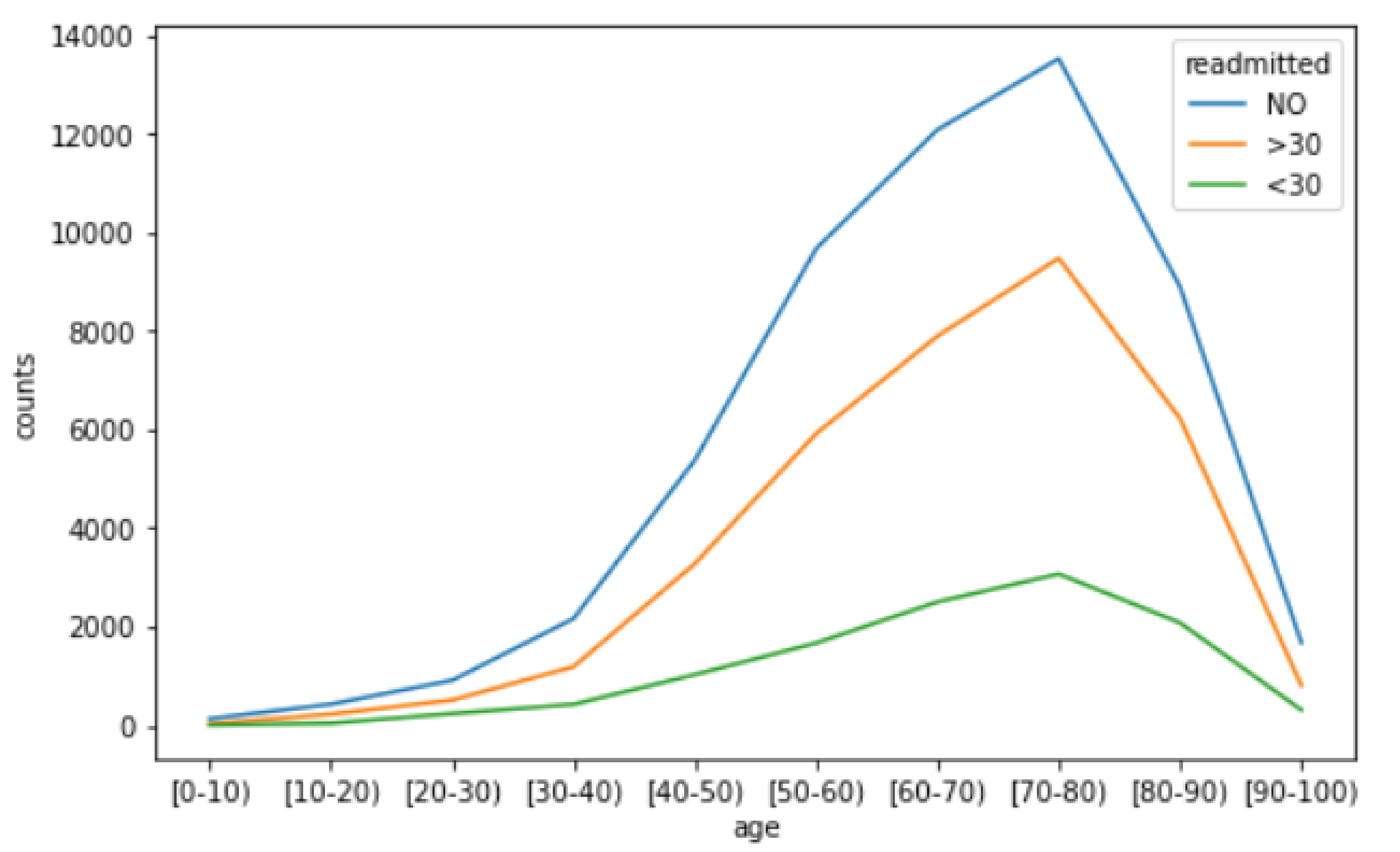
**Readmitted** : Days to inpatient readmission. Values: “<30” if the patient was readmitted in less than 30 days, “>30” if the patient was readmitted in more than 30 days, and “No” for no record of readmission

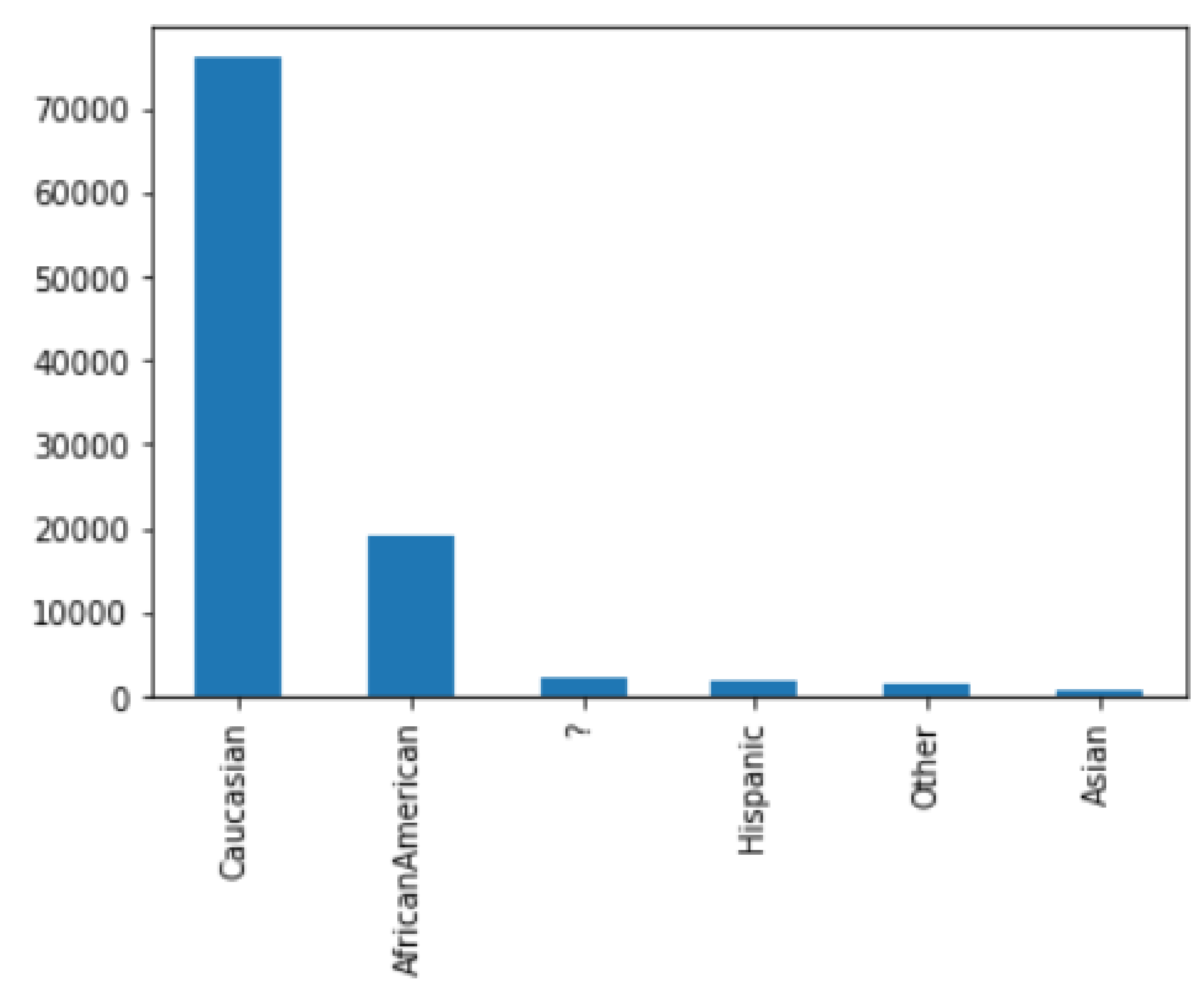
# VARIABLES

**Encounter ID**: Unique identifier of an encounter
**Patient number**: Unique identifier of a patient
**Race Values**: Caucasian, Asian, African American, Hispanic, and other
**Gender Values**: male, female, and unknown/invalid
**Age Grouped in 10-year intervals**: 0, 10), 10, 20), …, 90, 100)
**Weight**: Weight in pounds
**Admission type**: Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available
**Discharge disposition**: Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available
**Admission source**: Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital
**Time in hospital**: Integer number of days between admission and discharge
**Payer code**: Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay Medical
**Medical specialty**: Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon
**Number of lab procedures**: Number of lab tests performed during the encounter
**Number of procedures**: Numeric Number of procedures (other than lab tests) performed during the encounter
**Number of medications**: Number of distinct generic names administered during the encounter

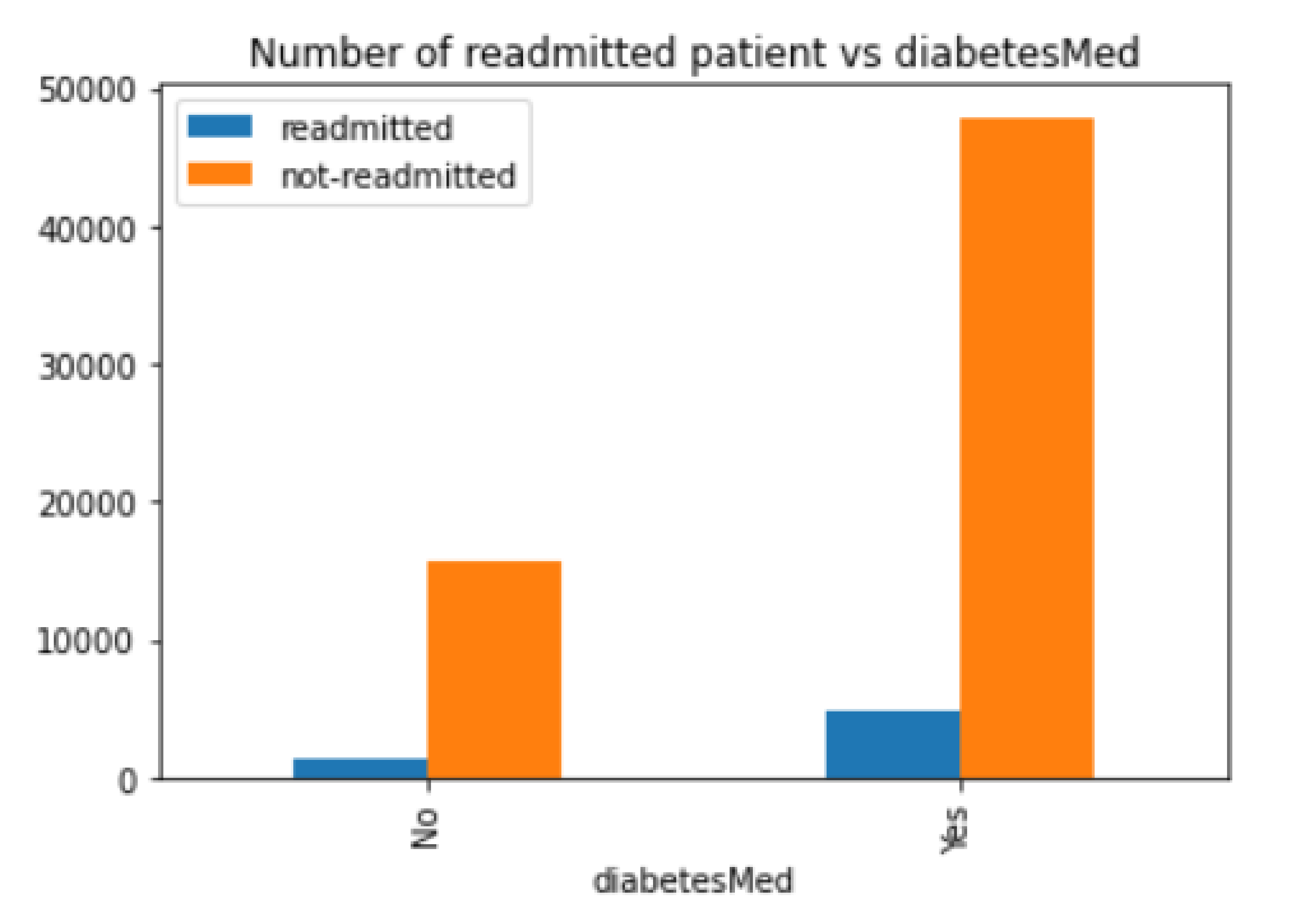
**Number of outpatient**: visits Number of outpatient visits of the patient in the year preceding the encounter
**Number of emergency**: visits Number of emergency visits of the patient in the year preceding the encounter
**Number of inpatient visits**: Number of inpatient visits of the patient in the year preceding the encounter
**Number of diagnoses**: Number of diagnoses entered to the system 0%
**Glucose serum** test result Indicates the range of the result or if the test was not taken. Values: “>200,” “>300,” “normal,” and “none” if not measured
**A1c** test result Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>7” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured.
**Change of medications**: Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”
**Diabetes medications**: Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”
24 features for medications Values: “up” if the dosage was increased during the encounter, “down” if the dosage was decreased, “steady” if the dosage did not change, and “no” if the drug was not prescribed

# Visalulizations

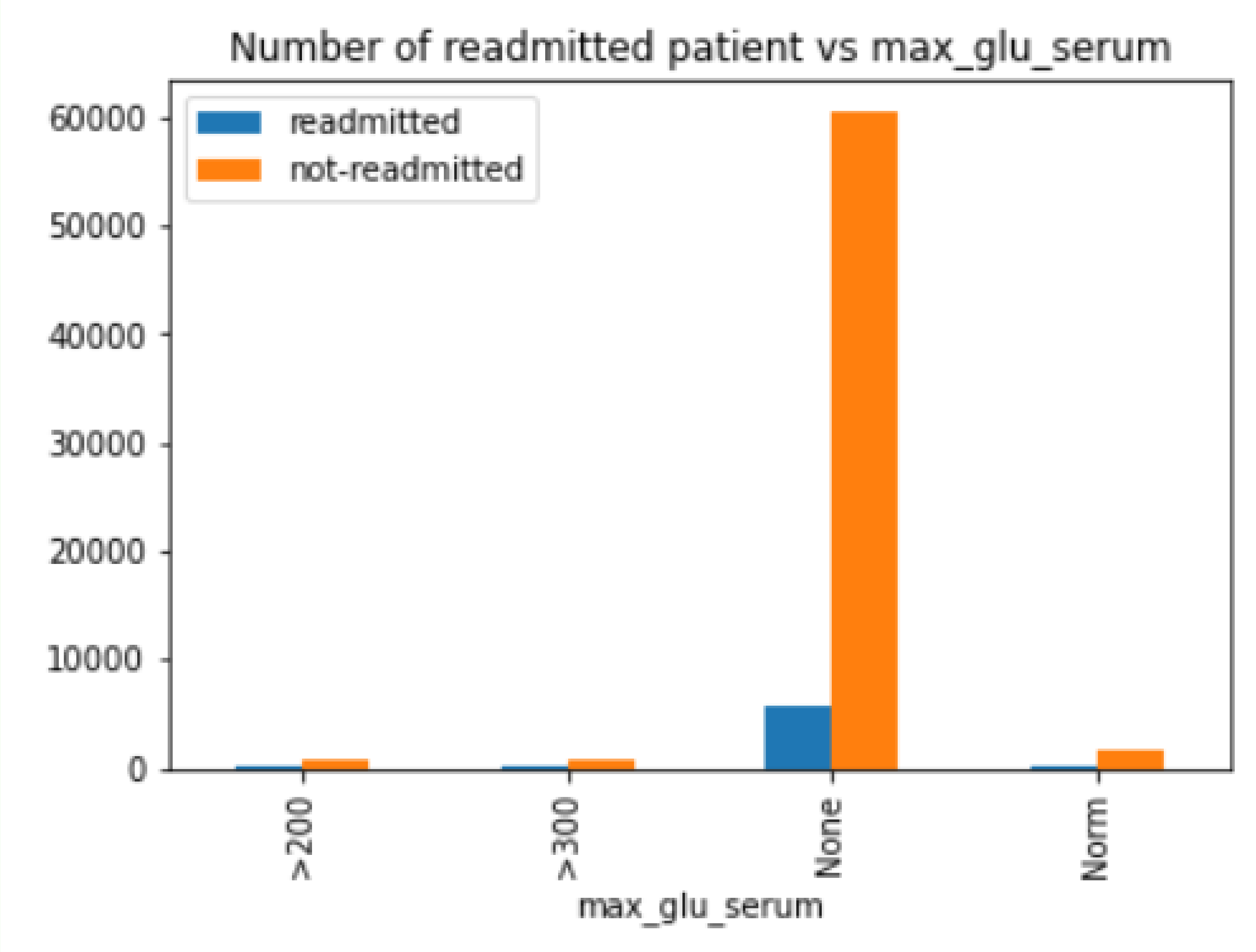


# Link between readmission and variables

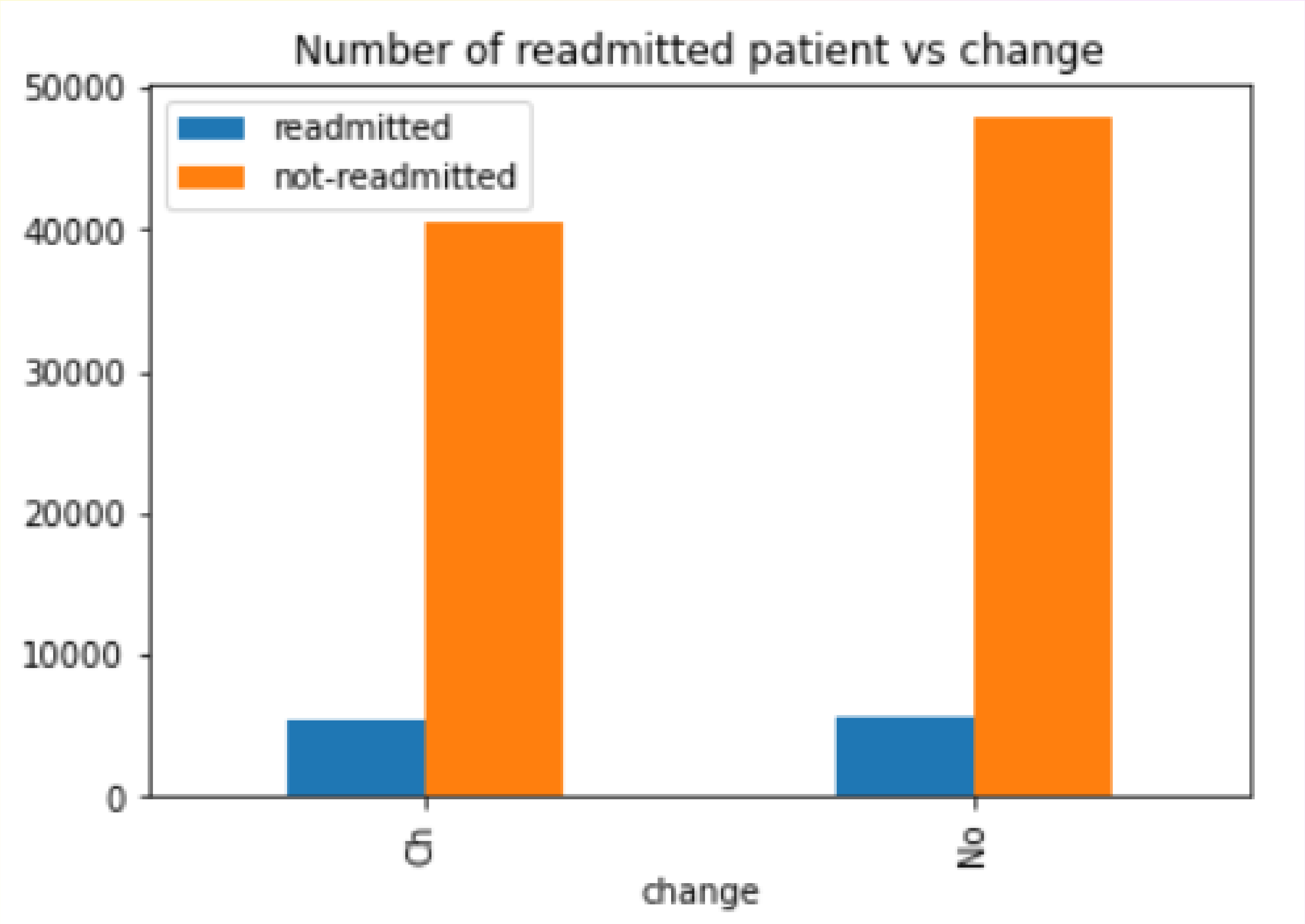



Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”

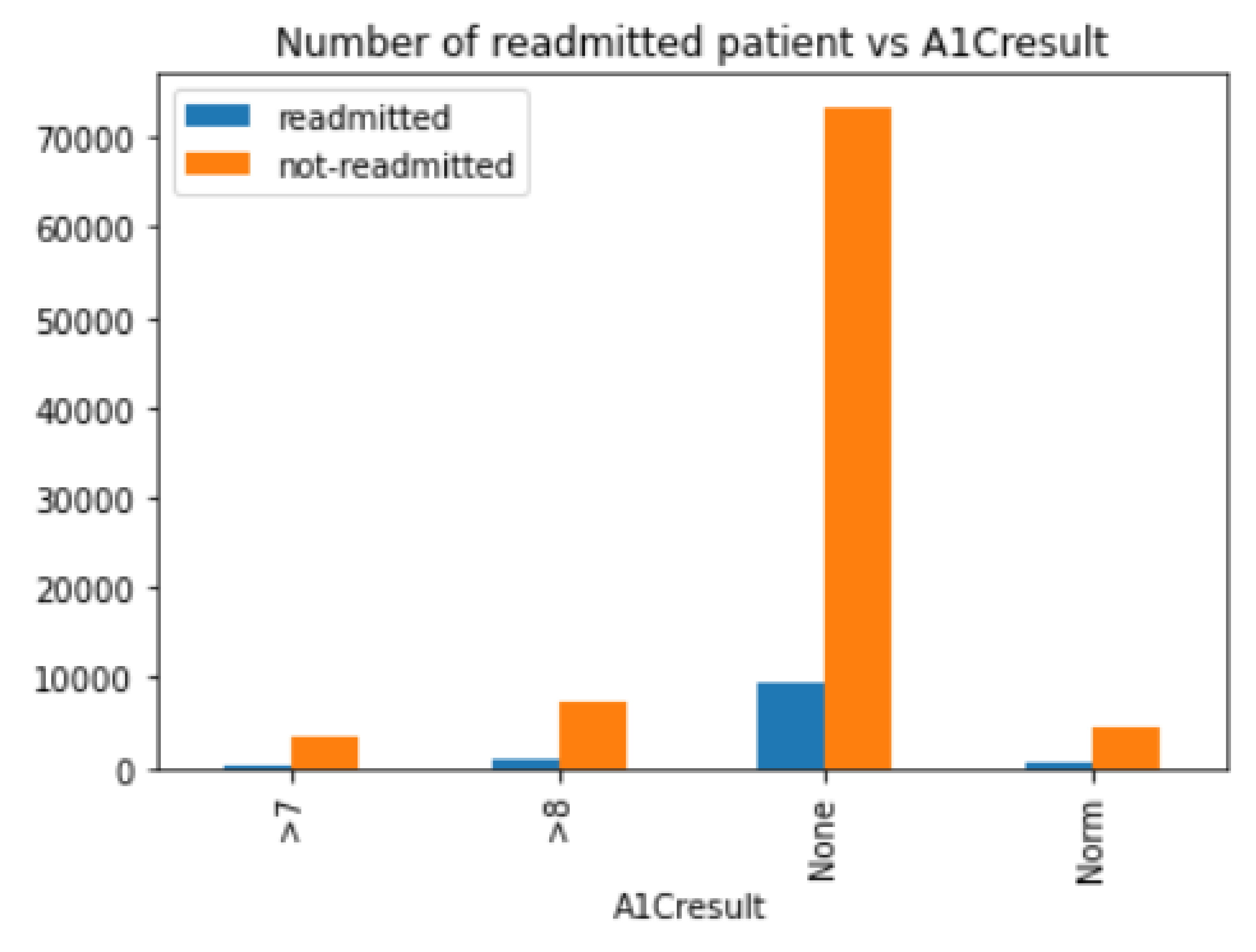



Indicates the range of the result or if the test was not taken. Values: “>200,” “>300,” “normal,” and “none” if not measured

# Link between readmission and variables




Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”




Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>7” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured.

# Data pre-processing

## 1- Identify Null values

```
List_NA = ['?','Unknown/Invalid','Not Mapped','NULL']

for col in df.columns:
    df[col] = df[col].apply(lambda x : np.NaN if(x in List_NA) else x)
```

```
check_null(df)
```

```
2.2335554114340743 % of col  race  is null.
0.002947939390366134 % of col  gender  is null.
96.85847925633315 % of col  weight  is null.
39.5574160328597 % of col  payer_code  is null.
49.08220820313268 % of col  medical_specialty  is null.
0.020635575732256294 % of col  diag_1  is null.
0.3517874339170253 % of col  diag_2  is null.
1.398305917497003 % of col  diag_3  is null.
['race', 'gender', 'weight', 'payer_code', 'medical_specialty', 'diag_1', 'diag_2', 'diag_3']
```

## 2- Handle Null values and useless features

```
df.drop(['weight','payer_code','encounter_id','examide','citoglipton','medical_specialty','diag_2','diag_3'],axis=1,inplace=True
check_null(df)
```

```
2.2335554114340743 % of col  race  is null.
0.002947939390366134 % of col  gender  is null.
0.020635575732256294 % of col  diag_1  is null.
['race', 'gender', 'diag_1']
```

```
useless_drugs = ['repaglinide','nateglinide','chlorpropamide','acetohexamide','glipizide','tolbutamide',
                 'acarbose','miglitol', 'troglitazone','tolazamide','glyburide-metformin',
                 'glipizide-metformin','glimepiride-pioglitazone','metformin-rosiglitazone','metformin-pioglitazone']

df.drop(useless_drugs,inplace=True,axis=1)
df.shape
```

```
(69658, 25)
```

# 3- Outputs transformation and drop duplicates

```
df['readmitted'] = df['readmitted'].apply(lambda x : 'YES' if(x == '<30') else 'NO')
df['readmitted'].value_counts()
```

```
NO      88309
YES     11164
Name: readmitted, dtype: int64
```

```
print(df.shape)
df = df.drop_duplicates(subset= ['patient_nbr'], keep = 'first')
print(df.shape)
```

```
(99473, 40)
(69658, 40)
```

```
def level1_diag1(x):
    if(type(x) == int):
        if (x >= 390 and x < 460) or (np.floor(x) == 785):
            return 1
        elif (x >= 460 and x < 520) or (np.floor(x) == 786):
            return 2
        elif (x >= 520 and x < 580) or (np.floor(x) == 787):
            return 3
        elif (np.floor(x) == 250):
            return 4
        elif (x >= 800 and x < 1000):
            return 5
        elif (x >= 710 and x < 740):
            return 6
        elif (x >= 580 and x < 630) or (np.floor(x) == 788):
            return 7
        elif (x >= 140 and x < 240):
            return 8
        else:
            return 0
    else:
        return 0
```

```
df['diag_1'] = df['diag_1'].apply(lambda x : level1_diag1(x))
df.head().T
```

# 4- Partition some of the features

```
df['admission_type_id'] = df['admission_type_id'].replace(2,1)   # Urgent to Emergency
df['admission_type_id'] = df['admission_type_id'].replace(7,1)   # Trauma center to Emergency
df['admission_type_id'] = df['admission_type_id'].replace(4,1)   # Newborn to Emergency
df['admission_type_id'] = df['admission_type_id'].replace(6,5)   # Null to Not available
df['admission_type_id'] = df['admission_type_id'].replace(8,5)   # Not Mapped to Not available

df.admission_type_id.value_counts()
```

```
1    48111
3    13606
5     7941
Name: admission_type_id, dtype: int64
```

# 5- Scale the data

```python
from sklearn.preprocessing import StandardScaler
```

```python
cols_to_standardize = ['time_in_hospital','num_lab_procedures','num_procedures','num_medications',
                       'number_diagnoses','service_utilization']
for col in cols_to_standardize :
    df2[col] =( df2[col] - df2[col].mean() ) / df2[col].std()
```

# 6- One hot encoding for some features

```python
def modify_and_add_col_in(col,df1=df,df2=df2):
    new_cols = []
    for val in df1[col].unique():
        df2[col+'_'+ str(val)] = df1[col].apply(lambda x : 1 if(x==val) else 0)
```

```python
for col in cols_to_change :
    modify_and_add_col_in(col)
```

| insulin |
|---|
| Steady |
| Up |
| Steady |
| Up |
| No |

| insulin_No | insulin_Up | insulin_Steady | insulin_Down |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |

# 7- Over sampling

```
from imblearn.over_sampling import SMOTE
```

```
sm = SMOTE(k_neighbors = 3 ,random_state=42)
print('X shape : ', X.shape,'  Y shape', y.shape)
X_res, y_res = sm.fit_resample(X, y)
print('Resampling...')
print('new X shape : ', X_res.shape,'new Y shape', y_res.shape)
```

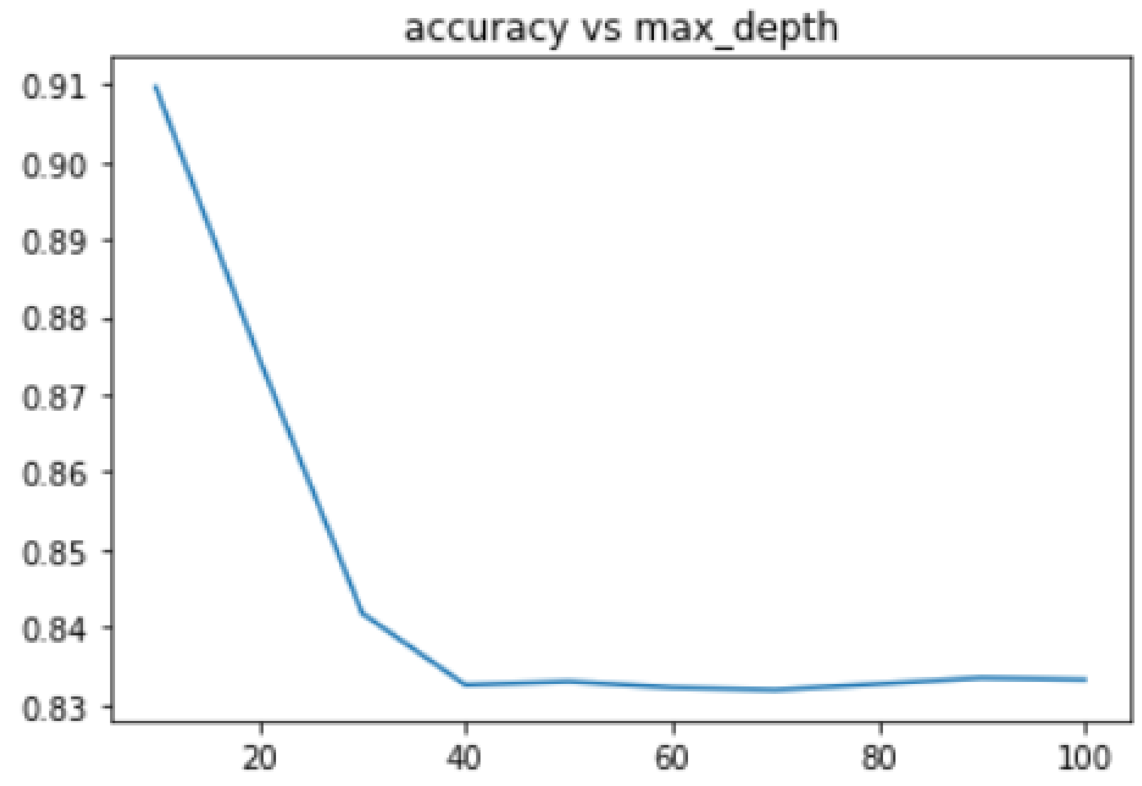
```
X shape :  (69658, 72)   Y shape (69658,)
Resampling...
new X shape :  (126986, 72) new Y shape (126986,)
```

# Decision tree

```
[[63427  5974]
 [   66   191]]
```

```
Best training accuracy =  0.9096830960206217
Best parameters :  {'criterion': 'entropy', 'max_depth': 10}
Validation accuracy =  0.9041774332472007
test repartition :
 0    12635
1     1297
Name: readmitted_YES, dtype: int64
Confusion matrix :
 [[12588  1288]
 [   47     9]]
```

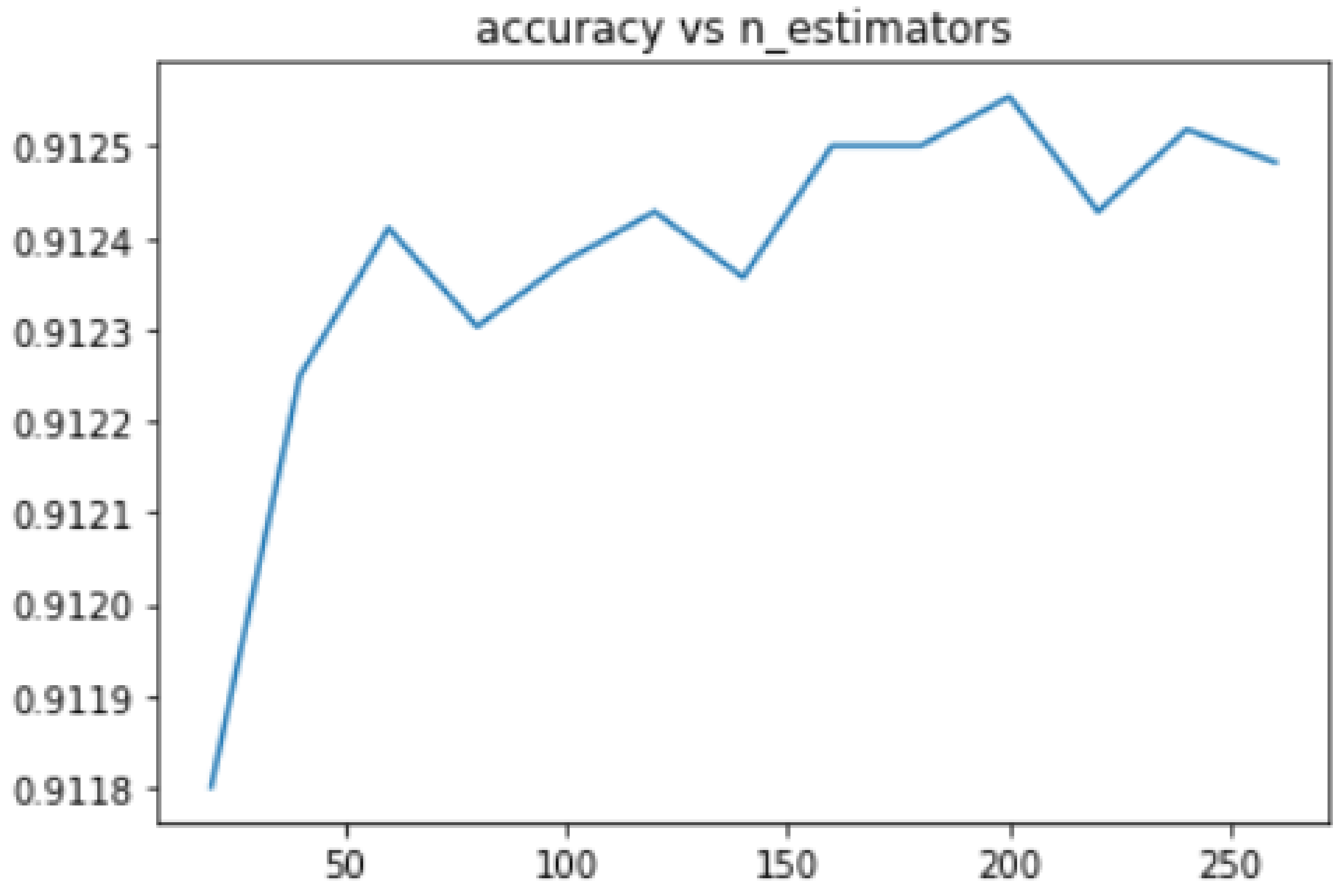
```
plt.plot(tree_res2['param_max_depth'],tree_res2['mean_test_score'])
plt.title('accuracy vs max_depth')
plt.show()
```



# Random Forest

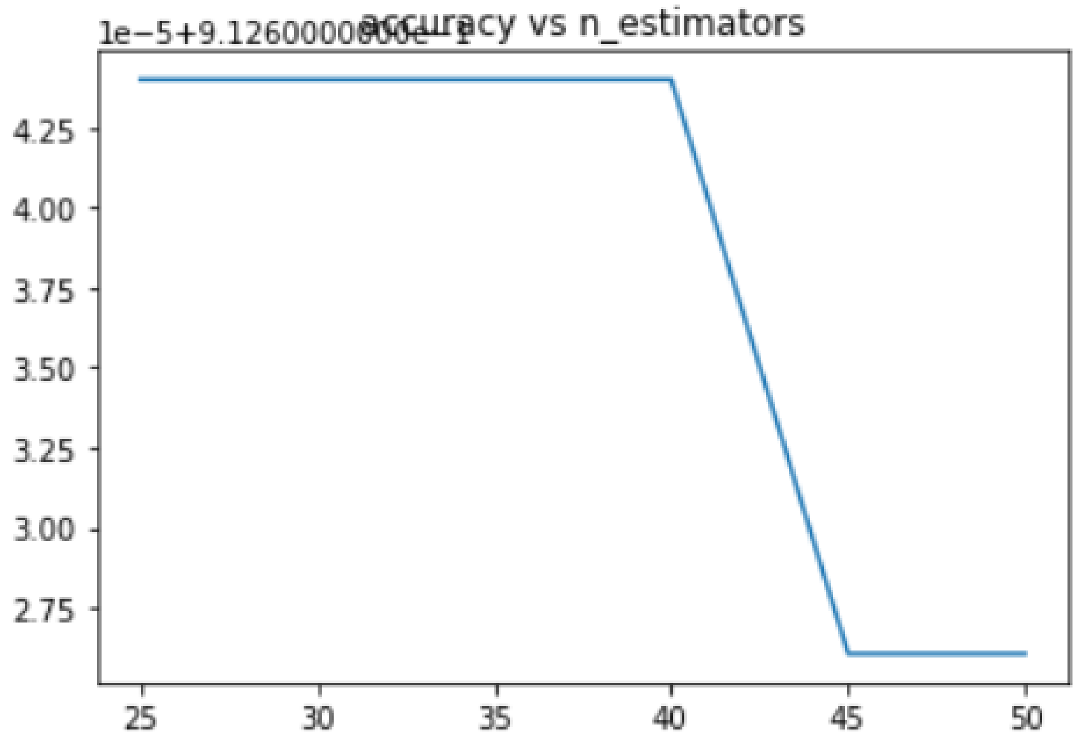
```
[[63487  1296]
 [    6  4869]]
```

```
Random Forest :
Best training accuracy =  0.9125542872097631
Best parameters :  {'max_features': 'sqrt', 'n_estimators': 200}
Validation accuracy =  0.9066178581682458
test repartition :
 0    12635
1     1297
Name: readmitted_YES, dtype: int64
Confusion matrix :
 [[12629  1295]
 [    6     2]]
```



# Gradient Boosting

```
Gradiant boosting :
Best training accuracy =  0.9126440087143864
Best parameters :  {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 25, 'random_state': 1, 'subsample': 1}
Validation accuracy =  0.9069049669824863
test repartition :
 0    12635
1     1297
Name: readmitted_YES, dtype: int64
Confusion matrix :
 [[12635  1297]
 [    0     0]]
```




```
[[63493  6164]
 [    0      1]]
```

# Why resample ?

→

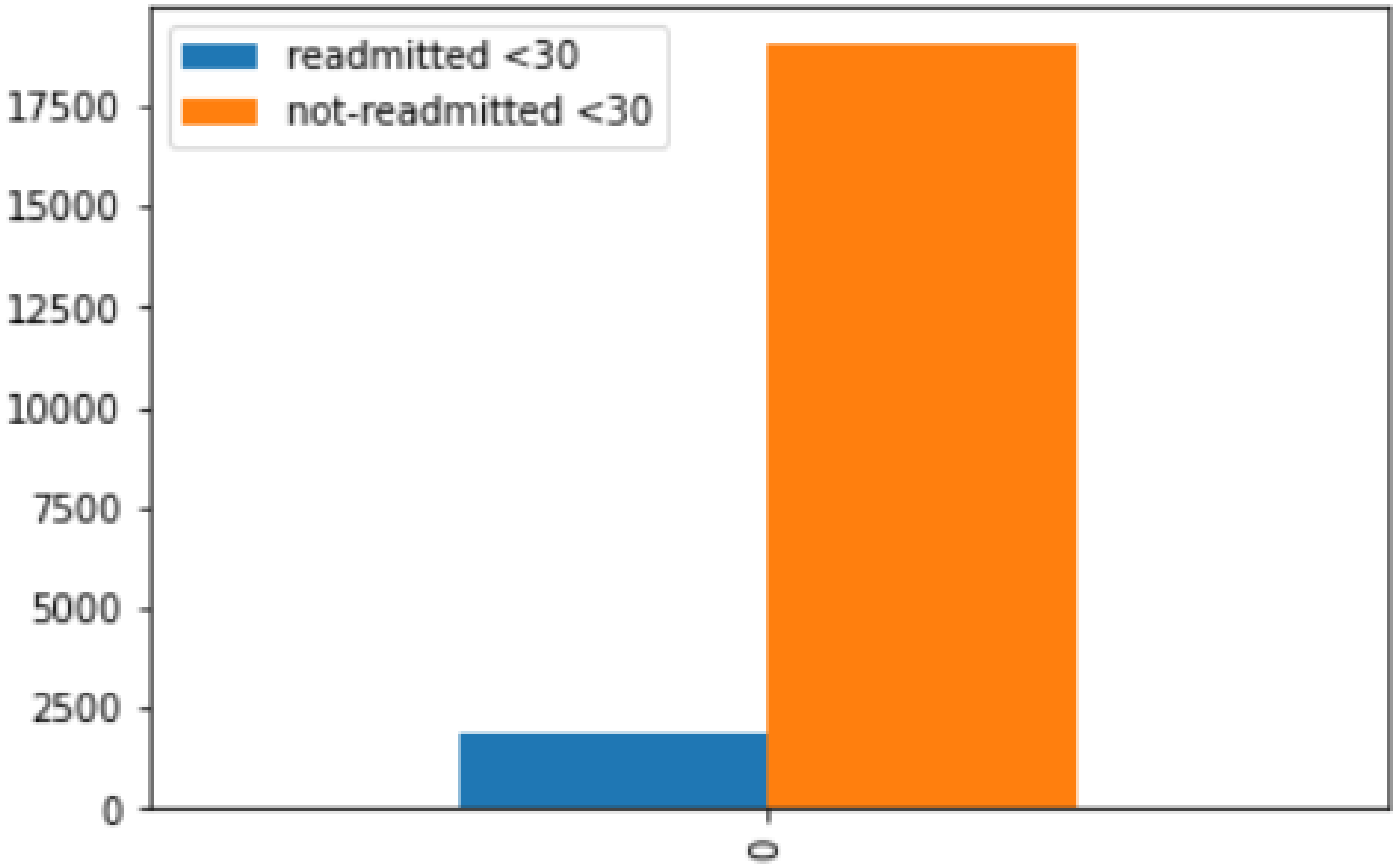
## 91.10%

- Non-readmitted patients of the training sample

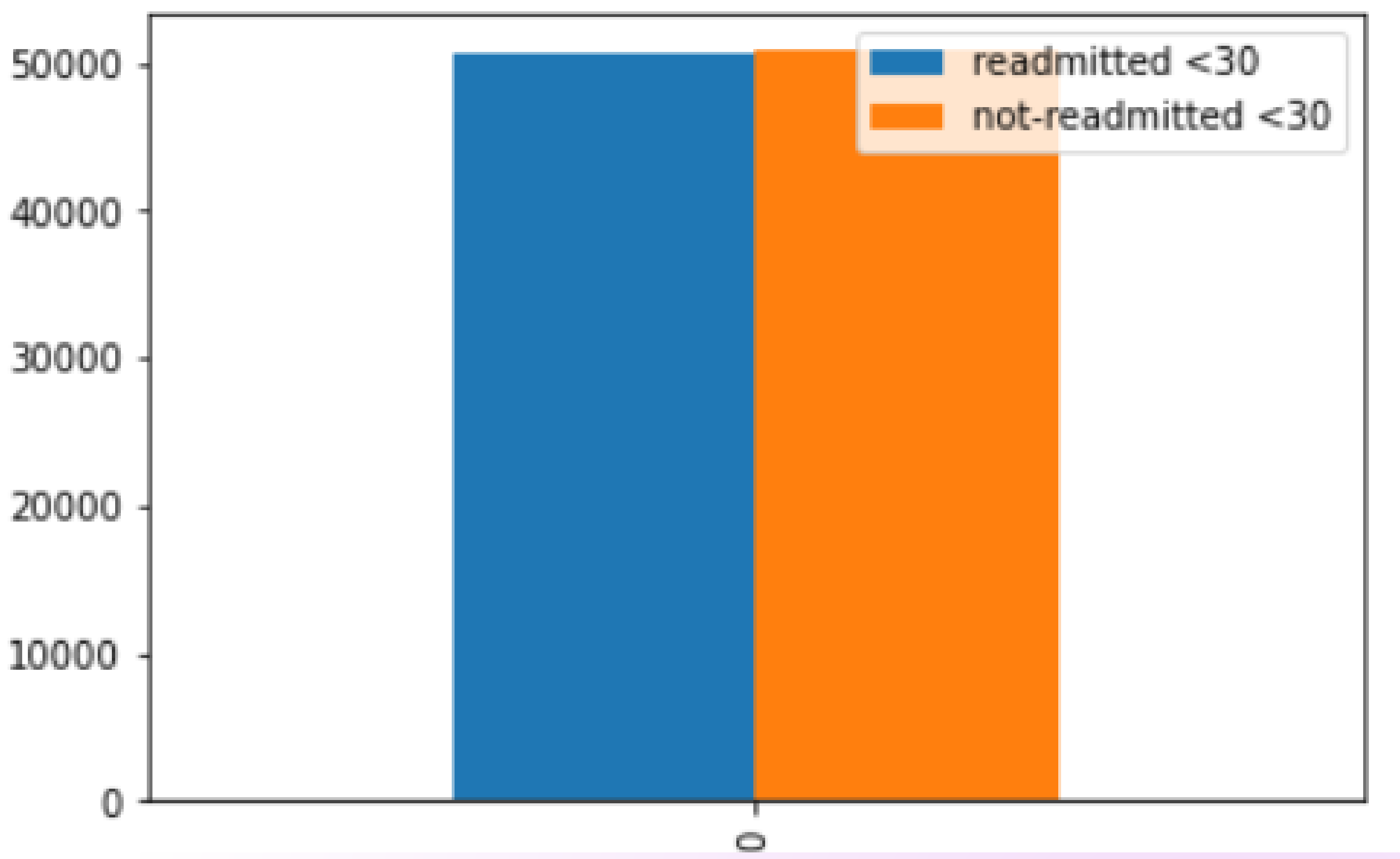
## Only 8.89%

- Readmitted patients of the training sample

# Before re-sampling




# After re-sampling



# Decision tree with oversampling

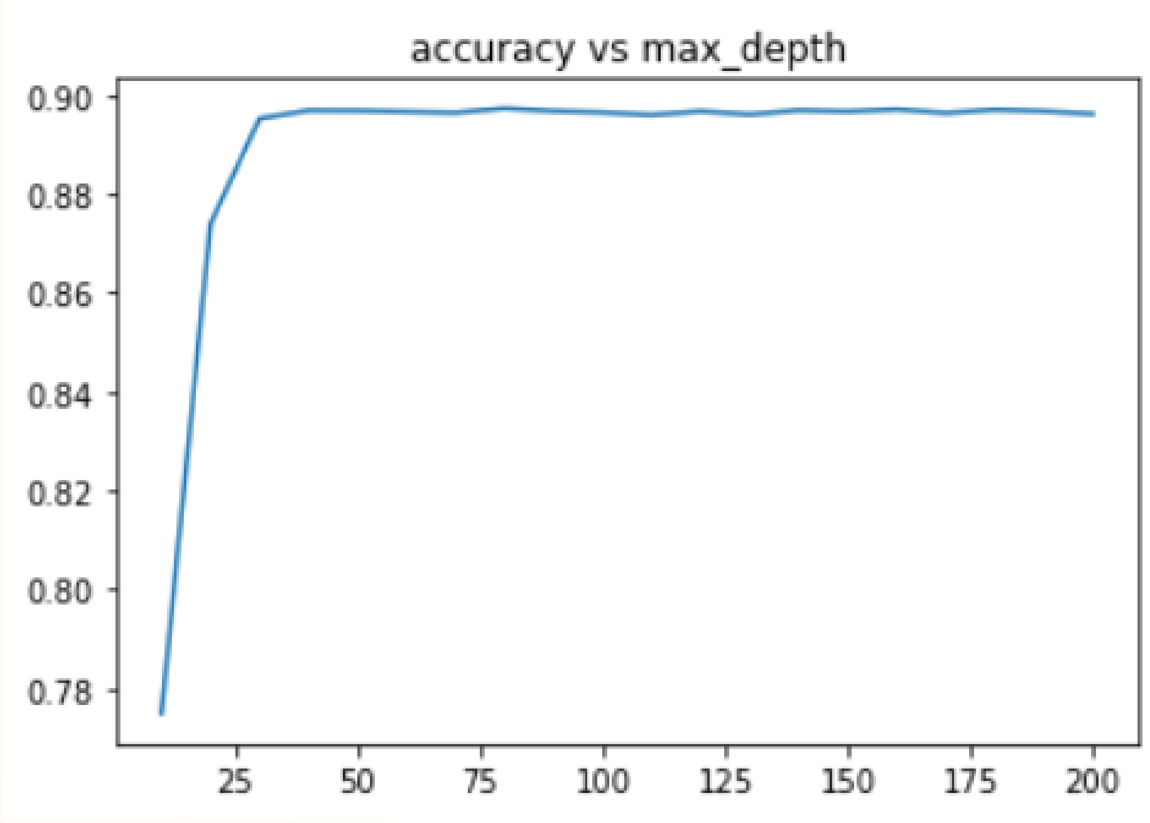
```
scoring_type='accuracy'

##----------------------------------------Decision tree-------------------------------------


param_tree = {'max_depth':[i*10 for i in range(1,21)],'criterion' : ["entropy"]}

model_tree,tree_res = test_tree_c(param_tree, scoring_type,X_train, X_test, y_train, y_test)
make_sound()
```

```
Best training accuracy =  0.897192574764498
Best parameters :  {'criterion': 'entropy', 'max_depth': 80}
Validation accuracy =  0.8995590203953067
test repartition :
 1    12741
0    12657
Name: readmitted_YES, dtype: int64
Confusion matrix :
 [[11245  1139]
 [ 1412 11602]]
```

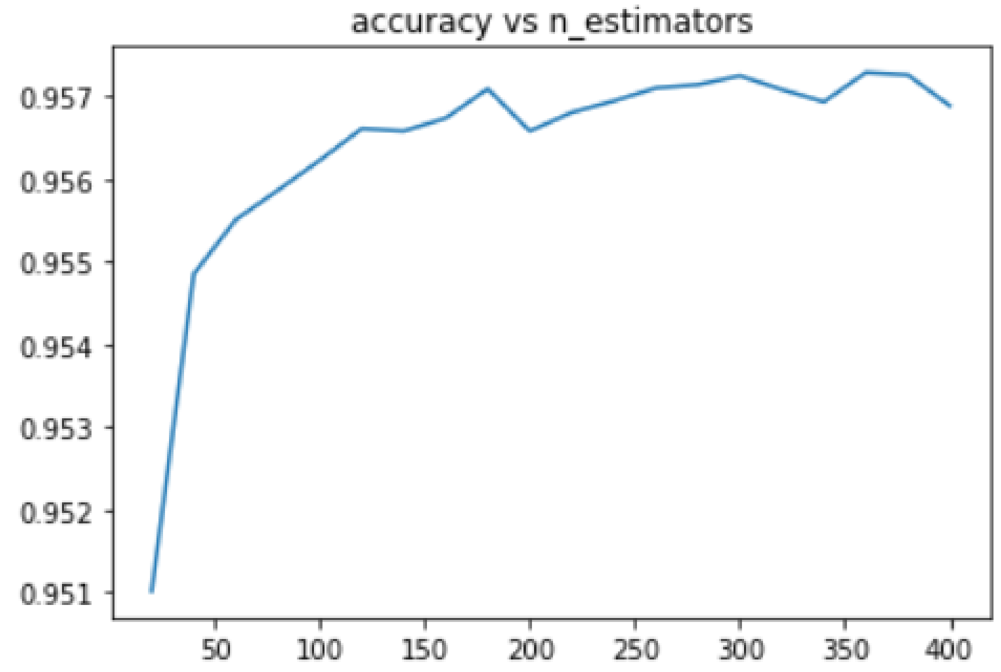


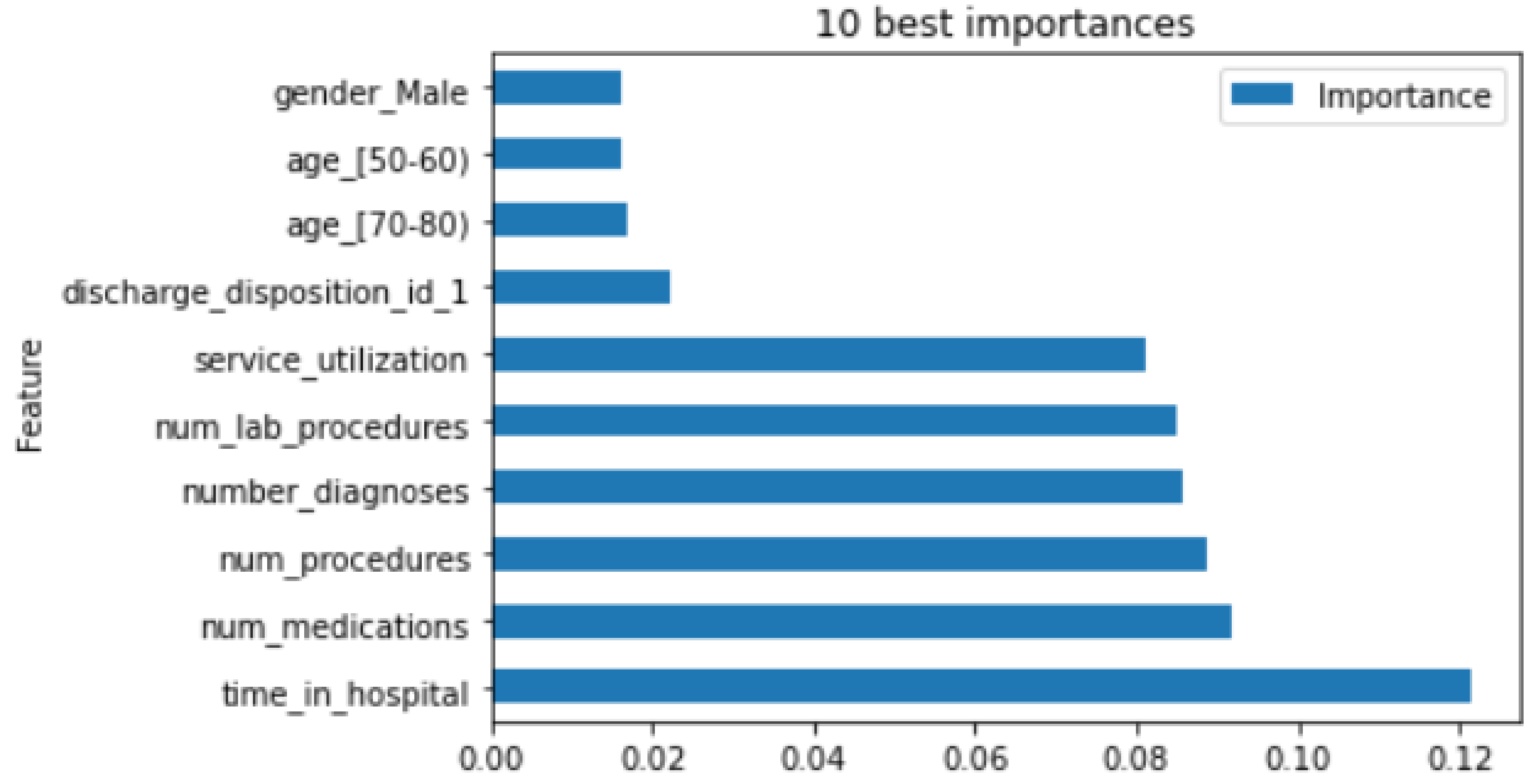

```
conf_dt =confusion_matrix(model_tree.predict(X),y)
conf_dt
```

```
array([[62081,   985],
       [ 1412,  5180]], dtype=int64)
```

# Random Forest with oversampling

```
Random Forest :
Best training accuracy =  0.9572882679885077
Best parameters :  {'max_features': 'sqrt', 'n_estimators': 360}
Validation accuracy =  0.9567288762894716
test repartition :
 1    12741
0    12657
Name: readmitted_YES, dtype: int64
Confusion matrix :
 [[12544   986]
 [  113 11755]]
```

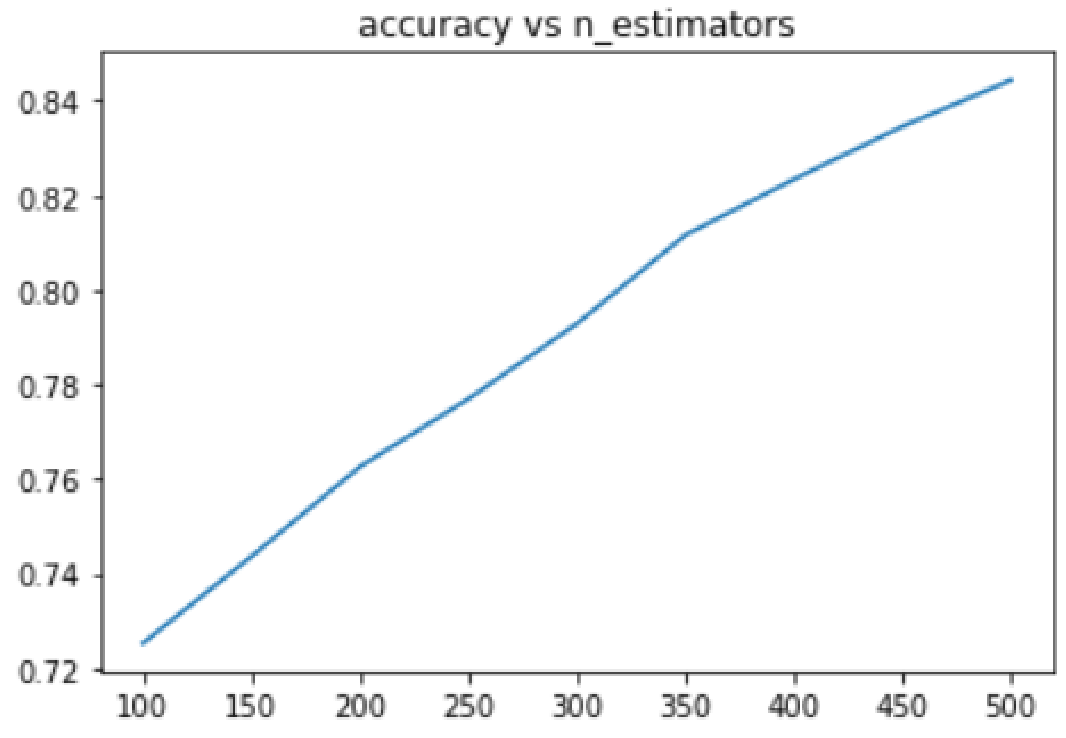



```
array([[63380,   958],
       [  113,  5207]], dtype=int64)
```

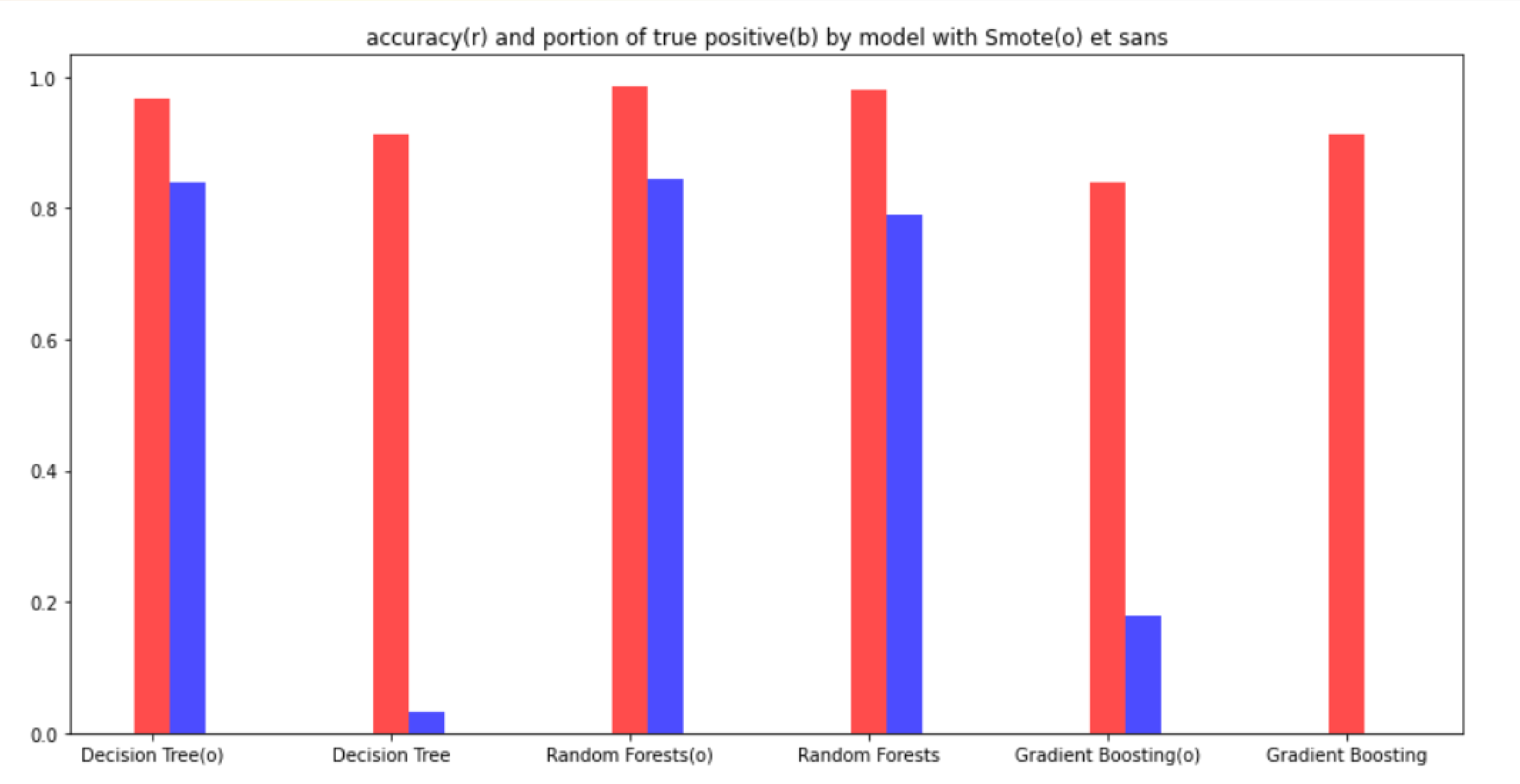
# Gradient Boosting with oversampling

```
Gradiant boosting :
Best training accuracy =  0.8443419051851366
Best parameters :  {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 500, 'random_state': 1, 'subsample': 1}
Validation accuracy =  0.8446334357035987
test repartition :
 1    12741
0    12657
Name: readmitted_YES, dtype: int64
Confusion matrix :
 [[11454  2743]
 [ 1203  9998]]
```




```
array([[57310,  5068],
       [ 6183,  1097]], dtype=int64)
```

# Conclusion


accuracy(r) and portion of true positive(b) by model with Smote(o) et sans
1.0
0.8
0.6
0.4
0.2
0.0
Decision Tree(o)
Decision Tree
Random Forests(o)
Random Forests
Gradient Boosting(o)
Gradient Boosting

# Thank you !

Do you have any questions ?