Drugs Consumption

link to dataset :

UCI Machine Learning Repository

Discover datasets around the world!

archive.ics.uci.edu

Jerome HENIN
Jack HEGER
François-Xavier HERANDE

Can we predict which drug a person has consumed based on their personnality ?

I. Data pre-processing

II. Data visualizations

III. Modelisation

IV. Final Modelisation

I. Data pre-processing

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	1	0.49788	0.48246	-0.05921	0.96082	0.12600	0.31287	-0.57545	-0.58331	-0.91699	-0.00665	-0.21712	-1.18084	CL5	CL2	CL0	CL2	CL6	CL0
1	2	-0.07854	-0.48246	1.98437	0.96082	-0.31685	-0.67825	1.93886	1.43533	0.76096	-0.14277	-0.71126	-0.21575	CL5	CL2	CL2	CL0	CL6	CL4
2	3	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.46725	0.80523	-0.84732	-1.62090	-1.01450	-1.37983	0.40148	CL6	CL0	CL0	CL0	CL6	CL3
3	4	-0.95197	0.48246	1.16365	0.96082	-0.31685	-0.14882	-0.80615	-0.01928	0.59042	0.58489	-1.37983	-1.18084	CL4	CL0	CL0	CL3	CL5	CL2

Rename the columns

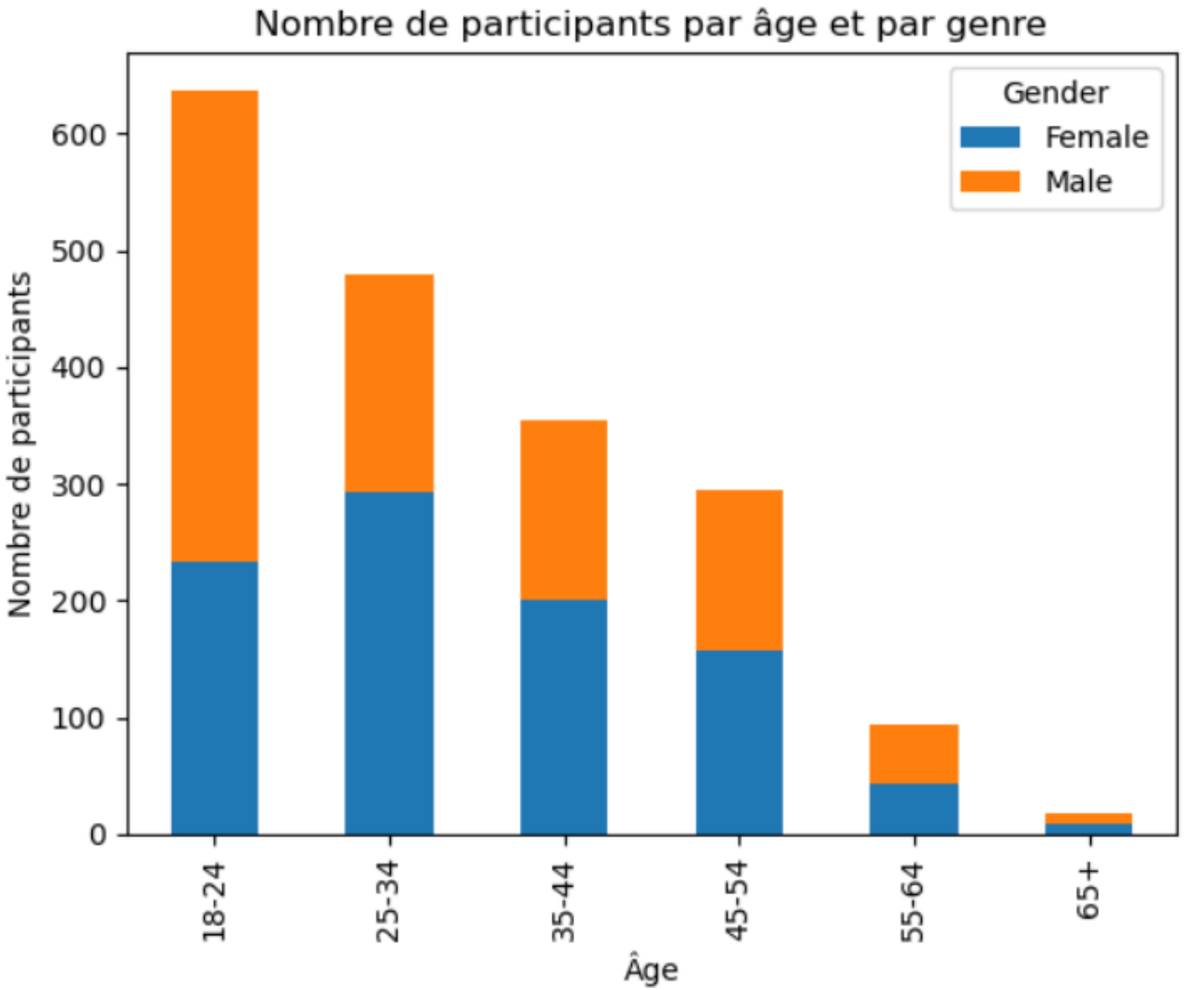
Normlization (already done)

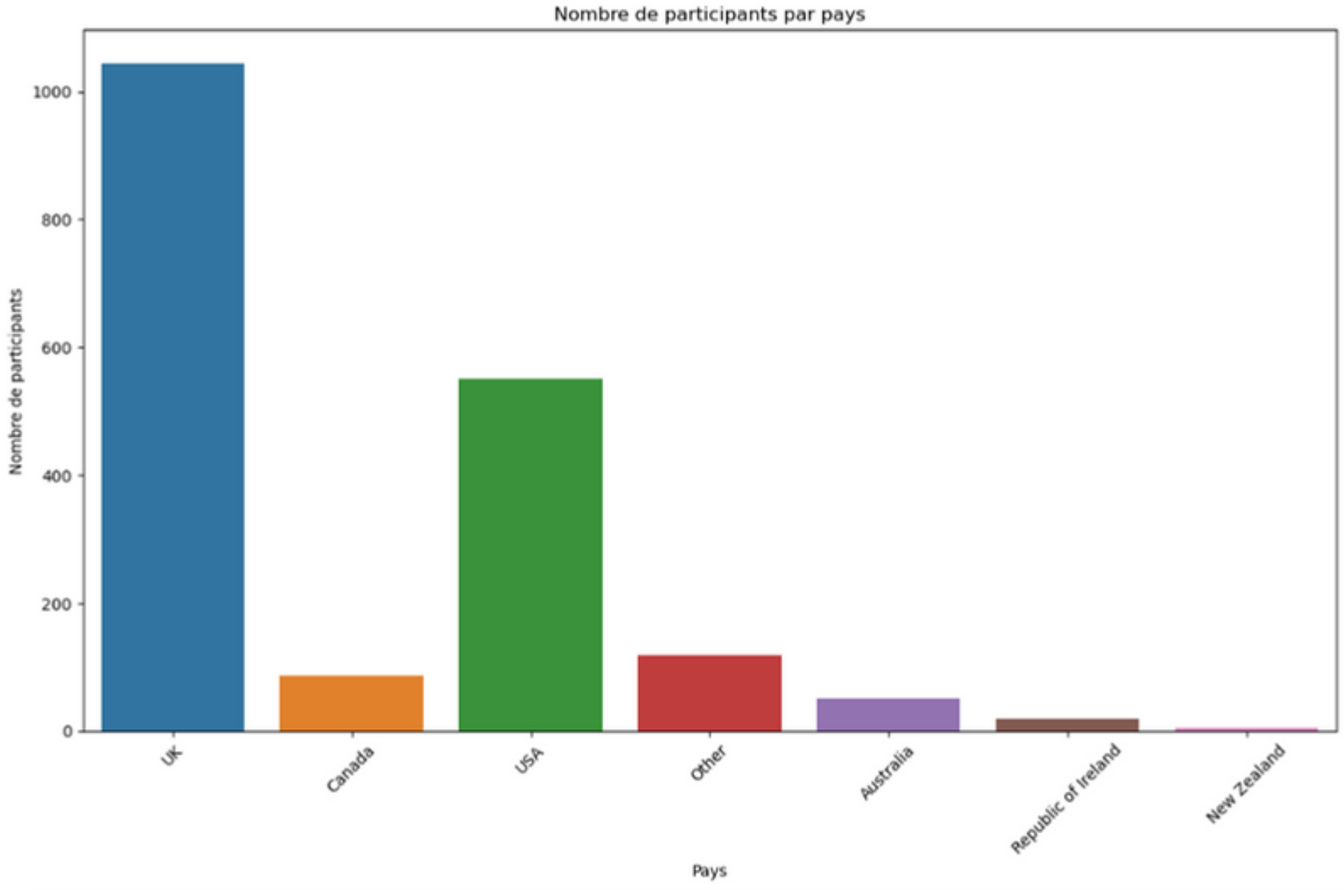
Chek if they are missing values

map to significant values **(Useful for the visualization)**

	ID	Age	Gender	Education	Country	Ethnicity	Nscore	Escore	Oscore	Ascore	Cscore	Impulsive	SS	Alcohol	Amphet	Amyl	Benzo
0	1	35-44	Female	Professional certificate/ diploma	UK	Mixed-White/Asian	39	36	42	37	42	-0.21712	-1.18084	5	2	0	
1	2	25-34	Male	Doctorate degree	UK	White	29	52	55	48	41	-0.71126	-0.21575	5	2	2	
2	3	35-44	Male	Professional certificate/ diploma	UK	White	31	45	40	32	34	-1.37983	0.40148	6	0	0	

II. Data Visualisation

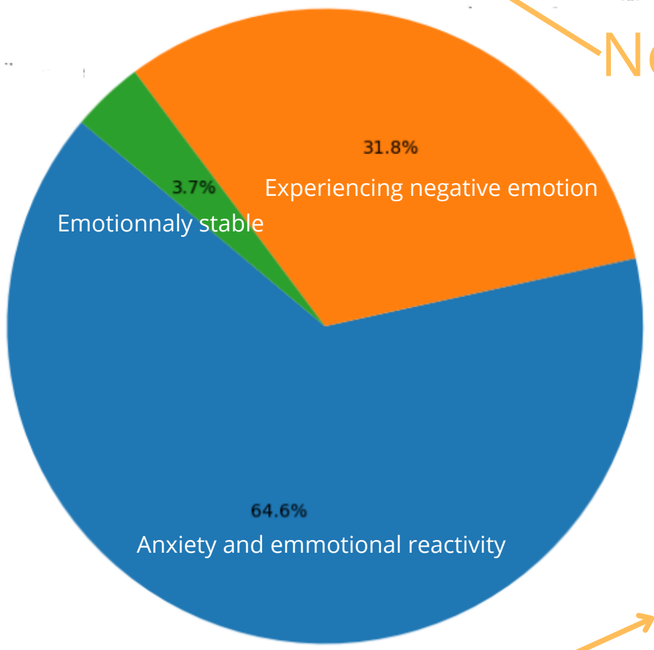




Distribution of each score in the population

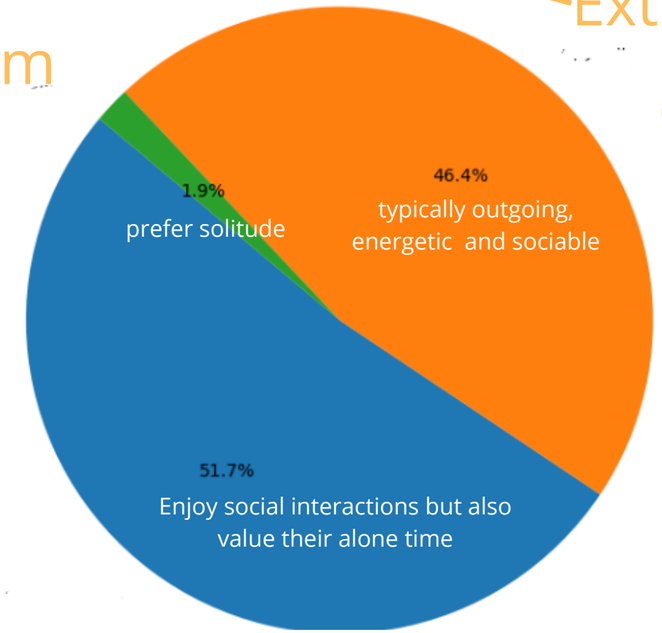
N Score

Neuroticism



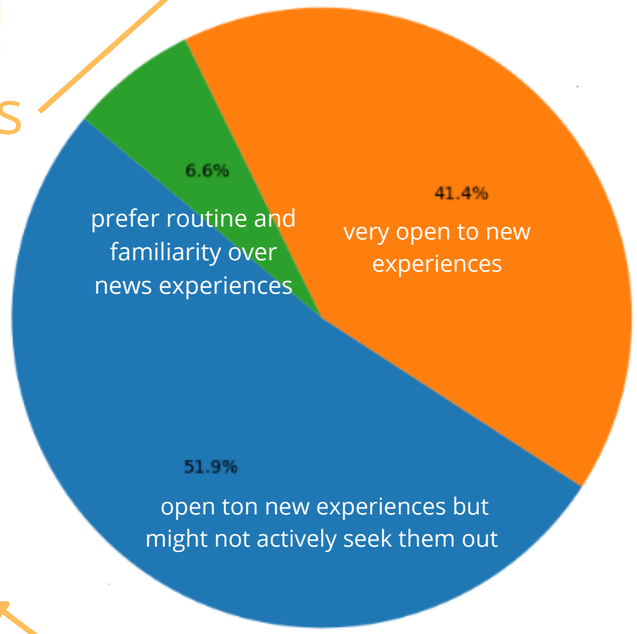
E Score

Extraversion



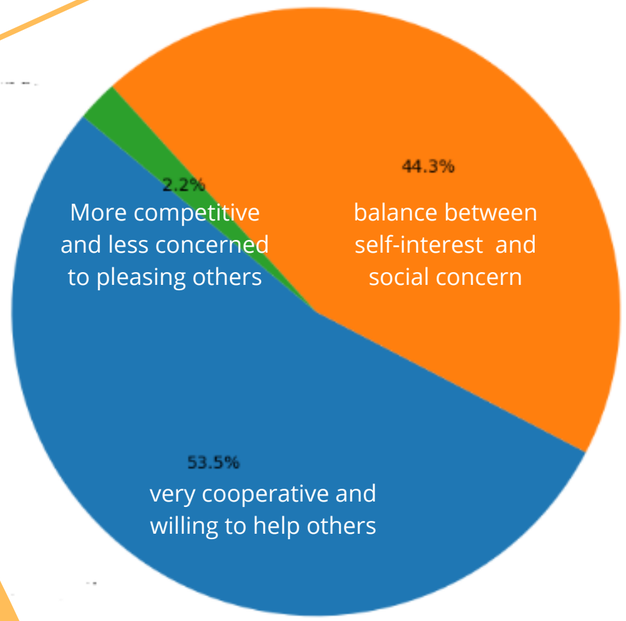
O Score

Openness



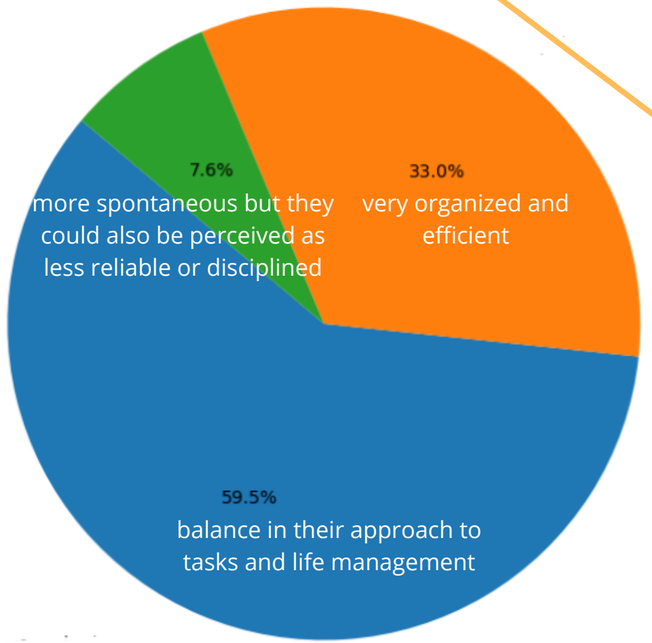
A Score

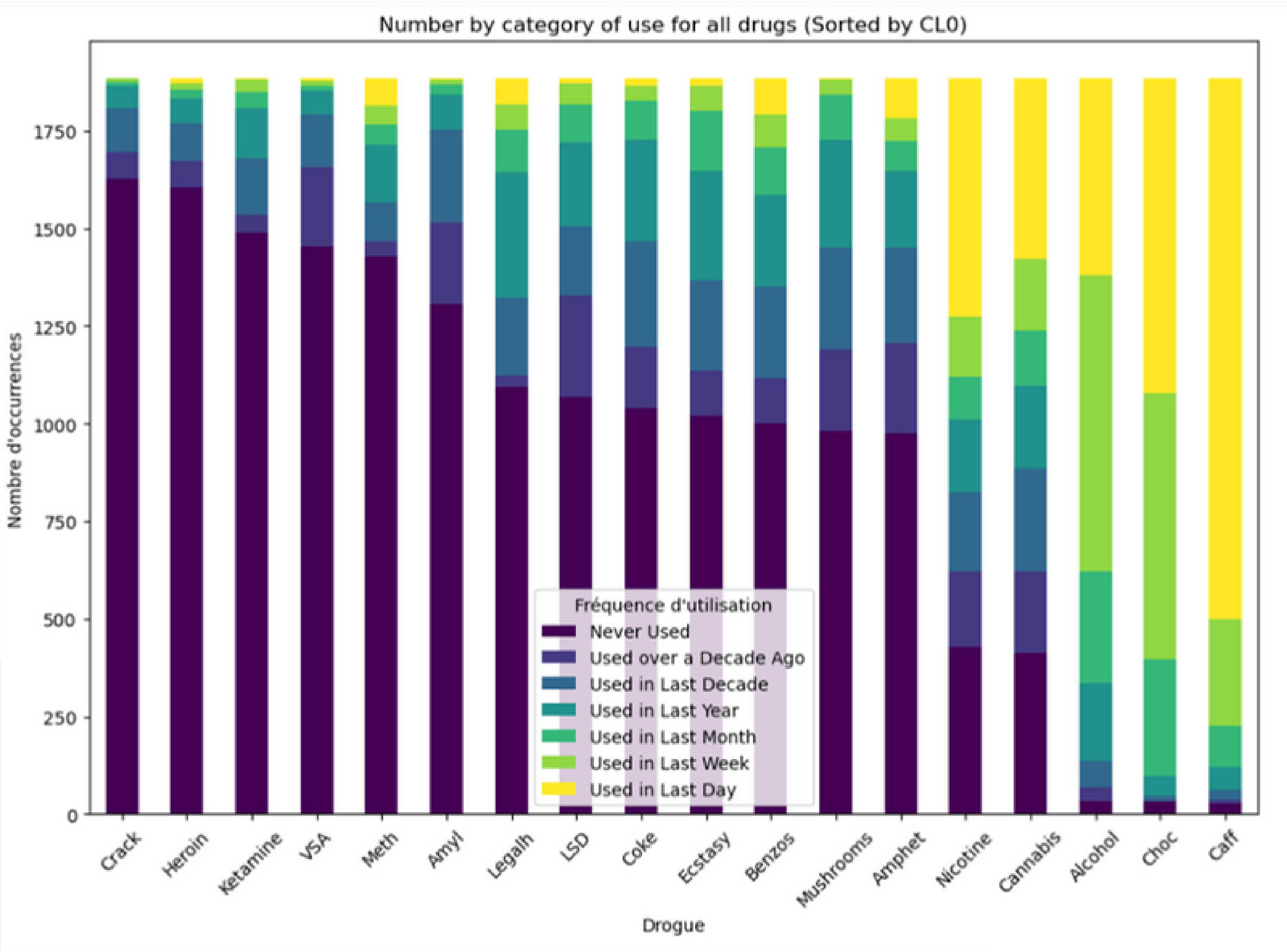
Agreeableness



S Score

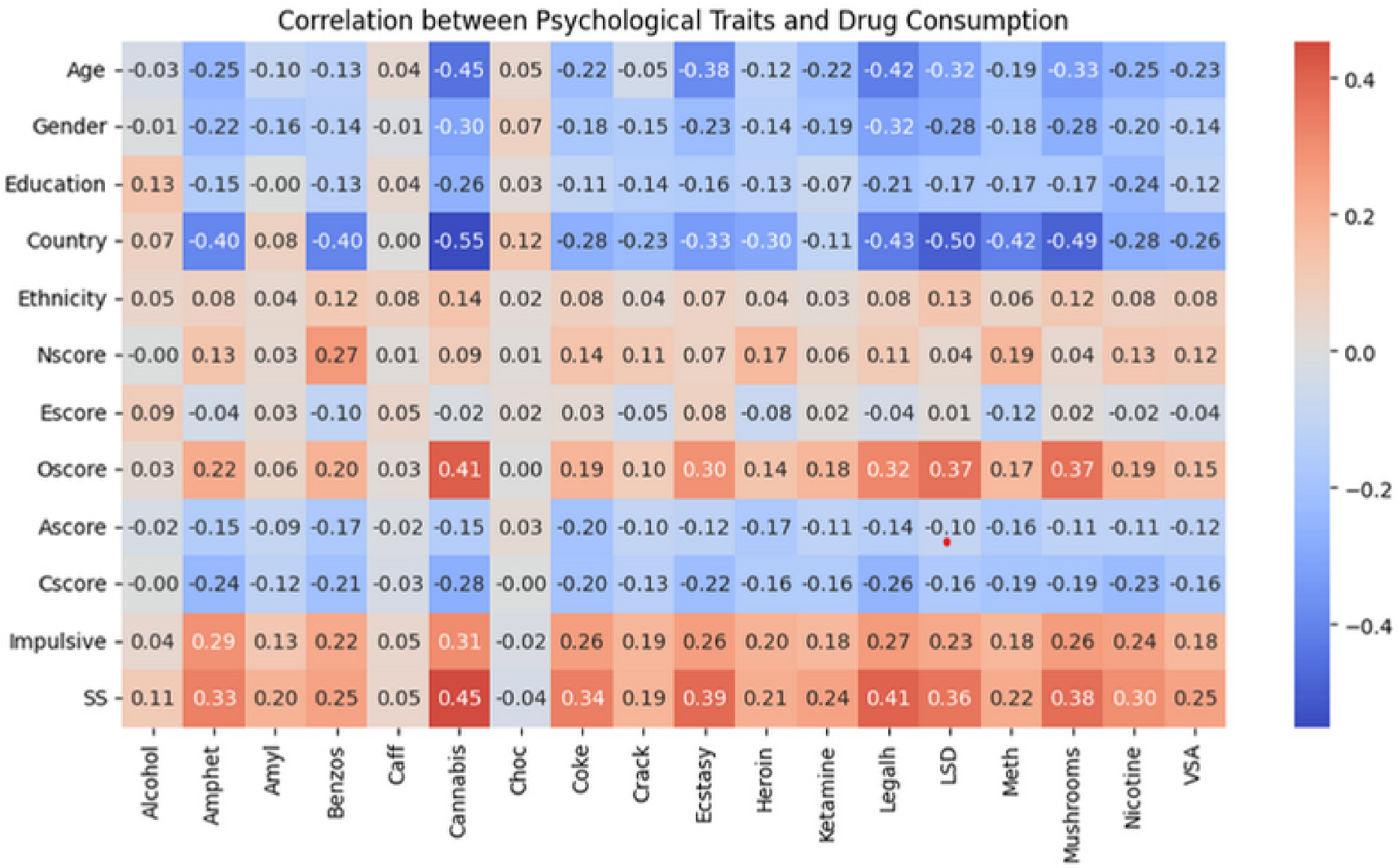
Stability



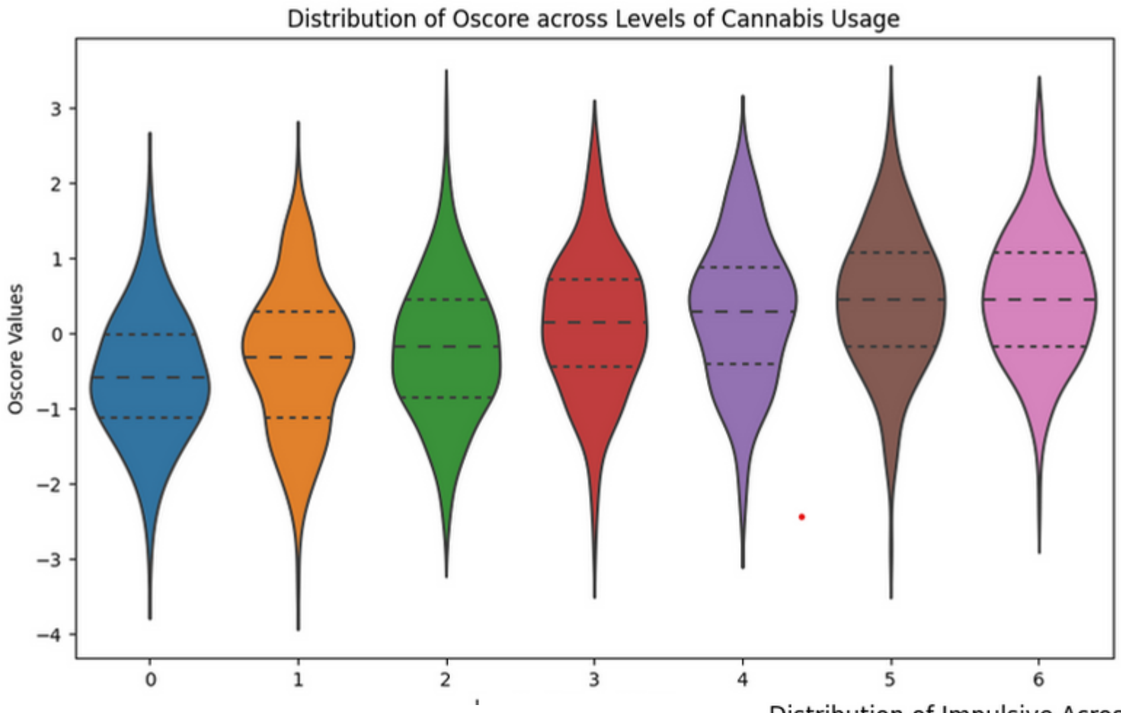

Number by category of use for all drugs (Sorted by CL0)
Nombre d'occurrences
0
250
500
750
1000
1250
1500
1750
Crack
Heroin
Ketamine
VSA
Meth
Amyl
Legalh
LSD
Coke
Ecstasy
Benzos
Mushrooms
Amphet
Nicotine
Cannabis
Alcohol
Choc
Caff
Drogue
Fréquence d'utilisation
Never Used
Used over a Decade Ago
Used in Last Decade
Used in Last Year
Used in Last Month
Used in Last Week
Used in Last Day

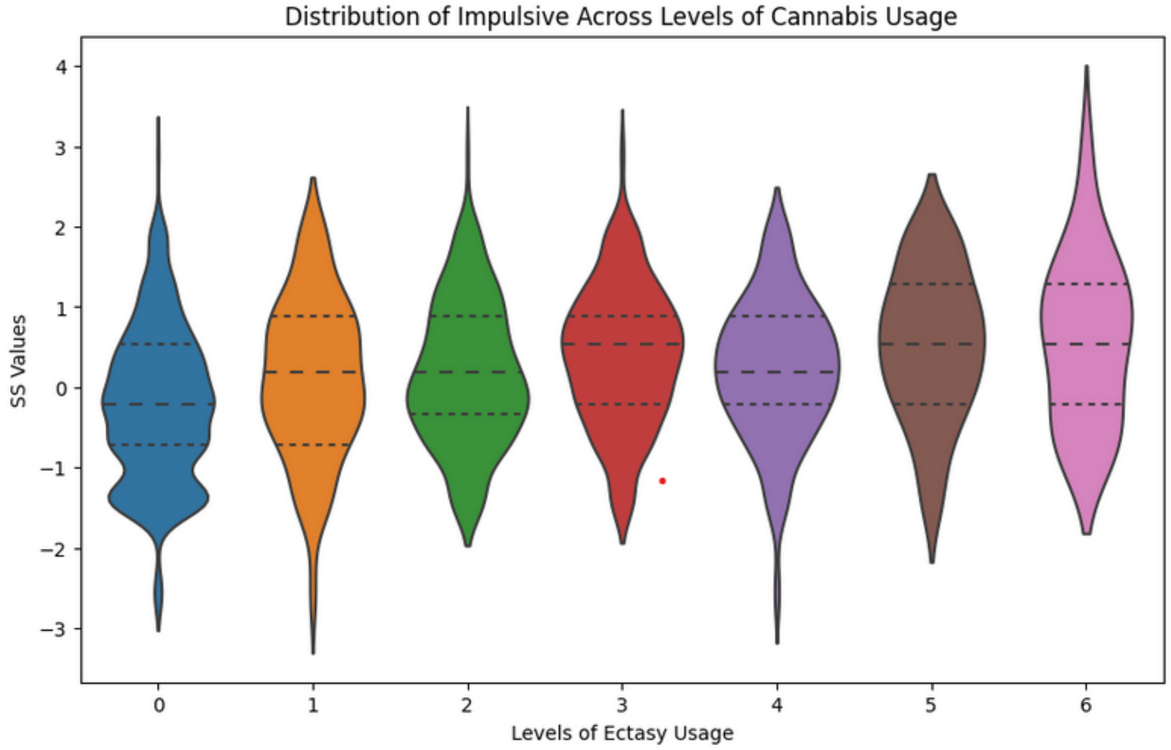
III. Modelisation

Look for correlation between features and drugs consumption:



Focus on


Distribution of Oscore across Levels of Cannabis Usage
Oscore Values
3
2
1
0
−1
−2
−3
−4
0
1
2
3
4
5
6


Distribution of Impulsive Across Levels of Cannabis Usage
SS Values
4
3
2
1
0
−1
−2
−3
0
1
2
3
4
5
6
Levels of Ectasy Usage

Lets try to predict a the SINGLE drug consumed.
Thanks to the correlation table, we can see that the correlations of each drug taken separately are low.

Implementation of knn algorithm to see the precision :

```
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import StratifiedKFold

selected_drug = 'LSD'
features = ["Nscore", "Escore", "Oscore", "Ascore", "Cscore", "Impulsive", "SS"]
X = df[features]
y = df[selected_drug]

knn_model = KNeighborsClassifier(n_neighbors=5)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
accuracy = cross_val_score(knn_model, X, y, cv=cv, scoring='accuracy')

print(f"Mean precision with cross validation : {accuracy.mean():.2f}")
```

```
Mean precision with cross validation : 0.53
```

Using this precedent model, we do not find a reliable model (i.e., around 95%). This is why it is necessary to group drugs to try to obtain a more relevant model.

Code to separate the drugs into many grups, we have many possibilities depanting on the correlation rate

```
drug_columns = ["Alcohol", "Amphet", "Amyl", "Benzos", "Cannabis", "Coke", "Crack", "Ecstasy", "Heroin", "Ketamine", "Legalh", "LSD

selected_df = df[drug_columns]

# Initialize correlation matrix
correlation_matrix = selected_df.corr()

# Find groups of drugs with strong correlations
correlation_groups = []

# we aim to get all the correlation_groups over 40% correlation

threshold = 0.8
while threshold > 0.4 :
    correlated_groups = []
    for drug in drug_columns :
        correlated_drugs = correlation_matrix[drug_columns][drug].abs() > threshold
        # get the list of the correlated drugs with "drug"
        correlated_group = list(correlation_matrix[drug_columns][correlated_drugs].index)
        if len(correlated_group) >= 2 and len(correlated_group)< 4 and correlated_group not in correlated_groups:
            correlated_groups.append(correlated_group)

    print("Groups with a correlation over: {:.2f}".format(threshold))
    for group in correlated_groups :
        print(group)
    print()

    threshold -= 0.1
```

```
Groups with a correlation over: 0.80

Groups with a correlation over: 0.70

Groups with a correlation over: 0.60
['Coke', 'Ecstasy']
['LSD', 'Mushrooms']

Groups with a correlation over: 0.50
['Amphet', 'Benzos', 'Meth']
['Amphet', 'Coke', 'Ecstasy']
['Crack', 'Heroin']
['Ecstasy', 'Ketamine']
['Benzos', 'Meth']
```

We have also categorized the drugs in the grups

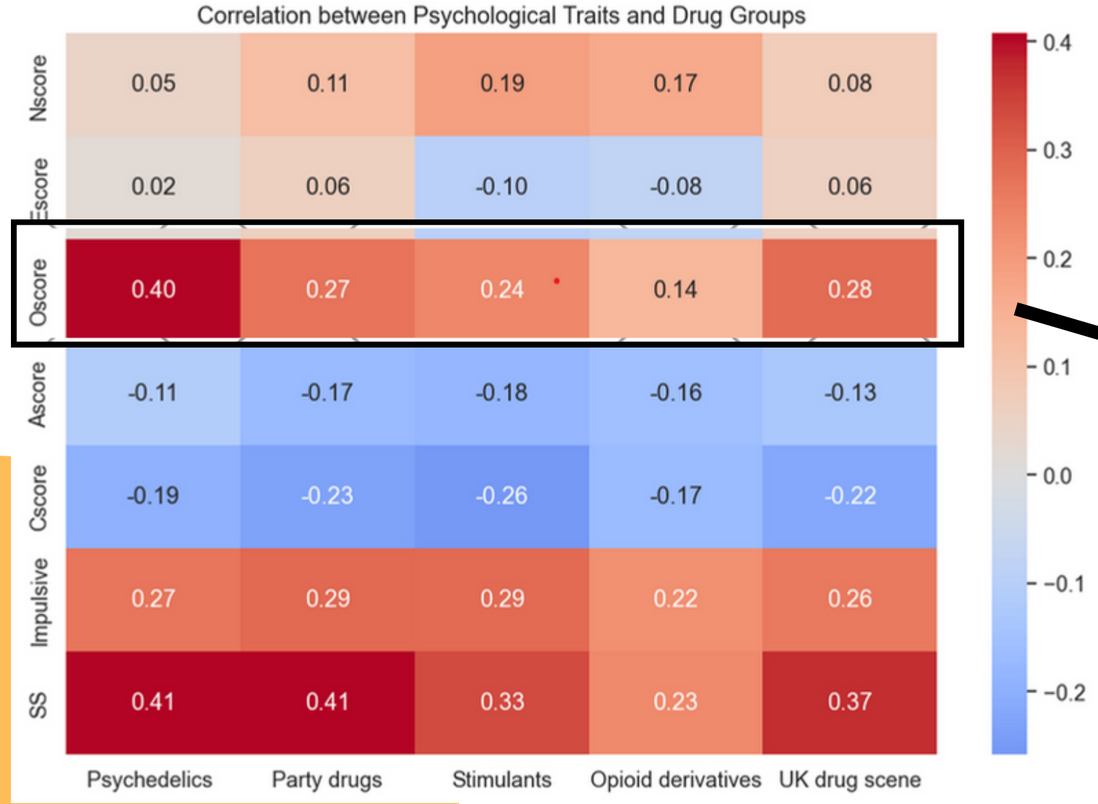
psychedelics : LSD and mushrooms

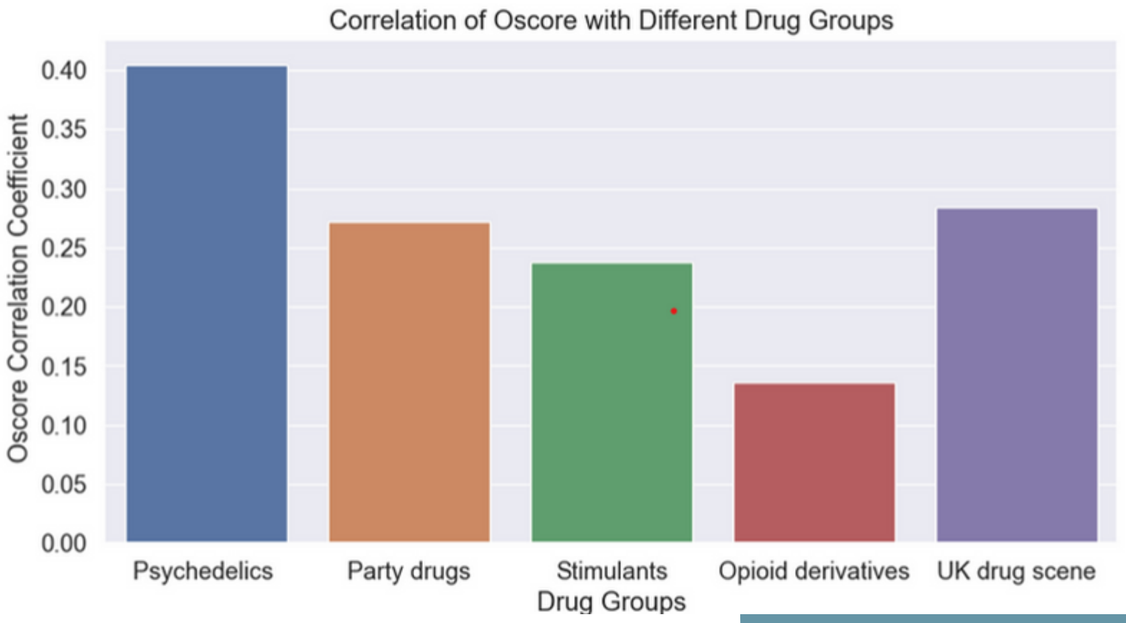
party_drugs = Coke and Ecstasy

stimulants = Amphetamines and Meth

opioids = Crack and Heroin

UK_drug_scene= Ecstasy, Ketamine





IV. Final Modelisation

Into our model we will consider that a drug consumer is "predisposed" to consume a type (in our code its a group) of drug, and the type of drug is what we are trying to predict

A user is a consumer of a type of drug if :
*he has used a drug of the group type in the last week
or
*he used both drugs in the last month

def predispose_to_group(drug_type_columns)

Example of the dataset:

	Nscore	Escore	Oscore	Ascore	Cscore	Impulsive	SS	drug_user
0	0.31287	-0.57545	-0.58331	-0.91699	-0.00665	-0.21712	-1.18084	0
1	-0.67825	1.93886	1.43533	0.76096	-0.14277	-0.71126	-0.21575	0
2	-0.46725	0.80523	-0.84732	-1.62090	-1.01450	-1.37983	0.40148	0
3	-0.14882	-0.80615	-0.01928	0.59042	0.58489	-1.37983	-1.18084	0
4	0.73545	-1.63340	-0.45174	-0.30172	1.30612	-0.21712	-0.21575	0
...	...	...	...	...	...	...	...	...
1880	-1.19430	1.74091	1.88511	0.76096	-1.13788	0.88113	1.92173	0
1881	-0.24649	1.74091	0.58331	0.76096	-1.51840	0.88113	0.76540	1
1882	1.13281	-1.37639	-1.27553	-1.77200	-1.38502	0.52975	-0.52593	0

Let’s do a grid search of a Knn model to find directly the parameters

Grid Search Function :

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import GridSearchCV
import numpy as np


def grid_search_knn(drug_group) :
    # Separate the features (X) and the target variable (y)
    X = drug_group.drop('drug_user', axis=1)
    y = drug_group['drug_user']
    # Define the parameter grid
    params = {
        "n_neighbors": np.arange(1, 20),
        "metric": ["euclidean", "manhattan","cityblock"],
        "weights": ["uniform", "distance"],
        "p": [1, 2]
    }

    # Create the KNN classifier
    knn = KNeighborsClassifier()

    # Create the GridSearchCV object
    grid = GridSearchCV(estimator=knn, param_grid=params)

    # Fit the grid search to the training data
    grid.fit(X, y)

    # Print the best score and best parameters
    print("Best Score:", grid.best_score_)
    print("Best Number of Neighbors:", grid.best_estimator_.n_neighbors)
    print("Best Metric:", grid.best_estimator_.metric)
    print("Best Weighting Scheme:", grid.best_estimator_.weights)
    print("Best p value:", grid.best_estimator_.p,"\n")
    return [grid.best_score_, grid.best_estimator_.n_neighbors, grid.best_estimator_.metric, grid.best_estimator_.weights, grid.be
```

```
Psychedelics  (LSD, Mushrooms)  :

Best Score: 0.9280765957446808
Best Number of Neighbors: 12
Best Metric: euclidean
Best Weighting Scheme: uniform
Best p value: 1

Party drugs (Coke, Ecstasy)  :

Best Score: 0.9131602836879432
Best Number of Neighbors: 12
Best Metric: euclidean
Best Weighting Scheme: uniform
Best p value: 1

Stimulants (Amphet, Meth)  :

Best Score: 0.8657432624113476
Best Number of Neighbors: 17
Best Metric: manhattan
Best Weighting Scheme: uniform
Best p value: 1

Opioid derivatives (Crack, Heroin)  :

Best Score: 0.9802893617021275
Best Number of Neighbors: 2
Best Metric: euclidean
Best Weighting Scheme: uniform
Best p value: 1

UK drug scene (Ecstasy, Ketamine)  :

Best Score: 0.9318070921985815
Best Number of Neighbors: 12
Best Metric: euclidean
Best Weighting Scheme: uniform
Best p value: 1
```

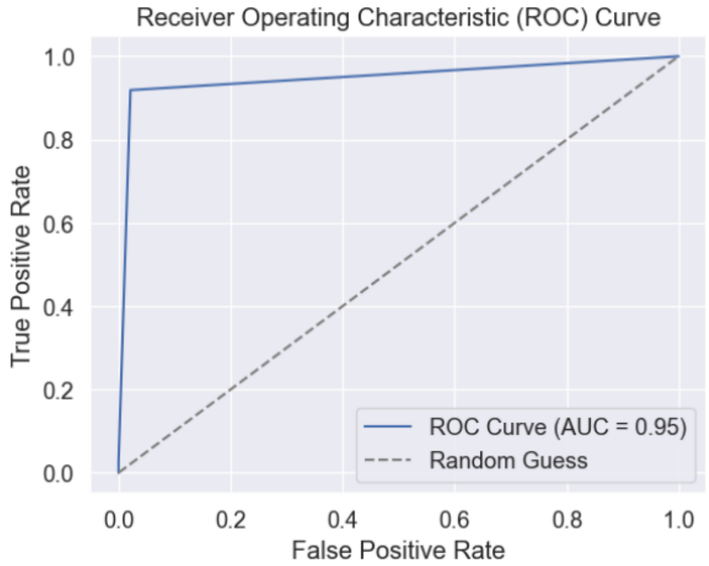
Lets find out how these values vary for our different drug groups

lets try our knn algorithm on the opiods derivatives (Crack, Heroin)

Accuracy on the test set: 0.9867021276595744

evaluate the performance of the KNN model using the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) metric.

We compute the False Positive Rate (FPR), True Positive Rate (TPR), and the corresponding classification thresholds using the roc_curve function. The FPR represents the proportion of falsely predicted negative samples, while the TPR represents the proportion of correctly predicted positive samples.
We calculate the AUC by calling the roc_auc_score function, which measures the overall performance of the model based on the ROC curve. The AUC value ranges between 0 and 1, with a higher value indicating better discrimination power.



Cross-Validation

(to evaluate the knn algo on opioids)

```
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(knn, X, y, cv=4)
print(accuracy)
print(f"Mean precision with cross validation : {accuracy.mean():.2f}")
```

```
[0.9787234  0.98081023 0.98081023 0.98081023]
Mean precision with cross validation : 0.98
```

INTERPRETATION : The ROC curve helps us understand the trade-off between true positive rate and false positive rate, while the AUC provides a single metric to assess the model's performance. here we have AUC = 0.95, which is really good !

Logistic rgression

First we will make logistic regression function to apply it to our different drug groups easily

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


def logistic_reg(drug_type) :
    X = drug_type.drop('drug_user', axis=1)
    y = drug_type['drug_user']

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42 )

    # Create a logistic regression model
    model = LogisticRegression()

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)

    print("Accuracy:", accuracy,"\n")
    return accuracy
```

Lets try to implement it with all our drug types, and see how it compares to the knn model

```
for drug_cols in all_drug_types :
    drug_list = ", ".join(all_drug_types[drug_cols])
    drug_group = predispose_to_group(all_drug_types[drug_cols])
    print(f"{drug_cols} ({drug_list})  :")
    drug_grid_knn = logistic_reg(drug_group)
```

```
Psychedelics  (LSD, Mushrooms)  :
Accuracy: 0.949468085106383

Party drugs (Coke, Ecstasy)  :
Accuracy: 0.9202127659574468

Stimulants (Amphet, Meth)  :
Accuracy: 0.875

Opioid derivatives (Crack, Heroin)  :
Accuracy: 0.9867021276595744

UK drug scene (Ecstasy, Ketamine)  :
Accuracy: 0.925531914893617
```

Overall, the logistic regression has better results than our knn algorithm, here are the results :

- 95 % accuracy -> logistic regression
- 93 % accuracy -> knn algorithm

Lets try to Tune the model using GridSearchCV

```
X = psy.drop('drug_user', axis=1)
y = psy['drug_user']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42 )

# Create a logistic regression model
psy_model = LogisticRegression()

param_grid = {
    'C': [0.1, 1, 10],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 200, 500]
}

grid_search = GridSearchCV(psy_model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print(best_params)
```

```
{'C': 0.1, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear'}
```

Lets apply the parameters above and see how it changes our logistic regression

```
psy_model = LogisticRegression(C=0.1,max_iter=100,penalty='l1',solver='liblinear')
psy_model.fit(X_train, y_train)

y_pred = psy_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)


print("Accuracy:", accuracy)
```

```
Accuracy: 0.9521276595744681
```

This time the results are better ! We are over 95%, and won 0.02 accuracy with this tuning

CONCLUSION

Jerome HENIN
Jacques HEGER
FX HERANDE

In conclusion we can predict with over 95% accuracy if someone is a big consumer of psychedelic drugs.
The best algorithm for our study is the logistic regression, with specific parameters.