PYTHON FOR DATA ANALYSIS REPORT

VAREZ ARTHUR

DIA5

HTTPS://GITHUB.COM/ARTHURVAREZ/FINALPROJECT_PYTHON/

INTRODUCTION

For this project I had to work on a dataset quantifying the drug consumptions for 1885 people.

For each respondent 12 attributes are given : 7 about personnalitie mesurements and feelings. Partcipent were asked about their habits on 18 drugs. Some are legals, some are not. There's also a fake drug in this panel.
For those drugs they had to define their consumption : "Never Used", "Used over a Decade Ago", "Used in Last Decade", "Used in Last Year", "Used in Last Month", "Used in Last Week", and "Used in Last Day".

All the datas are categoricals but some could be interpreted as real numbers.

THE DATASET

- By using read_csv method from Pandas package, I've been able to load the dataset and to have a quick look on it.

- Here we also have its shape and a view on the columns.

	ID	Age	Gender	Education	Country	Ethnicity	Nscore	Escore	Oscore	Ascore	...	Ecstasy	Heroin	Ketamine
0	1	0.49788	0.48246	-0.05921	0.96082	0.12600	0.31287	-0.57545	-0.58331	-0.91699	...	CL0	CL0	CL0
1	2	-0.07854	-0.48246	1.98437	0.96082	-0.31685	-0.67825	1.93886	1.43533	0.76096	...	CL4	CL0	CL2
2	3	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.46725	0.80523	-0.84732	-1.62090	...	CL0	CL0	CL0
3	4	-0.95197	0.48246	1.16365	0.96082	-0.31685	-0.14882	-0.80615	-0.01928	0.59042	...	CL0	CL0	CL2
4	5	0.49788	0.48246	1.98437	0.96082	-0.31685	0.73545	-1.63340	-0.45174	-0.30172	...	CL1	CL0	CL0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1880	1884	-0.95197	0.48246	-0.61113	-0.57009	-0.31685	-1.19430	1.74091	1.88511	0.76096	...	CL0	CL0	CL0
1881	1885	-0.95197	-0.48246	-0.61113	-0.57009	-0.31685	-0.24649	1.74091	0.58331	0.76096	...	CL2	CL0	CL0
1882	1886	-0.07854	0.48246	0.45468	-0.57009	-0.31685	1.13281	-1.37639	-1.27553	-1.77200	...	CL4	CL0	CL2
1883	1887	-0.95197	0.48246	-0.61113	-0.57009	-0.31685	0.91093	-1.92173	0.29338	-1.62090	...	CL3	CL0	CL0
1884	1888	-0.95197	-0.48246	-0.61113	0.21128	-0.31685	-0.46725	2.12700	1.65653	1.11406	...	CL3	CL0	CL0

1885 rows × 32 columns

```
#columns of the dataset
dataset.columns
```

```
Index(['ID', 'Age', 'Gender', 'Education', 'Country', 'Ethnicity', 'Nscore',
       'Escore', 'Oscore', 'Ascore', 'Cscore', 'Impulsive', 'SS', 'Alcohol',
       'Amphet', 'Amyl', 'Benzos', 'Caff', 'Cannabis', 'Choc', 'Coke', 'Crack',
       'Ecstasy', 'Heroin', 'Ketamine', 'Legalh', 'LSD', 'Meth', 'Mushrooms',
       'Nicotine', 'Semer', 'VSA'],
      dtype='object')
```

THE DATASET

- To have a better overview on the statistics we can use the method describe :

```
dataset.describe()
```

	ID	Age	Gender	Education	Country	Ethnicity	Nscore	Escore	Oscore
count	1885.000000	1885.00000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000
mean	945.294960	0.03461	-0.000256	-0.003806	0.355542	-0.309577	0.000047	-0.000163	-0.000534
std	545.167641	0.87836	0.482588	0.950078	0.700335	0.166226	0.998106	0.997448	0.996229
min	1.000000	-0.95197	-0.482460	-2.435910	-0.570090	-1.107020	-3.464360	-3.273930	-3.273930
25%	474.000000	-0.95197	-0.482460	-0.611130	-0.570090	-0.316850	-0.678250	-0.695090	-0.717270
50%	946.000000	-0.07854	-0.482460	-0.059210	0.960820	-0.316850	0.042570	0.003320	-0.019280
75%	1417.000000	0.49788	0.482460	0.454680	0.960820	-0.316850	0.629670	0.637790	0.723300
max	1888.000000	2.59171	0.482460	1.984370	0.960820	1.907250	3.273930	3.273930	2.901610

LET'S DISCUSS ABOUT THE VARIABLES

- The 12 first variables are the inputs. A respondent is defined by its age, gender, education level, his current residence country and its ethnicity.
- We also have personnality scores : Nscore (Neuroticism), Escore(Extraversion), Oscore(Openness to experience), Ascore (Agreeableness), Cscore(Conscientiousness), Impulsive(measuring impulsiveness) and SS (sensation seeing).
- All the inputs are categorical but can be considered as real valued after quantification.

- Here are the meanings for the drugs :

-CL0 : Never Used

-CL1 : Used over a Decade Ago

-CL2 : Used in Last Decade

-CL3 : Used in Last Year

-CL4 : Used in Last Month

-CL5 : Used in Last Week

-CL6 : Used in Last Day

LET'S DISCUSS ABOUT THE VARIABLES : MEANING

- Gender:
  - 0.48246 Female
  - 0.48246 -> Male
- Age : Age categrory of the of paticipants:
  - -0.95197 18-24 yo
  - -0.07854 25-34 yo
  - -0.49788 35-44 yo
  - -1.09449 45-54 yo
  - -1.82213 55-64 yo
  - -2.59171 65+ yo
- Eduaction : Level of education :
  - -2.43591 Left school before 16
  - -1.73790 Left school at 16 years
  - -1.43719 Left school at 17 years
  - -1.22751 Left school at 18 years
  - -0.61113 Some college or university, no certificate or degree
  - -0.05921 Professional certificate/ diploma
  - -0.45468 University degree
  - -1.16365 Masters degree
  - -1.98437 Doctorate degree

LET'S DISCUSS ABOUT THE VARIABLES : MEANING

- - Ethnicity : ethnicity of participants:
  - -0.50212 Asian
  - -1.10702 Black
  - -1.90725 Mixed-Black/Asian
  - -0.12600 Mixed-White/Asian
  - -0.22166 Mixed-White/Black
  - -0.11440 Other
  - -0.31685 White

- Country: current residence country :
  - -0.09765 Australia
  - -0.24923 Canada
  - -0.46841 New Zealand
  - -0.28519 Other
  - -0.21128 Republic of Ireland
  - -0.96082 UK
  - -0.57009 USA

THE VARIABLES

- Then we wanted to check variables types and if there's any missing values.
- Object type columns are categorized drugs.
- We can also see that we have no missing values, so we don't need to replace some missing ones.

Out[8]:

ID	int64
Age	float64
Gender	float64
Education	float64
Country	float64
Ethnicity	float64
Nscore	float64
Escore	float64
Oscore	float64
Ascore	float64
Cscore	float64
Impulsive	float64
SS	float64
Alcohol	object
Amphet	object
Amyl	object
Benzos	object
Caff	object
Cannabis	object
Choc	object
Coke	object
Crack	object
Ecstasy	object
Heroin	object
Ketamine	object
Legalh	object
LSD	object
Meth	object
Mushrooms	object
Nicotine	object
Semer	object
VSA	object

dtype: object

Out[9]:

ID	0
Age	0
Gender	0
Education	0
Country	0
Ethnicity	0
Nscore	0
Escore	0
Oscore	0
Ascore	0
Cscore	0
Impulsive	0
SS	0
Alcohol	0
Amphet	0
Amyl	0
Benzos	0
Caff	0
Cannabis	0
Choc	0
Coke	0
Crack	0
Ecstasy	0
Heroin	0
Ketamine	0
Legalh	0
LSD	0
Meth	0
Mushrooms	0
Nicotine	0
Semer	0
VSA	0

dtype: int64

THE VARIABLES

- Now if decided to transform some columns into categories to have a better visualization after. So I've definded which ones are categoricals or not :

In [10]:
```
categoricals = ['Age', 'Gender', 'Education', 'Country'
                , 'Ethnicity', 'Alcohol'
                , 'Amphet', 'Amyl', 'Benzos', 'Caff', 'Cannabis'
                , 'Choc', 'Coke', 'Crack', 'Ecstasy', 'Heroin'
                , 'Ketamine', 'Legalh', 'LSD', 'Meth', 'Mushrooms', 'Nicotine', 'Semer', 'VSA']
numericals = list(set(dataset.columns)-set(categoricals))
numericals.remove('ID')
numericals
```

Out[10]:
```
['Impulsive', 'Escore', 'Cscore', 'Ascore', 'Nscore', 'SS', 'Oscore']
```

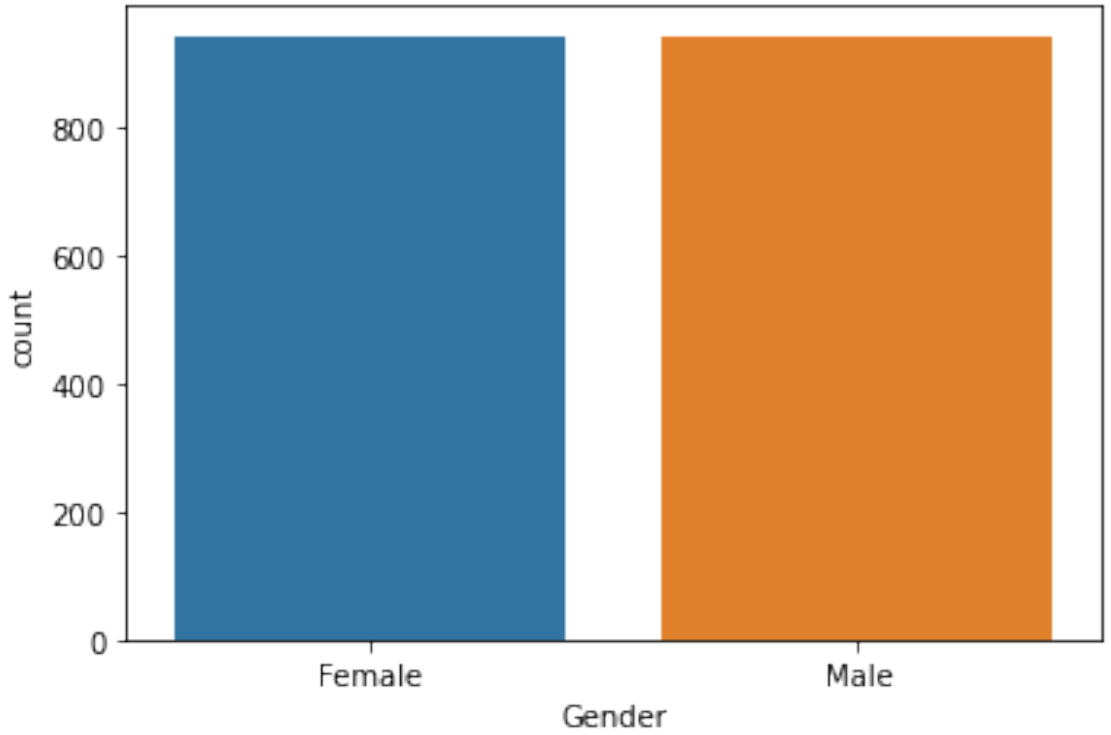
- By following this way, we can transform all the categoricals variables :

In [12]:
```
dataset["Gender"].unique()
d = dict(zip(dataset["Gender"].unique(),["Female","Male"]))
dataset["Gender"].replace(d,inplace=True)
```

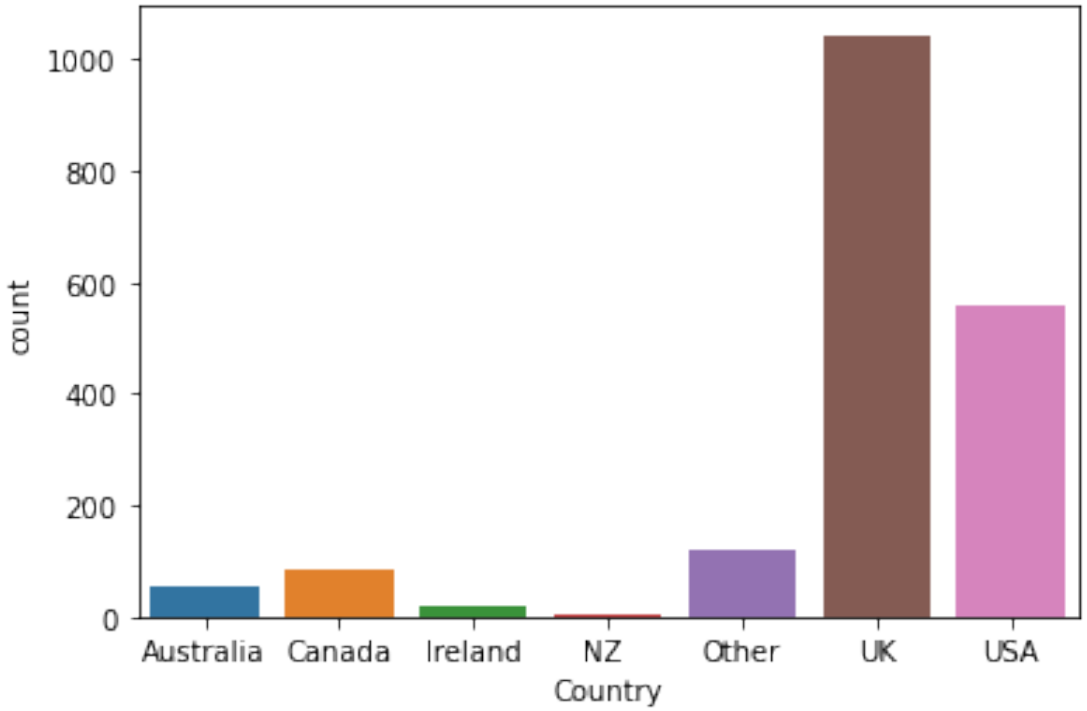
In [13]:
```
#replacing age categories

temp=dict(zip(dataset["Age"].unique(),["35-44","25-34","18-24","65+","45-54","55-64"]))
dataset["Age"].replace(temp,inplace=True)
```

VISUALIZATION : INPUTS

Gender distribution



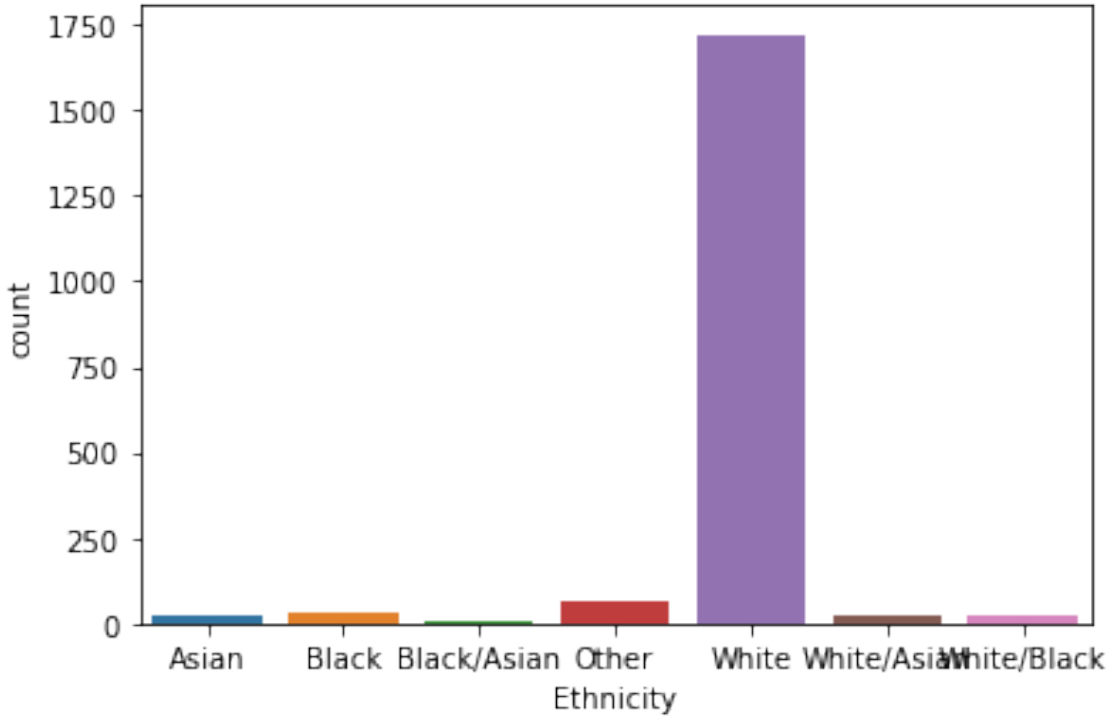
Current residence country distribution



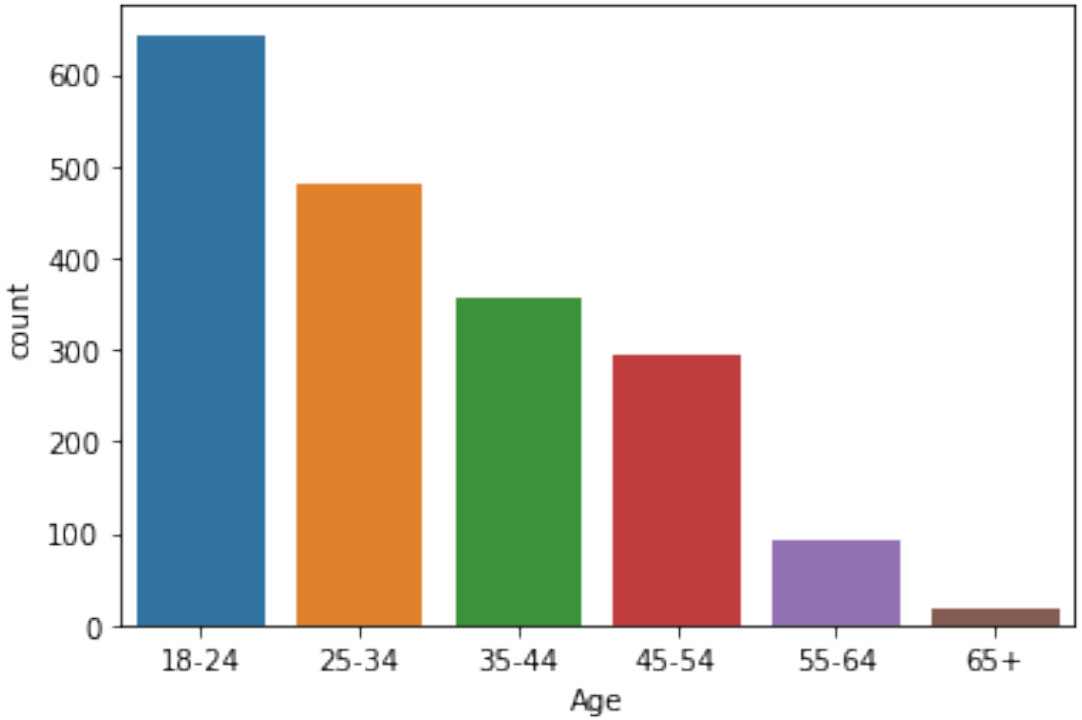
- We can see that we have almost the same number of Male and Female.
- In this dataset, Americans and English are the most represented.

VISUALIZATION : INPUTS

Ethnicity distribution



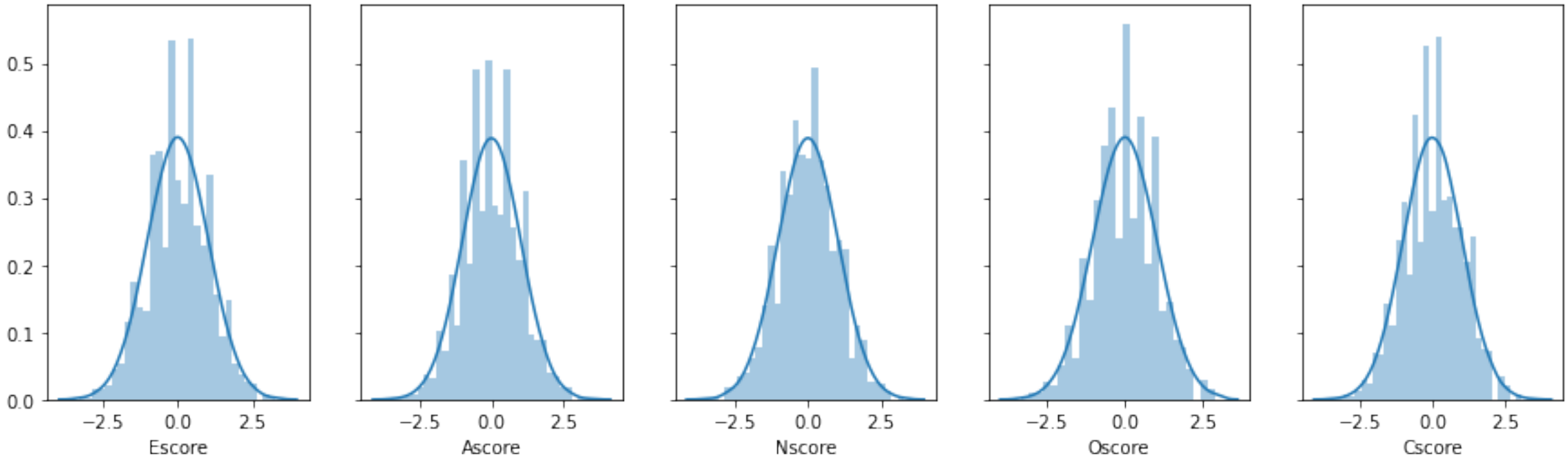
Age distribution



- There's a huge proportion of white people here.
- The older the people are , the less they are represented.

VISUALIZATION : INPUTS

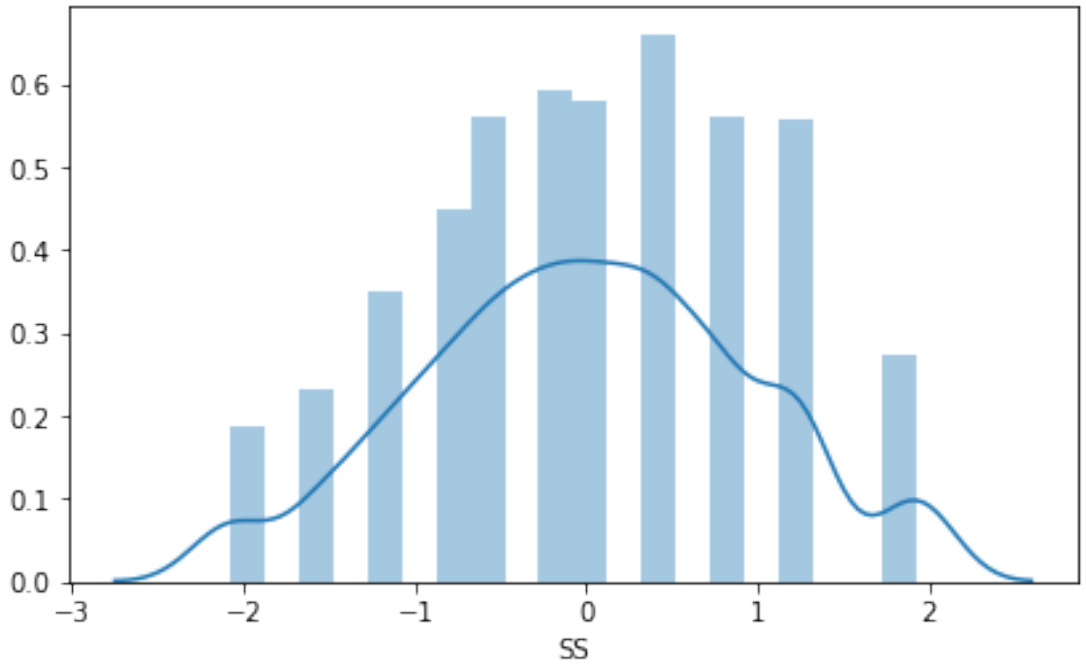
Personnality score ditributions

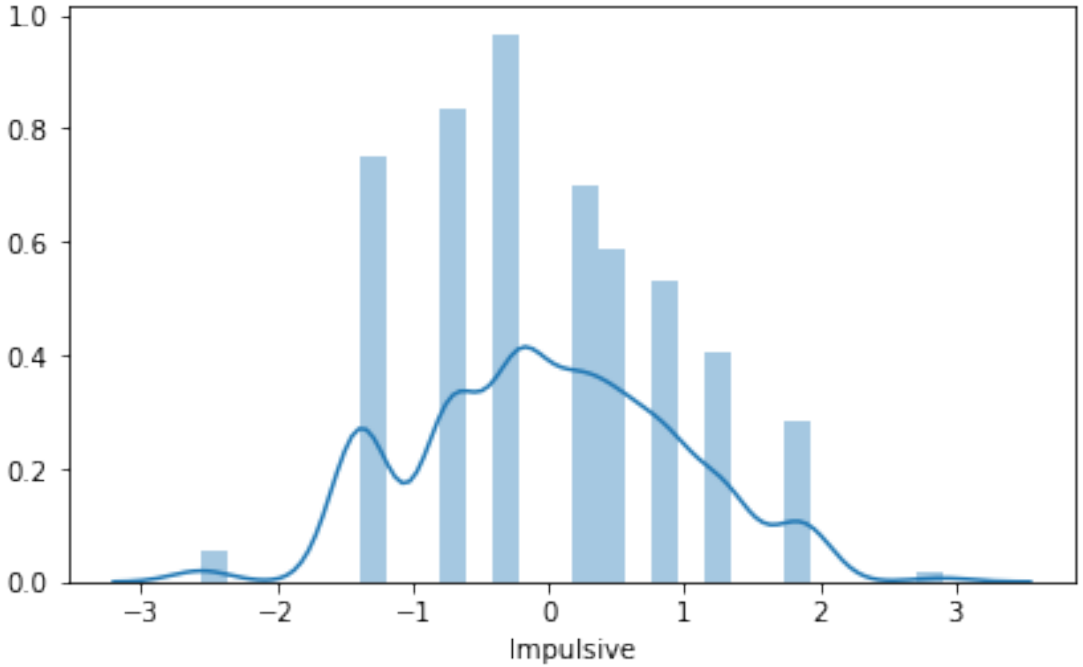


- Score distributions tend to be Gaussian distributions.

VISUALIZATION : INPUTS

sensation seeing and impulsiveness distributions

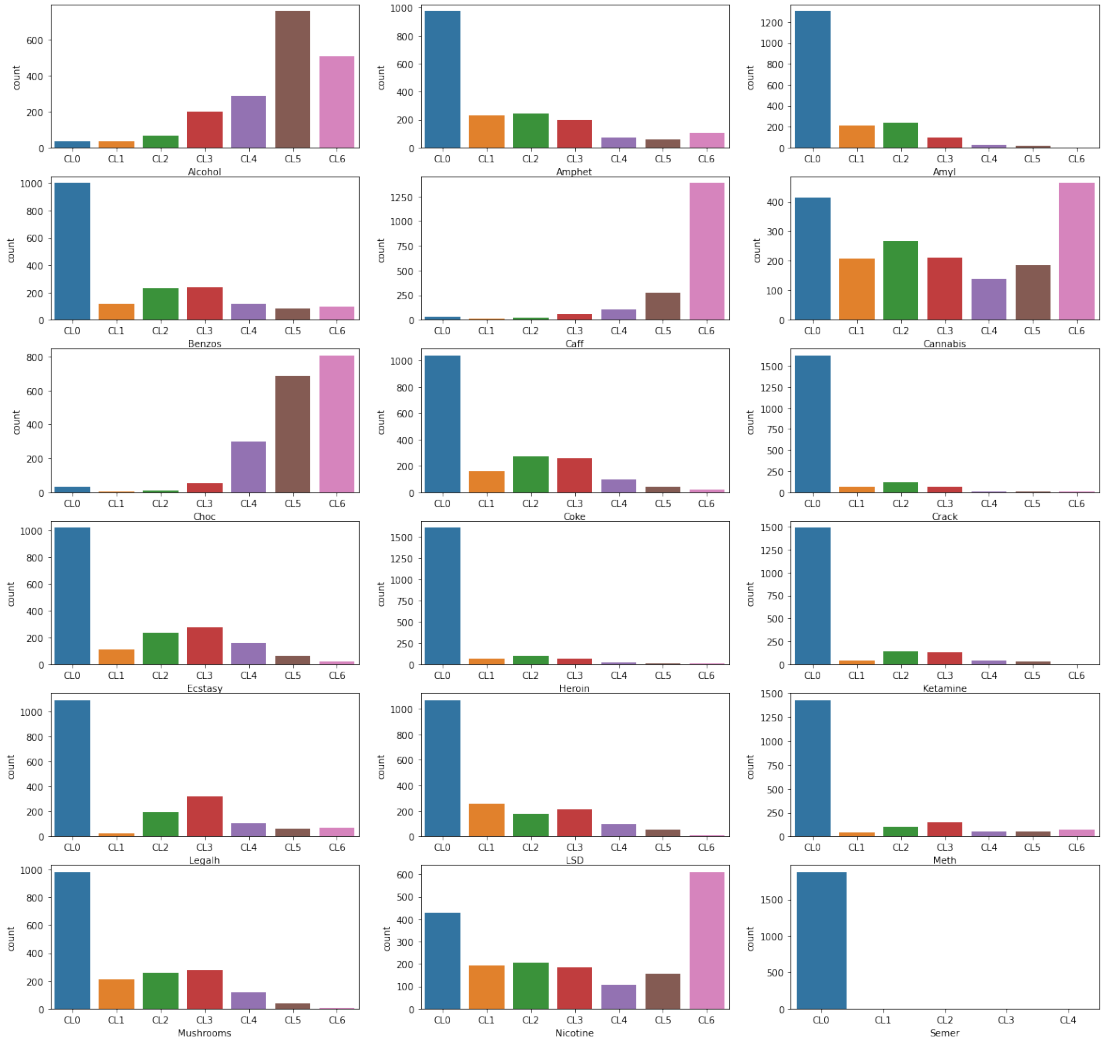




- Here distribution are less like Gaussian distributions, may be it's because the values are less continous.

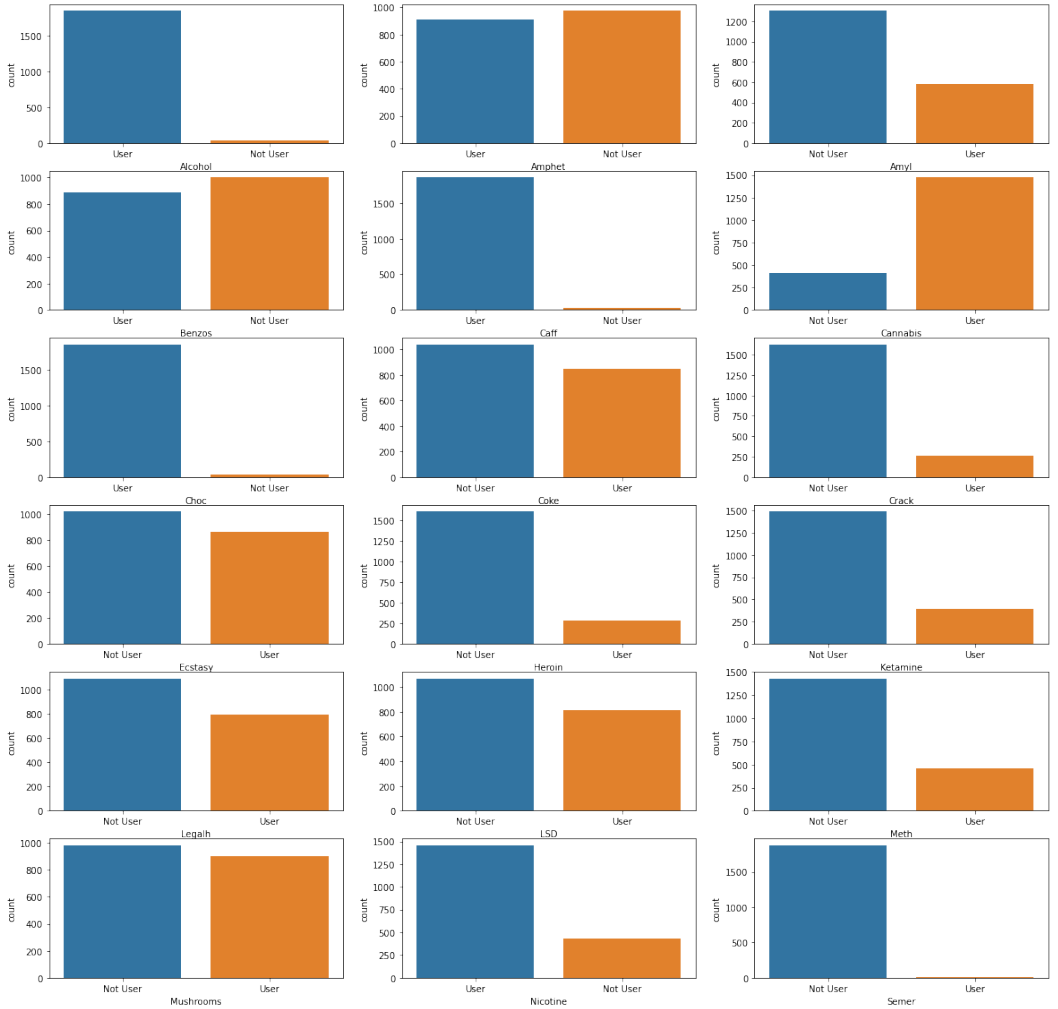
VISUALIZATION : DRUGS

- Visualisation of the repartion of users classes for each drug.
- Legal or ”almost legal” (Cannabis) drugs seems to have been a lot experienced.



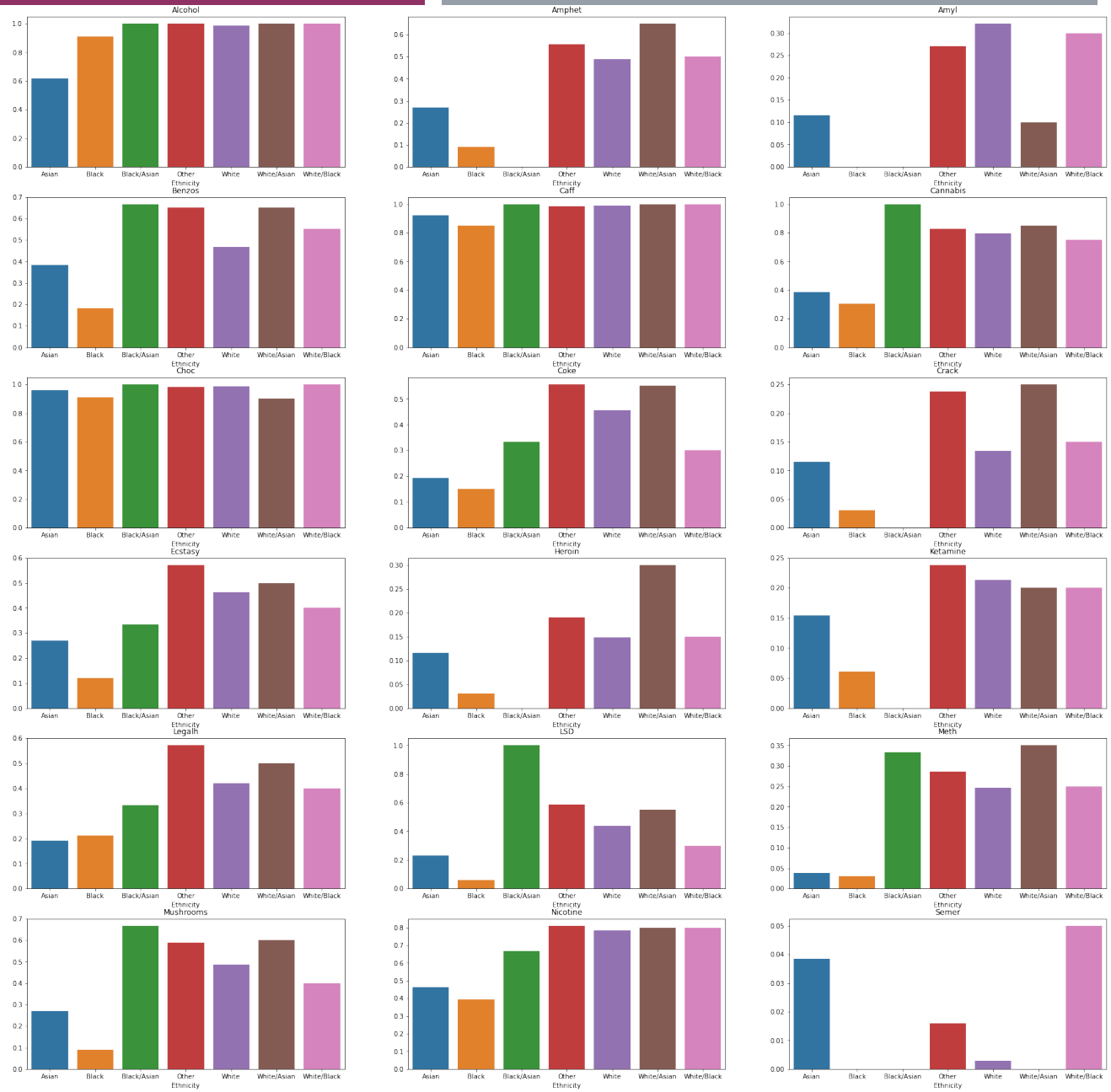
VISUALIZATION : DRUGS

- Proportion of users and not users for each drug:
- We can draw the same conclusion as we did just before, here the proportions are more relevant.
- Some drugs like ectasy, mushrooms, LSD, coke have also been a lot experienced at least one time.



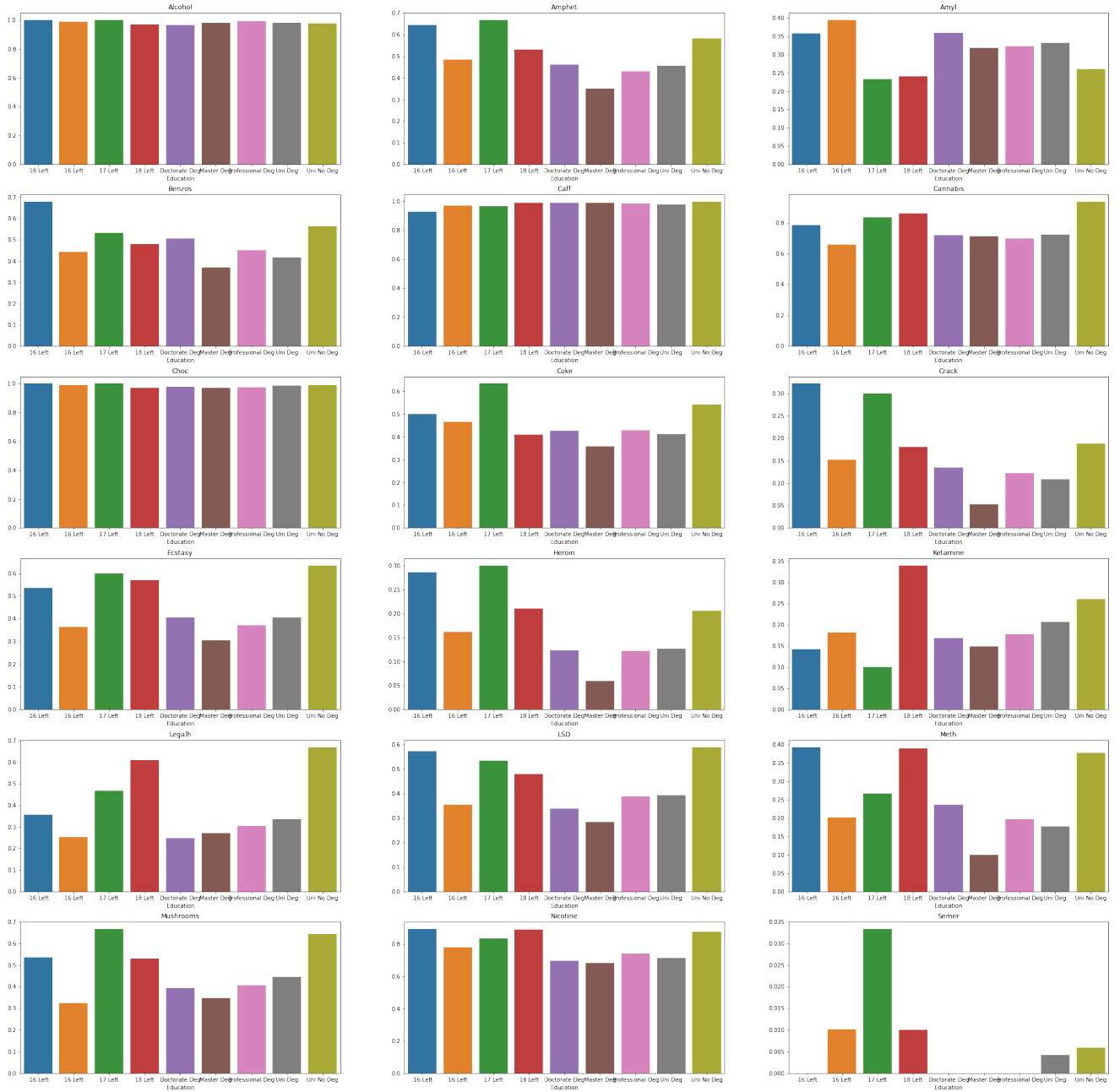
Here we can see the influence of the ethnicity on drug uses.

To visualize the proportions between ethnicities, because a huge part of our individuals are white, we need to have the ratio between the users of an ethnicity and the non users



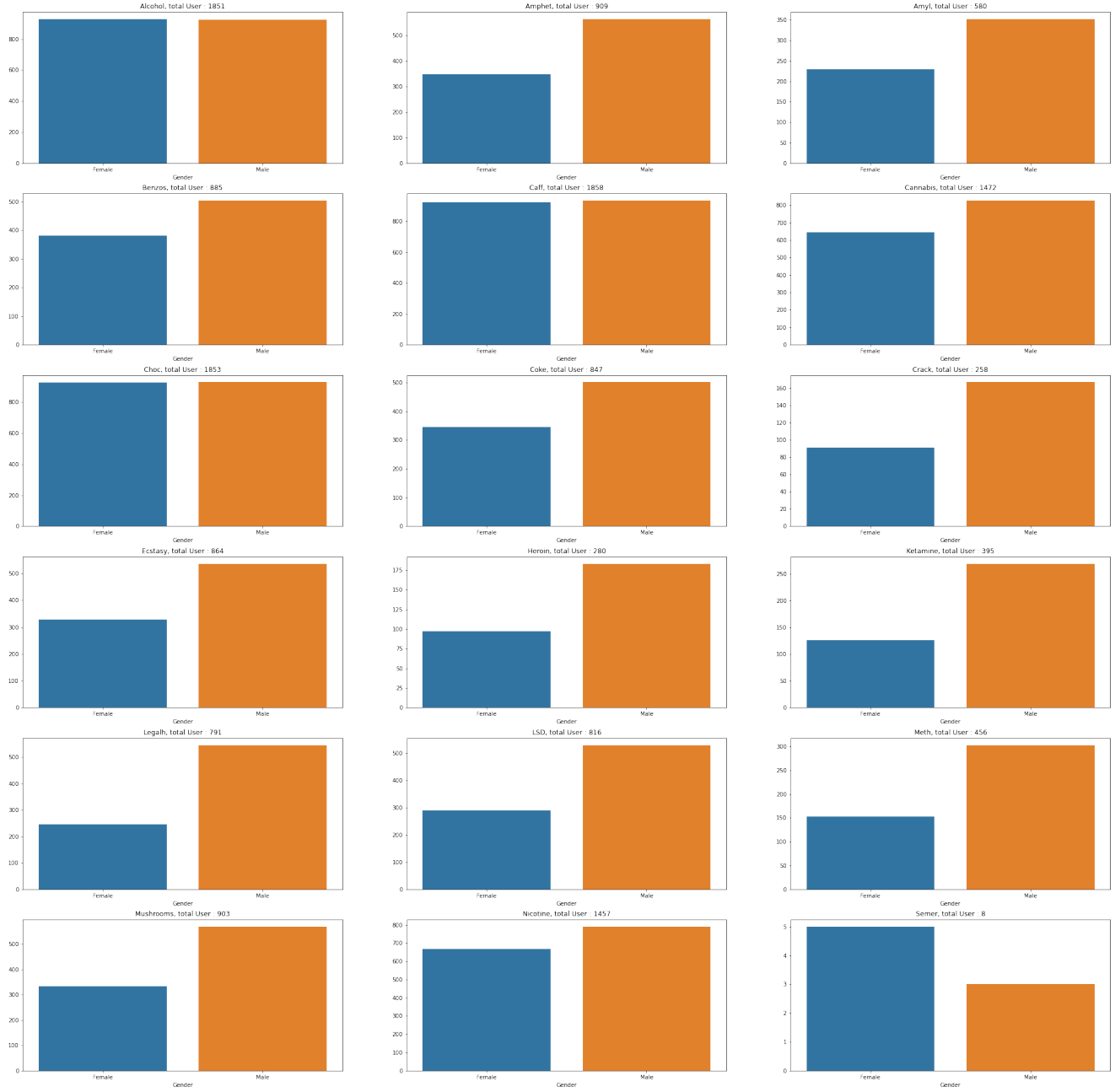
Finaly I wanted to see the influence of the education background on experiencing drugs

Here we can see that for most of the drugs , study background is not really determinant about experiencing drugs



To end with visualization, we can try to see the influence of gender about taking a drug at least one time .

It seems like mens are more likely to try drugs rather the womens.



PREPROCESSING

- To deploy Machine Learning on my dataset, I decided to do some transformation on it.
- Firstly, I transformed the drugs categorical variables, by using an Encoder from Scikit Learn, to have real values.
- Secondly, I've dropped the "ID" feature considering it was useless.

```
from sklearn.preprocessing import LabelEncoder

col =list(dataset[drugs])
for c in col:
    le = LabelEncoder()
    data[c] = le.fit_transform( data[c])
```

```
data.drop("ID",axis=1,inplace=True)
```

PREPROCESSING

- So here is my dataset after the previous transformations :

```
In [37]: data
```

Out[37]:

	Age	Gender	Education	Country	Ethnicity	Nscore	Escore	Oscore	Ascore	Cscore	...	Ecstasy	Heroin	Ketamine	Legalh	LSD	Meth
0	0.49788	0.48246	-0.05921	0.96082	0.12600	0.31287	-0.57545	-0.58331	-0.91699	-0.00665	...	0	0	0	0	0	0
1	-0.07854	-0.48246	1.98437	0.96082	-0.31685	-0.67825	1.93886	1.43533	0.76096	-0.14277	...	4	0	2	0	2	3
2	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.46725	0.80523	-0.84732	-1.62090	-1.01450	...	0	0	0	0	0	0
3	-0.95197	0.48246	1.16365	0.96082	-0.31685	-0.14882	-0.80615	-0.01928	0.59042	0.58489	...	0	0	2	0	0	0
4	0.49788	0.48246	1.98437	0.96082	-0.31685	0.73545	-1.63340	-0.45174	-0.30172	1.30612	...	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1880	-0.95197	0.48246	-0.61113	-0.57009	-0.31685	-1.19430	1.74091	1.88511	0.76096	-1.13788	...	0	0	0	3	3	0
1881	-0.95197	-0.48246	-0.61113	-0.57009	-0.31685	-0.24649	1.74091	0.58331	0.76096	-1.51840	...	2	0	0	3	5	4
1882	-0.07854	0.48246	0.45468	-0.57009	-0.31685	1.13281	-1.37639	-1.27553	-1.77200	-1.38502	...	4	0	2	0	2	0
1883	-0.95197	0.48246	-0.61113	-0.57009	-0.31685	0.91093	-1.92173	0.29338	-1.62090	-2.57309	...	3	0	0	3	3	0
1884	-0.95197	-0.48246	-0.61113	0.21128	-0.31685	-0.46725	2.12700	1.65653	1.11406	0.41594	...	3	0	0	3	3	0

PREPROCESSING

- By displaying the correlation matrix between our variables (Please check it on the notebook) , I've decided to predict Cannabis consumption with the following features :
- I've also split my dataset into a test set and a training set with Scikit Learn.

```
features = ['Age', 'Gender', 'Education', 'Country', 'Ethnicity', 'Nscore',
       'Escore', 'Oscore', 'Ascore', 'Cscore', 'Impulsive', 'SS']
target = ["Cannabis"]
```

```
from sklearn.model_selection import train_test_split
X = data[features]
y = data[target].astype("category")


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

MACHINE LEARNING : KNN

- Firstly I've tried to predict the 6 classes for the target Cannabis , to see what it can lead to.

In [41]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,precision_score
from sklearn.neighbors import KNeighborsClassifier
mod = KNeighborsClassifier()
mod.fit(X_train,y_train.values.ravel())
pred = mod.predict(X_test)
accuracy_score(y_test,pred)
```

Out[41]:
```
0.3395225464190981
```

- As you can see the prediction is really bad.

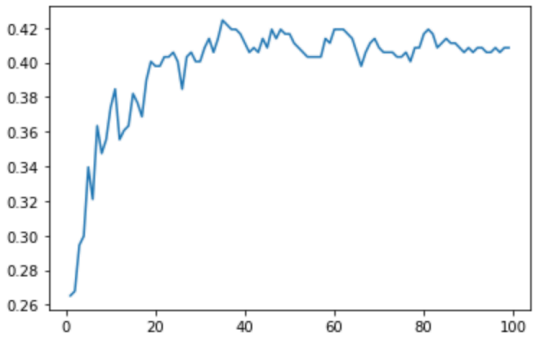
MACHINE LEARNING : KNN

- I've tried to optimize it with finding the best number of neighbors and to see the impact on the accuracy.

- It seems it's not this path we had to follow to obtain good predictions.

In [42]:
```
from sklearn.neighbors import KNeighborsClassifier


graph= dict()
for i in range(1,100):
    temp = KNeighborsClassifier(n_neighbors=i)
    temp.fit(X_train,y_train.values.ravel())
    pred_temp = temp.predict(X_test)

    graph[i]=accuracy_score(y_test,pred_temp)

x=list(graph.keys())
y=list(graph.values())
plt.plot(x,y)
sorted_d = sorted(graph.items(), key=lambda x: x[1],reverse=True)
sorted_d[0]
```

Out[42]: (35, 0.4244031830238727)



MACHINE LEARNING

- So it leads me to create a new target : User_Canna. If a person has ever taken this drug one time, he will be categorized as an User. If not he'll be a Not User.

In [105]:
```
element = [ 'Not User' if e=="CL0" else 'User'for e in dataset["Cannabis"] ]
```

In [45]:
```
data["User_Canna"]= element
```

In [46]:
```
data.replace({"User":True,'Not User':False},inplace=True)
```

In [47]:
```
data.User_Canna.value_counts()
```

Out[47]:
```
True     1472
False     413
Name: User_Canna, dtype: int64
```

MACHINE LEARNING

- I decided to scale the features too and I split again my dataset with my new target and my scaled features :

```
#Scalling datas
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data[features])
data[features]=scaler.transform(data[features])
```

```
from sklearn.model_selection import train_test_split
X2 = data[features]
y2 = data["User_Canna"]


X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.20)
```

MACHINE LEARNING : THE MODELS

- So here are the 4 model I chose: KNN , Logistic Regression, Gradient Boosting and Random Forest

```
kfinal = KNeighborsClassifier(sorted_d[0][0])
kfinal.fit(X_train2,y_train2.values.ravel())
yfinal = kfinal.predict(X_test2)
kscores= precision_score(y_test2,yfinal),accuracy_score(y_test2,yfinal
kscores
```

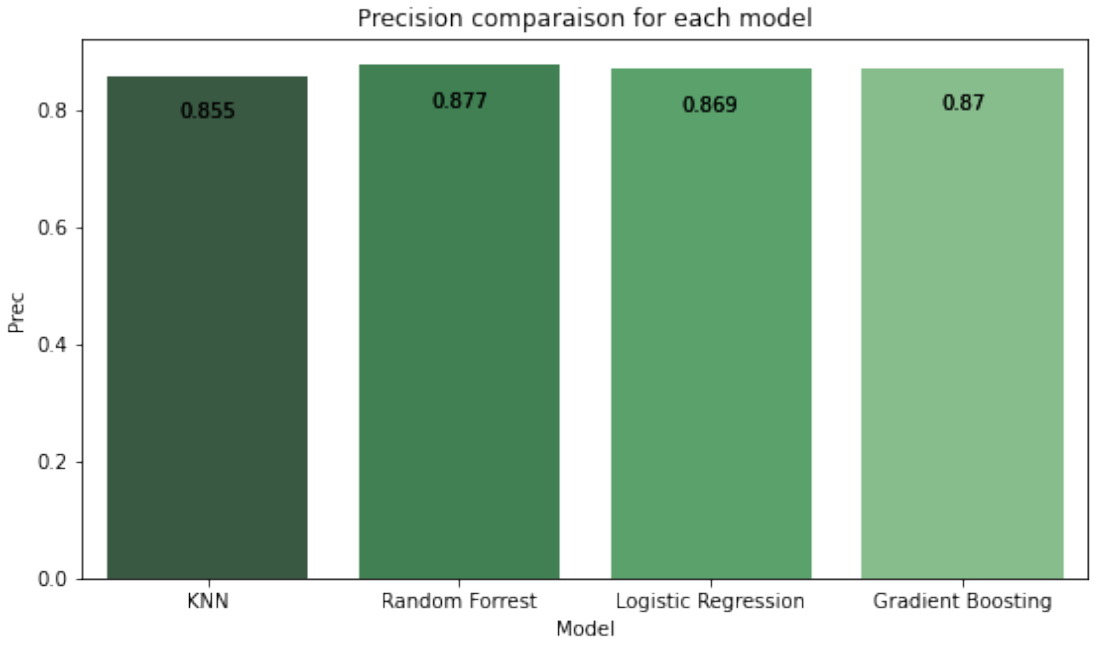
```
from sklearn.linear_model import LogisticRegression
Lr = LogisticRegression(max_iter=100)
Lr.fit(X_train2,y_train2)
y_rf = Lr.predict(X_test2)
rdscores = precision_score(y_test2,y_rf), accuracy_score(y_test2,y_rf)
rdscores
```

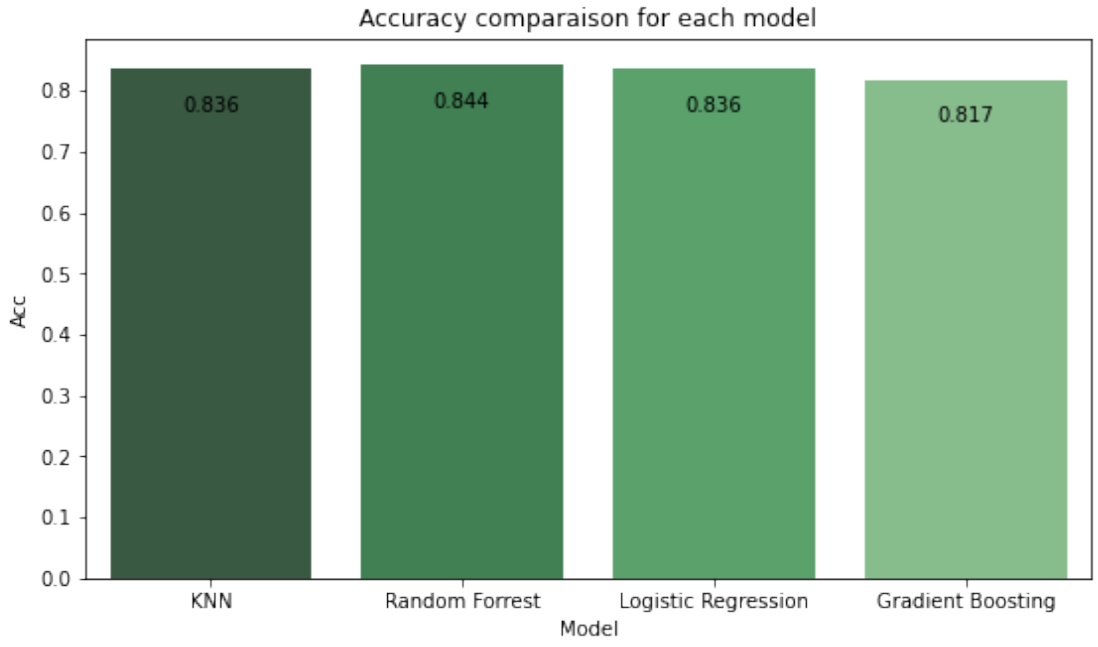
```
from sklearn.ensemble import GradientBoostingClassifier
GBC = GradientBoostingClassifier()
GBC.fit(X_train2,y_train2)
pred_GBC = GBC.predict(X_test2)
GBCscores=precision_score(y_test2,pred_GBC),accuracy_score(y_test2,pred_GBC)
GBCscores
```

```
rf = RandomForestClassifier()
rf.fit(X_train2,y_train2)
rf_predict=rf.predict(X_test2)
logscores = precision_score(y_test2,rf_predict), accuracy_score(y_test2,rf_predict)
logscores
```

MACHINE LEARNING : COMPARING THE MODELS

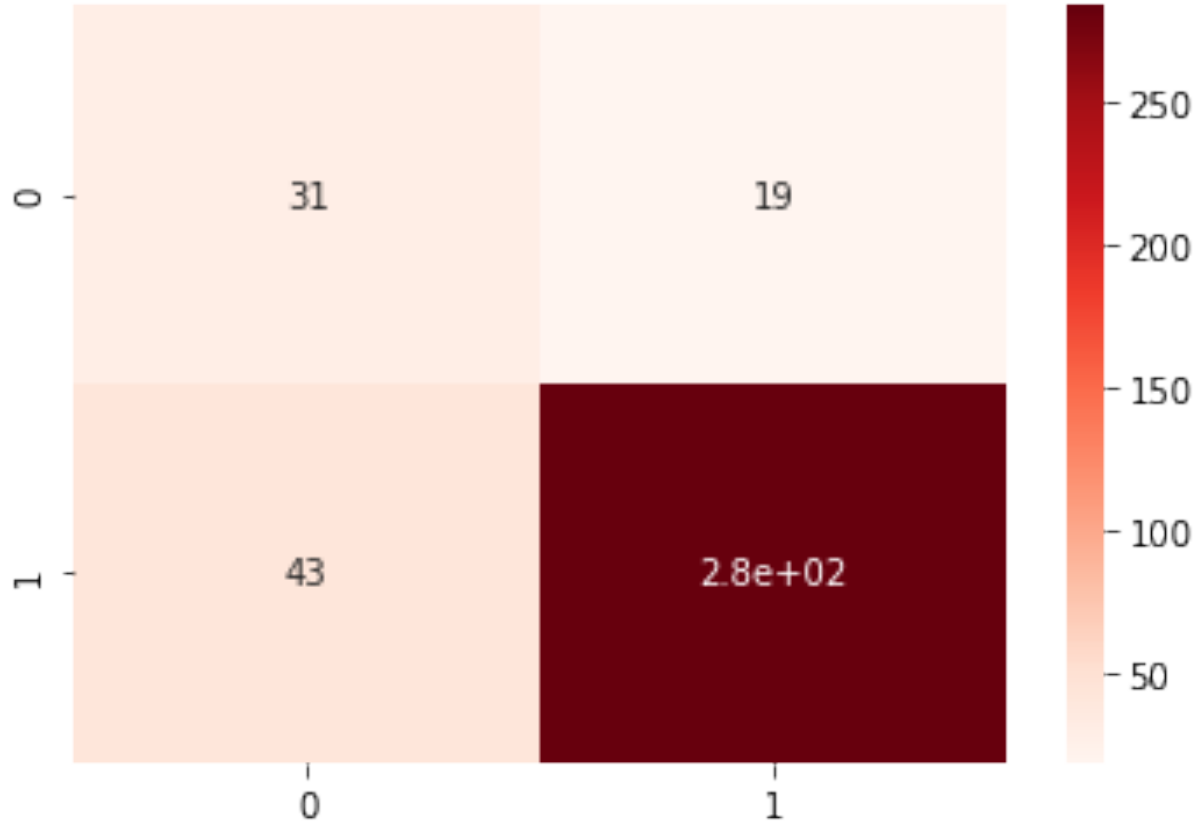
- We can see the performances of the four previous models applied on our dataset.
- Random Forrest seems to be the most efficient here.





MACHINE LEARNING : RANDOM FOREST

- Here is my confusion matrix for the Random Forest model
- We can see that user are well predicted.



MACHINE LEARNING : IMPROVING RANDOM FOREST

- To improve my model I think of using a grid search in order to tune the hyperparameters to have a better accuracy and precision.

```
param_grid = {
    'bootstrap': [True],
    'criterion': ['gini','entropy'],
    'oob_score': [True],
    'max_depth': range(1,12),
    'max_features': ['sqrt'],
    'n_estimators': [75, 100, 200, 300, 500]
}
```

```
grid_search = GridSearchCV(estimator = rf, scoring = ['accuracy','precision'], param_grid = param_grid,
                           cv = 5, n_jobs = -1, verbose = 2, refit='precision')
```

MACHINE LEARNING : IMPROVING RANDOM FOREST

- Here are the best hyperparameters for the Random Forest :

- Best estimator:

  RandomForestClassifier(criterion='entropy', max_depth=11, max_features='sqrt', oob_score=True)

```
grid_search.fit(X_train2, y_train2)
```

```
Fitting 5 folds for each of 110 candidates, totalling 550 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   15.3s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done 550 out of 550 | elapsed:  3.7min finished
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'bootstrap': [True], 'criterion': ['gini', 'entropy'],
                         'max_depth': range(1, 12), 'max_features': ['sqrt'],
                         'n_estimators': [75, 100, 200, 300, 500],
                         'oob_score': [True]},
             refit='precision', scoring=['accuracy', 'precision'], verbose=2)
```

```
best=grid_search.best_estimator_
```

```
grid_search.best_score_
```

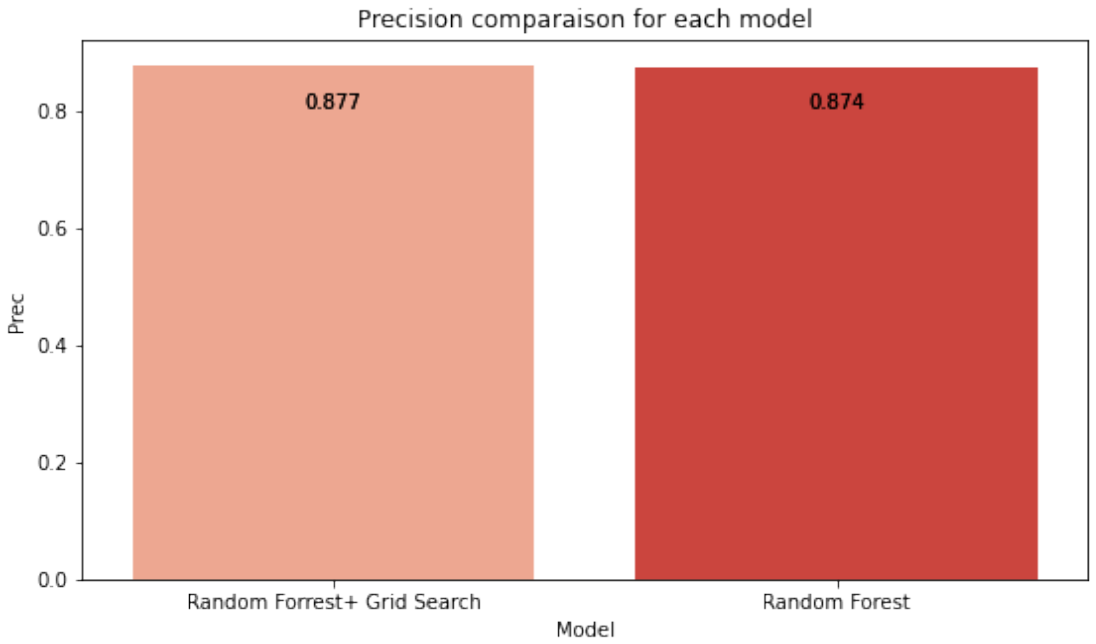
```
0.8462801324848499
```

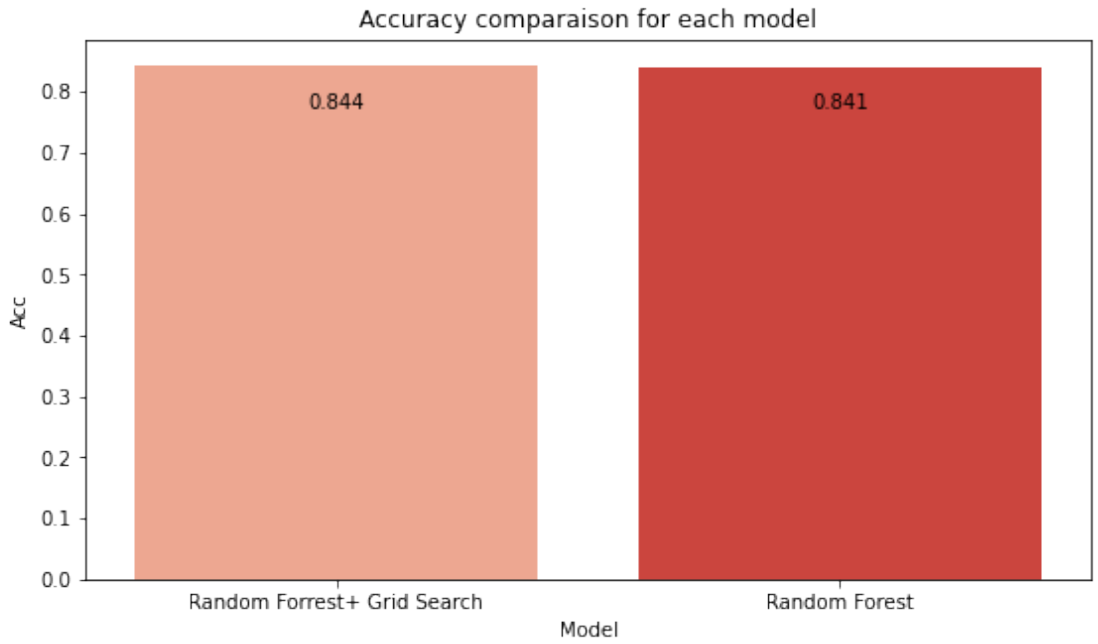
```
params = grid_search.best_params_
params
```

```
{'bootstrap': True,
 'criterion': 'entropy',
 'max_depth': 11,
 'max_features': 'sqrt',
 'n_estimators': 100,
 'oob_score': True}
```

MACHINE LEARNING : IMPROVING RANDOM FOREST

- Comparaison between the two models of Random Forest. We can see there's a little improvment after using a grid search :





THE FLASK API

- To deploy my previous Machine Learning, I decided to work on Flask to build an API.

- 5 files are needed to allow it to work : the main python code (app.py), requirement.txt in which we have all the package dependencies needed to run app.py and 3 html files used to render templates.

THE FLASK API

- Here are to modules required to deploy our API.
- Flask module will allow us to build our API, use the templates we defined and to set the route to access the different responses we want to give to the user.
- Bootstrap gives us templates we can heritate and modify to custom our displays.
- WTF forms bring the tools to fill forms in order to make our predictions with the given datas.
- Joblib, Pandas and ScikitLearn are useful to get the prediction by using our previous ML model.

```
from flask import Flask, render_template,request,redirect
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
from wtforms import StringField, IntegerField , FloatField , SelectField, SubmitField
from wtforms.validators import  DataRequired, InputRequired , NumberRange
from flask_wtf import Form
import joblib
import pandas as pd
import sklearn
```

THE FLASK API

- The 2 routes of our api are here :

- The base route is the welcome page using our template welcome.

- The /predict route is used, in the one hand, to fill the form by render the prediction template and asking user for input and, in the other hand, to display the results by render the result template on which we have our model response.

```
@app.route("/",methods=["GET","POST"])

def Hello():

    return render_template('welcome.html')



@app.route("/predict",methods=["GET","POST"])
```

CONCLUSION

- This dataset was very interresting to analyze. Predicting the drug consumptions given personnality scores could be a powerful tool to prevent drug addictions I think. Nevertheless our dataset is just representative for white American or English people, the population asked isn't enough contrasted.

- What could have been further done: I wish I had more time to deploy other ML models for remaining drugs and optimize the one I've created.