DRUG CONSUMPTION CLASSIFICATION

ESILV - Python for data analysis - project 2023/2024

Dounia BOUGAMZA
Joalie CORNELIE
Margaux DELAFOSSE

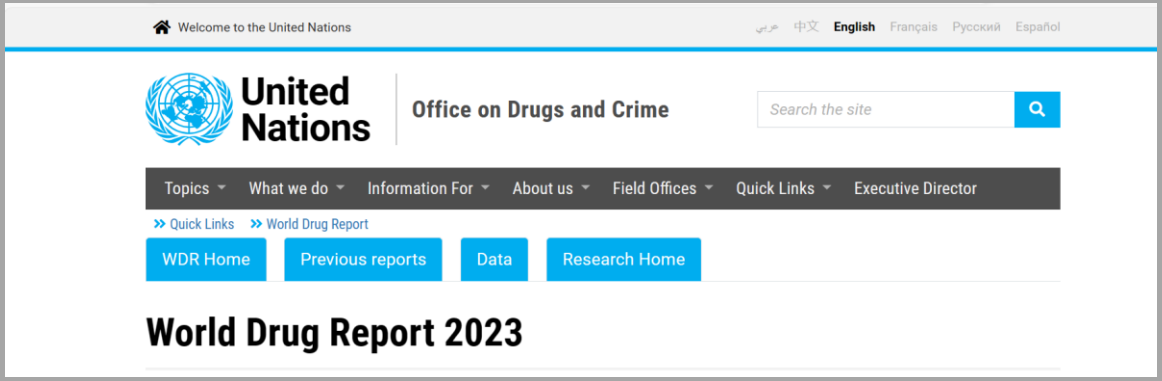
URL of the data set: [Drug consumption (quantified) - UCI Machine Learning Repository]

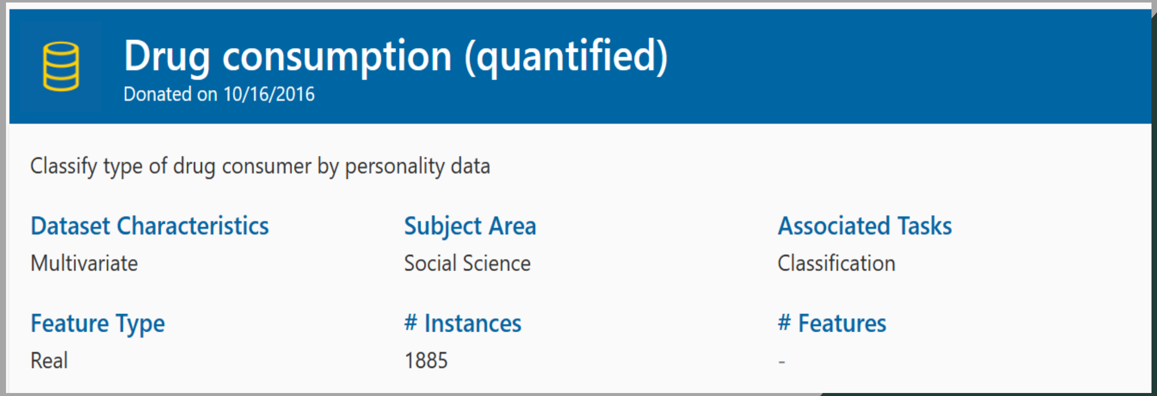
Contextualisation

- Actual Subject
- Other studies on the impacts of drug :

  -DRUG-RELATED CRIMES
  -DRUG RELATED MORBIDITY
  -CANNABIS REGULATION

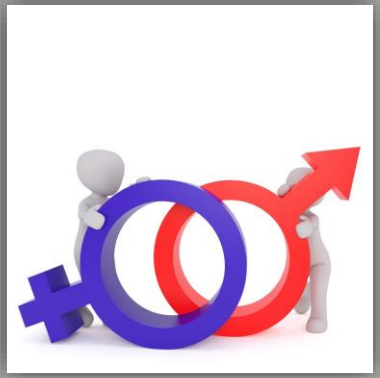
- Our dataset :
  -Personality analysis
  -Social analysis
  -Consumption frequency
  -Presence of legal drug (alcohol, nicotine)





Reflections and Perspectives

- Can a person's gender be predicted only through their legal drug consumption habits ?
- Can the gender of a person be predicted through all the parameters of the dataset ?
- Can drug use be predicted based on individual personality?



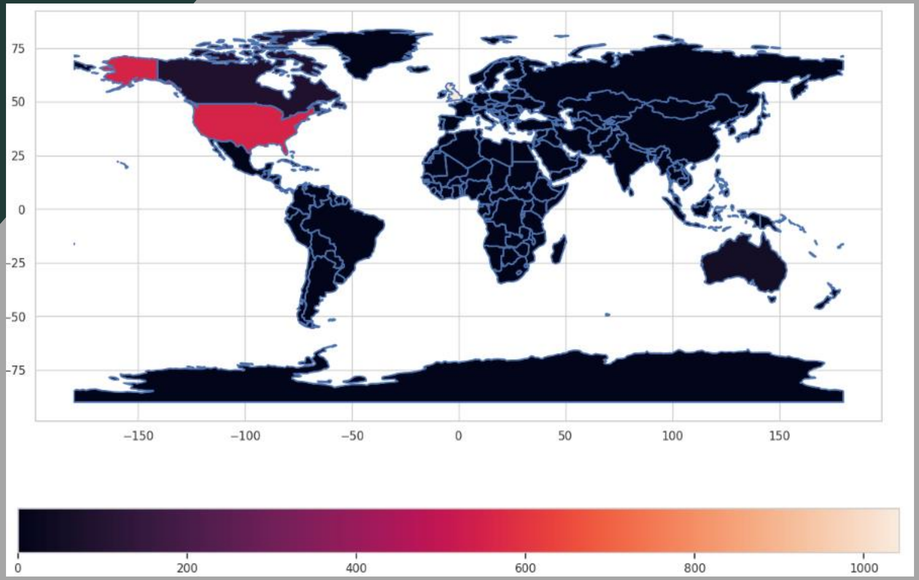
Our creation to solve the problem

1	0.49788	0.48246	-0.05921	0.96082	0.12600	0.31287	-0.57545	-0.58331	-0.91699	-0.00665	-0.21712
2	-0.07854	-0.48246	1.98437	0.96082	-0.31685	-0.67825	1.93886	1.43533	0.76096	-0.14277	-0.71126
3	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.46725	0.80523	-0.84732	-1.62090	-1.01450	-1.37983
4	-0.95197	0.48246	1.16365	0.96082	-0.31685	-0.14882	-0.80615	-0.01928	0.59042	0.58489	-1.37983
5	0.49788	0.48246	1.98437	0.96082	-0.31685	0.73545	-1.63340	-0.45174	-0.30172	1.30612	-0.21712
6	2.59171	0.48246	-1.22751	0.24923	-0.31685	-0.67825	-0.30033	-1.55521	2.03972	1.63088	-1.37983
...	...	...	...	...	...	...	...	...	...	...	...
884	-0.95197	0.48246	-0.61113	-0.57009	-0.31685	-1.19430	1.74091	1.88511	0.76096	-1.13788	0.88113
885	-0.95197	-0.48246	-0.61113	-0.57009	-0.31685	-0.24649	1.74091	0.58331	0.76096	-1.51840	0.88113
886	-0.07854	0.48246	0.45468	-0.57009	-0.31685	1.13281	-1.37639	-1.27553	-1.77200	-1.38502	0.52975
887	-0.95197	0.48246	-0.61113	-0.57009	-0.31685	0.91093	-1.92173	0.29338	-1.62090	-2.57309	1.29221
888	-0.95197	-0.48246	-0.61113	0.21128	-0.31685	-0.46725	2.12700	1.65653	1.11406	0.41594	0.88113

Id	Age	Gender	Education	Country	Ethnicity	Nscore	Escore	Oscore	Ascore	...	Crack	Ecstacy	Heroin
3	35-44	0	Professional certificate/ diploma	UK	White	-0.468323	0.808848	-0.849857	-1.625081	...	0	0	0
4	18-24	1	Masters degree	UK	White	-0.149326	-0.807991	-0.018369	0.592165	...	0	0	0
5	35-44	1	Doctorate degree	UK	White	0.736518	-1.638043	-0.452630	-0.302366	...	0	1	0
6	65+	1	Left school at 18 years	Canada	White	-0.679699	-0.300457	-1.560695	2.045349	...	0	0	0
7	45-54	0	Masters degree	USA	White	-0.468323	-1.094880	-0.452630	-0.302366	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
884	18-24	1	Some college or university, no certificate or ...	USA	White	-1.196668	1.747698	1.893952	0.763162	...	0	0	0

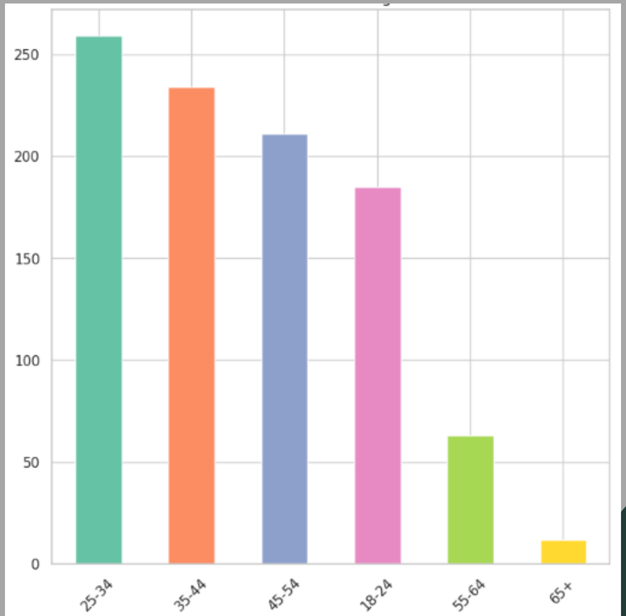
```
legal_drug = ['alcohol','caff','choc','legalh','nicotine']
illegal_drug = ['amphet', 'amyl', 'benzos', 'cannabis', 'coke',
                'crack','ecstacy', 'heroin', 'ketamine', 'lsd',
                'meth', 'mushrooms', 'vsa']
```

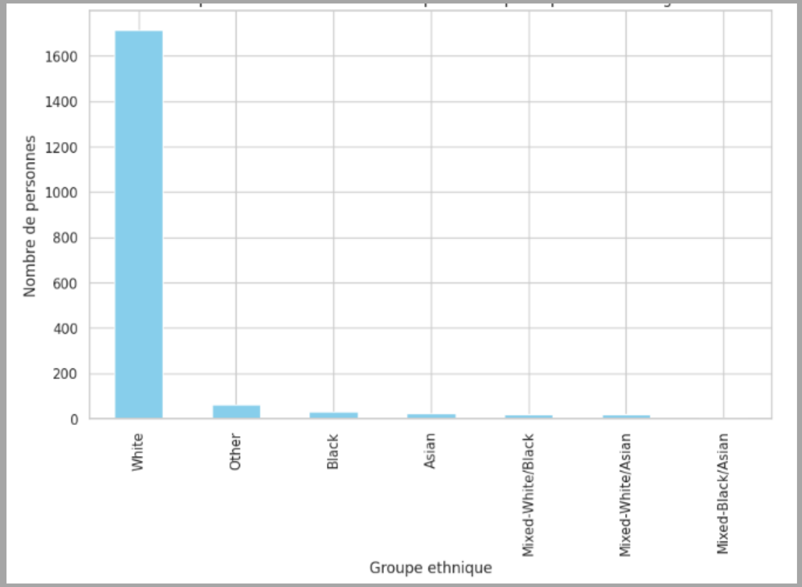
Analysis of survey participant



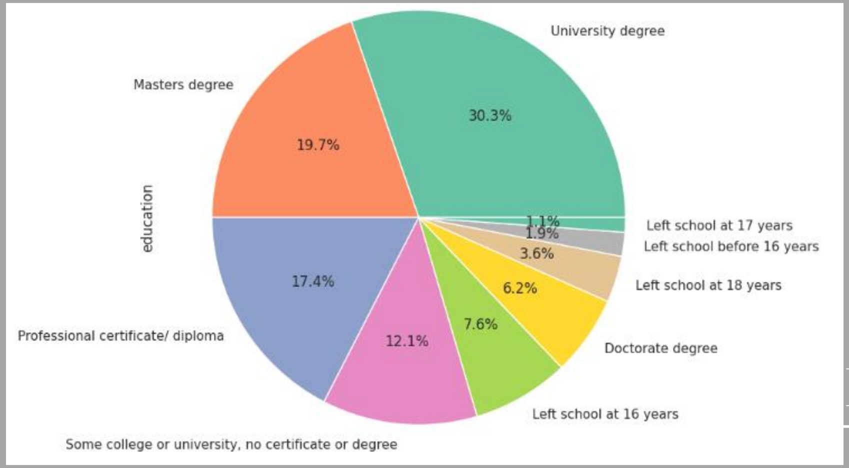
Number of participants per country

Distribution of age



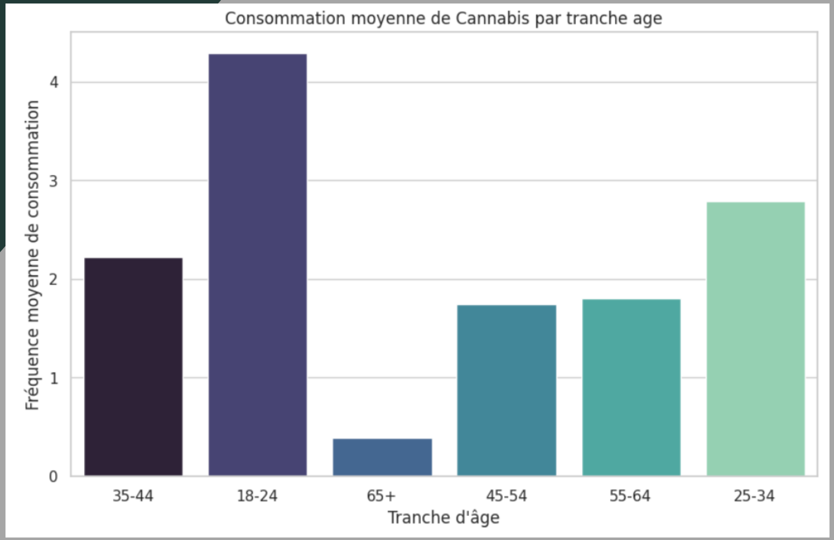


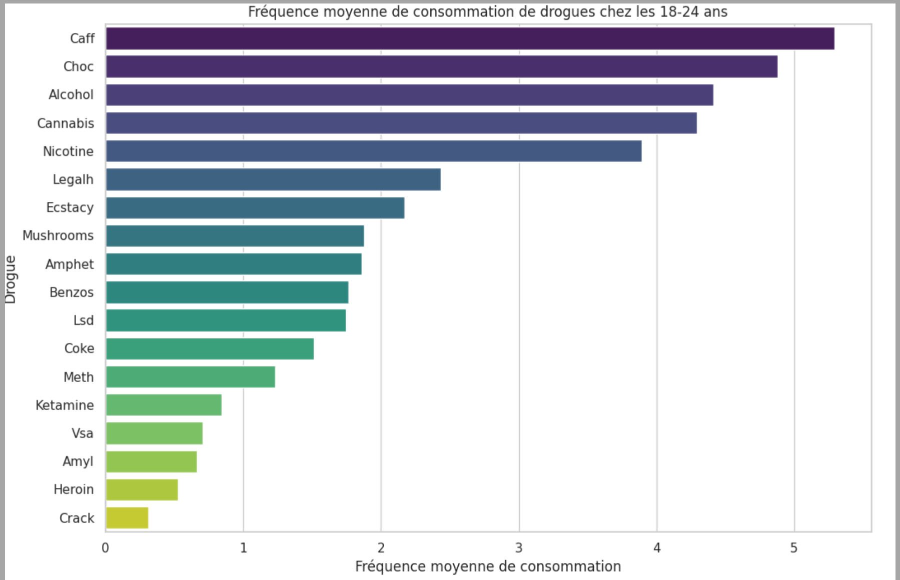
Distribution according to ethnic groups

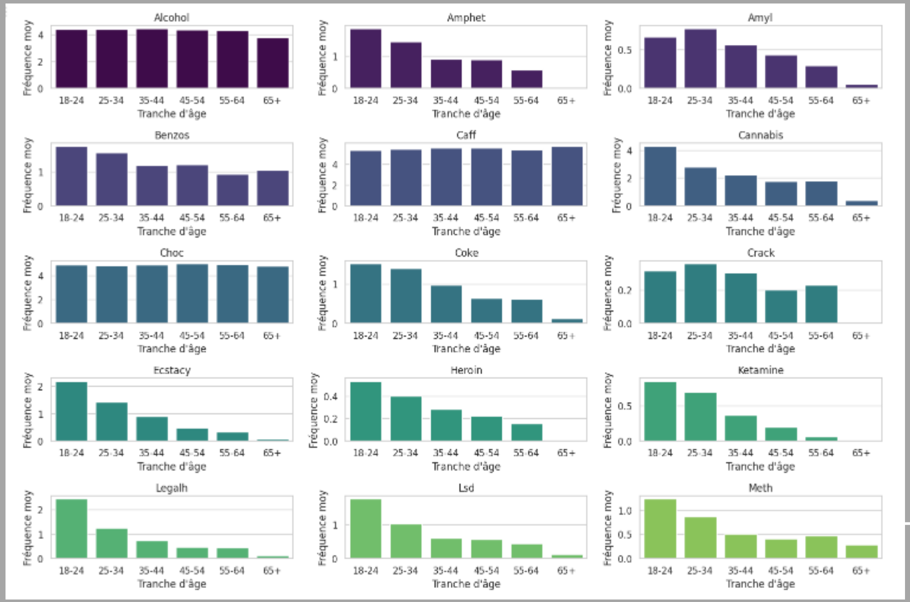


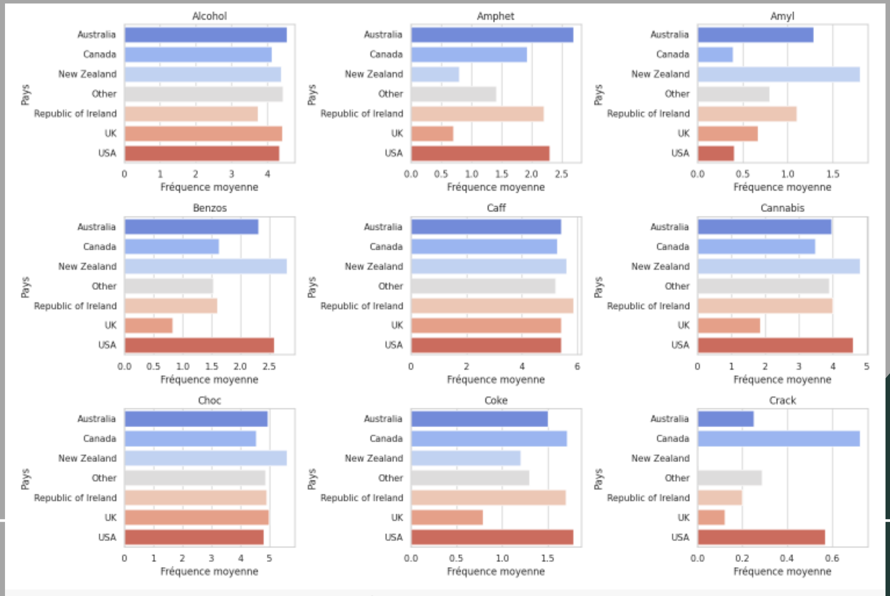
Distribution of education level

Visualization









Prediction of Gender based on legal drug consumption

1st Step : Selection of the variables

```
Entrée [202]: df_legal = df.loc[:, ['gender'] + list(df[legal_drug])]
              df_legal.head(5)
Out[202]:
```

	gender	alcohol	caff	choc	legalh	nicotine
0	Male	Used in Last Day	Used in Last Day	Used in Last Month	Never Used	Never Used
1	Female	Used in Last Month	Used in Last Week	Used in Last Month	Never Used	Used in Last Decade
2	Female	Used in Last Month	Used in Last Day	Used in Last Day	Used over a Decade Ago	Used in Last Decade
3	Female	Used in Last Decade	Used in Last Day	Used in Last Month	Never Used	Used in Last Day
4	Male	Used in Last Day	Used in Last Day	Used in Last Week	Never Used	Used in Last Day

```
Entrée [24]: X = df_legal[legal_drug]

             Y = df_legal['gender']
```

2nd Step : Encoding

```
Entrée [26]: from sklearn.preprocessing import OrdinalEncoder
             from sklearn.preprocessing import LabelEncoder
             from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

             Transformer_X=OrdinalEncoder()
             Transformer_Y = LabelEncoder()

Entrée [27]: X =Transformer_X.fit_transform(X)
             Y = Transformer_Y.fit_transform(Y)
```

3rd Step : Test of our models with a pipeline

- RandomForestClassifier
- DecisionTreeClassifier
- KNeighborsClassifier
- LogisticRegression

```
Entrée [215]: from sklearn.model_selection import cross_val_score

# Hyperparamètres pour chaque modèle
dtc_params = {'max_depth': 10, 'min_samples_split': 5}
knn_params = {'n_neighbors': 5}
rfc_params = {'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 5}
lr_params = {'C': 1}

pipelines = []

dtc_pipeline = Pipeline([('scaler', StandardScaler()), ('model', DecisionTreeClassifier(**dtc_params))])
knn_pipeline = Pipeline([('scaler', StandardScaler()), ('model', KNeighborsClassifier(**knn_params))])
rfc_pipeline = Pipeline([('scaler', StandardScaler()), ('model', RandomForestClassifier(**rfc_params))])
lr_pipeline = Pipeline([('scaler', StandardScaler()), ('model', LogisticRegression(**lr_params))])

pipelines.append(('Decision Tree Classifier', dtc_pipeline))
pipelines.append(('K-Nearest Neighbors Classifier', knn_pipeline))
pipelines.append(('Random Forest Classifier', rfc_pipeline))
pipelines.append(('Logistic Regression', lr_pipeline))

# Entrainement et évaluation de chaque modèle
for name, pipeline in pipelines:
    scores = cross_val_score(pipeline, X_train, Y_train, cv=10, scoring='accuracy')
    print(f"Modèle: {name}")
    print(f"Score de validation croisée moyen: {np.mean(scores)}\n")
```

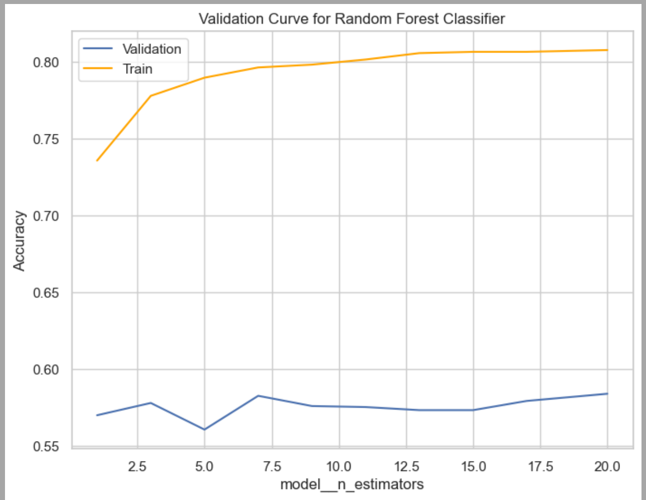
Results

GridSearch

```python
# Définir les grilles de paramètres pour chaque modèle
dtc_params = {'model__max_depth': [5, 10, 15, 20], 'model__min_samples_split': [2, 5, 10]}
knn_params = {'model__n_neighbors': [3, 5, 7, 9]}
rfc_params = {'model__n_estimators': [50, 100, 200], 'model__max_depth': [5, 10, 15, 20], 'model__min_
lr_params = {'model__C': [0.1, 1, 10]}
```

```python
# Boucle pour entraîner et évaluer chaque modèle avec GridSearchCV
for name, pipeline, param_grid in pipelines:
    grid = GridSearchCV(estimator=pipeline, param_grid=param_grid, cv=10, scoring='accuracy')
    grid.fit(X_train, Y_train)
    print(f"Modèle: {name}")
    print(f"Meilleurs hyperparamètres: {grid.best_params_}")
    print(f"Score de validation croisée: {grid.best_score_}\n")
```

Validation Curve for the Random Forest Classfier



Before the GridSearch

```
Modèle: Decision Tree Classifier
Score de validation croisée moyen: 0.5986666666666667

Modèle: K-Nearest Neighbors Classifier
Score de validation croisée moyen: 0.5986666666666667

Modèle: Random Forest Classifier
Score de validation croisée moyen: 0.6519999999999999

Modèle: Logistic Regression
Score de validation croisée moyen: 0.63
```

After the GridSearch

```
Modèle: Decision Tree Classifier
Meilleurs hyperparamètres: {'model__max_depth': 5, 'model__min_samples_split': 5}
Score de validation croisée: 0.6386666666666667

Modèle: K-Nearest Neighbors Classifier
Meilleurs hyperparamètres: {'model__n_neighbors': 7}
Score de validation croisée: 0.6080000000000001

Modèle: Random Forest Classifier
Meilleurs hyperparamètres: {'model__max_depth': 20, 'model__min_samples_split': 2, 'model__n_estimators': 200}
Score de validation croisée: 0.662

Modèle: Logistic Regression
Meilleurs hyperparamètres: {'model__C': 0.1}
Score de validation croisée: 0.6306666666666667
```

Prediction of gender based on all parameters

```python
from sklearn.model_selection import train_test_split

# Séparation des données en ensembles d'entraînement et de test
X = data_drogue.drop('Gender', axis=1)
y = data_drogue['Gender']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
from sklearn.ensemble import RandomForestClassifier

# Initialisation du modèle RandomForestClassifier
model = RandomForestClassifier()
```

```python
from sklearn.model_selection import GridSearchCV

# Hyperparameter tuning avec GridSearchCV
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10]}
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

```python
# Meilleurs paramètres
best_params = grid_search.best_params_
print(f"Meilleurs paramètres: {best_params}")
```

```
Meilleurs paramètres: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 50}
```

```python
# Prédiction sur l'ensemble de test
y_pred = grid_search.predict(X_test)
print(y_pred)
```

```
[0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1
 1 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0
 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 0 0
 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0
 1 1 1 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 0 0 0
 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1
 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0
 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0
 1 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0
 1 1 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1
 0 0 1 1 0 1 0]
```

```python
from sklearn.metrics import accuracy_score, confusion_matrix

# Évaluation du modèle
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy}")  #100% = good !
print(f"Matrice de Confusion:\n{conf_matrix}")
```

```
Accuracy: 1.0
Matrice de Confusion:
[[187   0]
 [  0 190]]
```

Random Forest

Personality analysis

- Splitting the users in 3 categories: Non Users, Users and Active Users

```
user_mapping={
    'Never Used':'Non User',
    'Used over a Decade Ago': 'Non User',
    'Used in Last Decade' : 'User',
    'Used in Last Year' : 'Active User',
    'Used in Last Month':'Active User',
    'Used in Last Week':'Active User',
    'Used in Last Day' : 'Active User'
}
```

Mapping users

- Setting up the users in a binary way: 0 for users (used in the last decade) and 1 for active users(used in last week, used in last month, used in last day)

```
drugs_mapping={
        'User':0,
        'Active User':1,
}
```

Visualization example

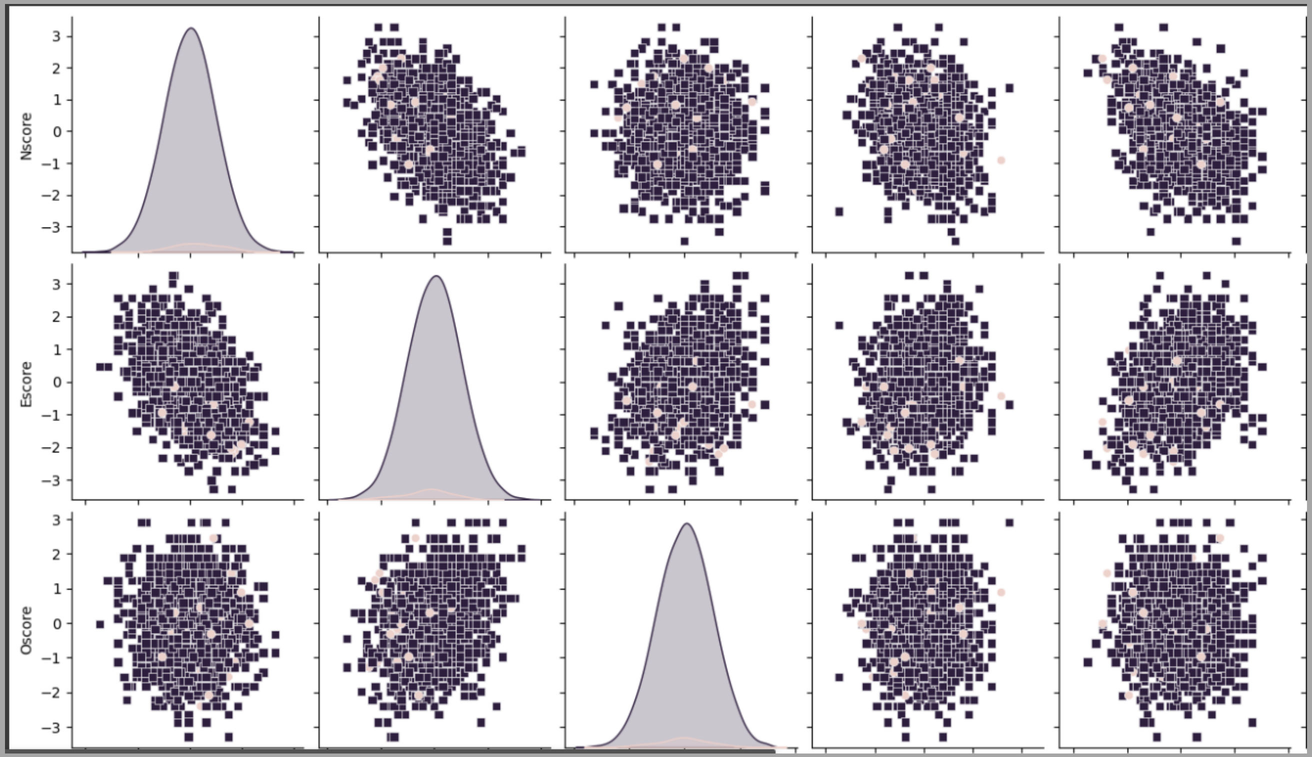
- Creating a dataset with the drug and the personnality traits

```
[142] df_alcohol= df_user_cat[df_user_cat['Alcohol'] != 'Never Used']
      df_alcohol=df_alcohol[df_alcohol['Age'] != '65+']
```

```
df_alcohol=df_alcohol[['Nscore','Escore','Oscore','Ascore','Cscore','Impulsiveness','SS','Alcohol']]
```

```
[144] df_alcohol['Alcohol']=df_alcohol.Alcohol.map(drugs_mapping)
```

Pairplot

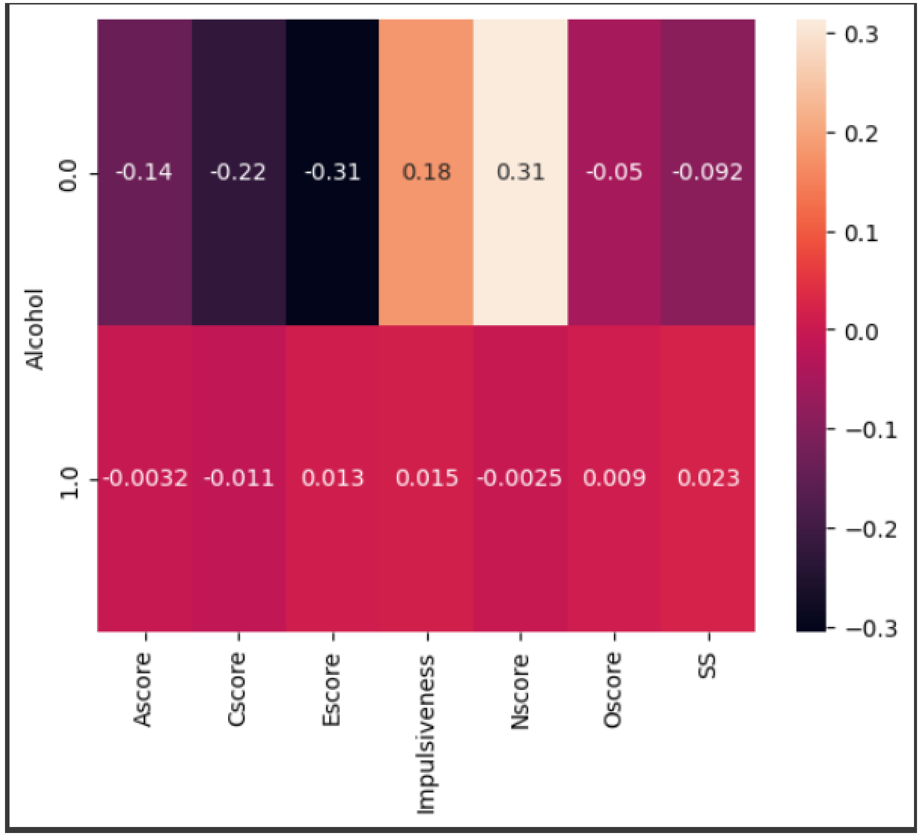


Pivoting tables

```
df_a= pd.pivot_table(df_alcohol,index='Alcohol',values=['Nscore','Escore','Oscore','Ascore','Cscore','Impulsiveness','SS'],aggfunc='mean')
```

	Ascore	Cscore	Escore	Impulsiveness	Nscore	Oscore	SS
Alcohol							
0.0	-0.138461	-0.223304	-0.305250	0.181701	0.312813	-0.050159	-0.092225
1.0	-0.003218	-0.010854	0.013439	0.015036	-0.002541	0.008997	0.022559

HEATMAP



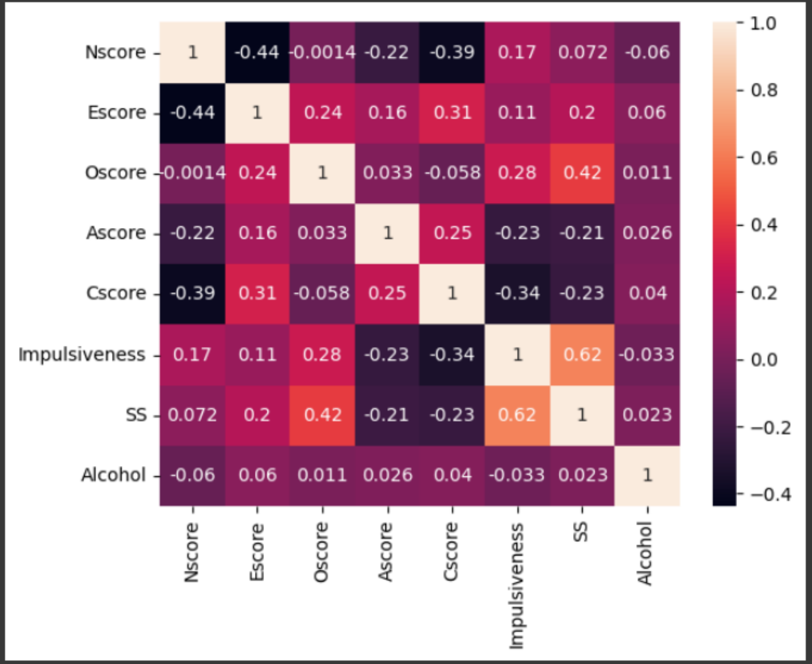
Alcohol
0.0
1.0
-0.14
-0.22
-0.31
0.18
0.31
-0.05
-0.092
-0.0032
-0.011
0.013
0.015
-0.0025
0.009
0.023
Ascore
Cscore
Escore
Impulsiveness
Nscore
Oscore
SS
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3

CORRELATION HEATMAP

```
[149] df_a_corr=df_alcohol.corr()
```

```
sns.heatmap(df_a_corr,annot=True)
```

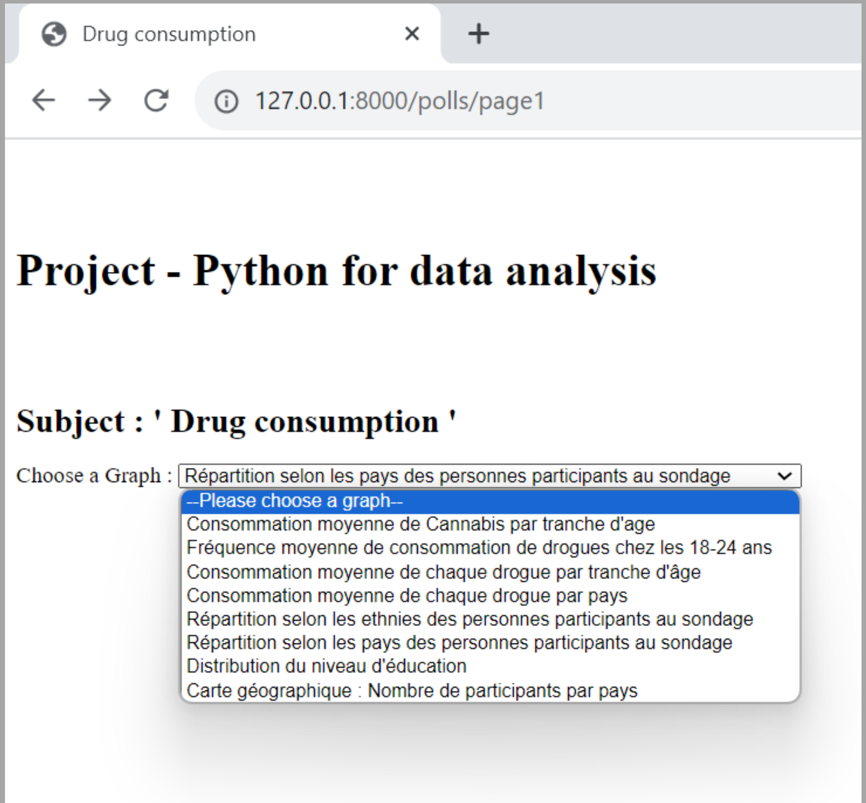
```
<Axes: >
```

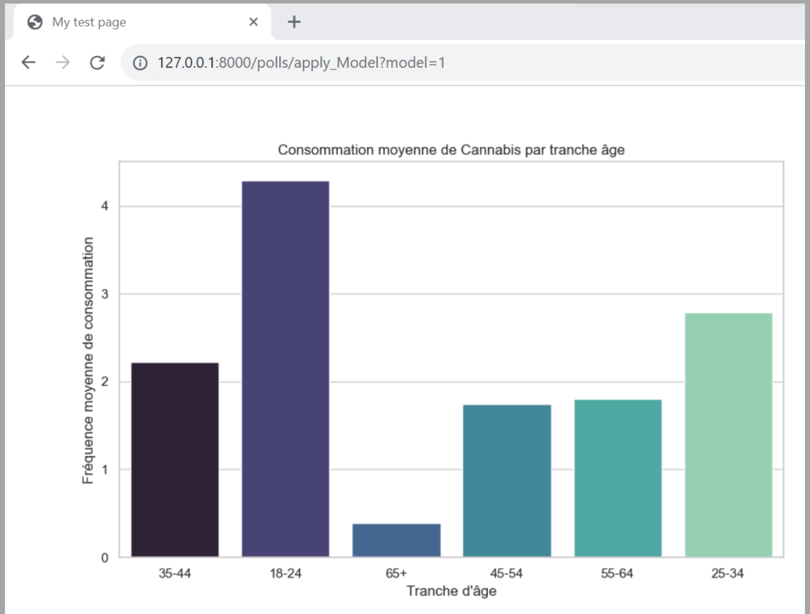


Generalization

```
for drug in drugs :
  print("Analyse de :", drug)
  #creating a dataset specific to the drug
  #Taking into account only people under 65
  df_drug= df_user_cat[df_user_cat[drug] != 'Never Used']
  df_drug=df_drug[df_drug['Age'] != '65+']
  df_drug=df_drug[['Nscore','Escore','Oscore','Ascore','Cscore','Impulsiveness','SS',drug]]
  df_drug[drug]=df_drug[drug].map(drugs_mapping)
  #pivoting the table to have the personnalities scores depending on the drug
  df_d= pd.pivot_table(df_drug,index=drug,values=['Nscore','Escore','Oscore','Ascore','Cscore','Impulsiveness','SS'],aggfunc='mean')
  #calculating the correlation in the drug dataset
  df_d_corr=df_drug.corr()
  f1=px.imshow(df_d_corr,text_auto=True)
  f1.show()
  f2=px.imshow(df_d,text_auto=True)
  f2.show()
  g=drug_pairplot=sns.pairplot(df_drug,hue=drug,markers=["o","s"])
  g.fig.suptitle(f"Pairplot of ${drug}")
  print(drug_pairplot)
  print("\n")
```

Django


Drug consumption
127.0.0.1:8000/polls/page1
Project - Python for data analysis
Subject : ' Drug consumption '
Choose a Graph : Répartition selon les pays des personnes participants au sondage
--Please choose a graph--
Consommation moyenne de Cannabis par tranche d'age
Fréquence moyenne de consommation de drogues chez les 18-24 ans
Consommation moyenne de chaque drogue par tranche d'âge
Consommation moyenne de chaque drogue par pays
Répartition selon les ethnies des personnes participants au sondage
Répartition selon les pays des personnes participants au sondage
Distribution du niveau d'éducation
Carte géographique : Nombre de participants par pays


My test page
127.0.0.1:8000/polls/apply_Model?model=1
Consommation moyenne de Cannabis par tranche âge
Fréquence moyenne de consommation
Tranche d'âge
35-44
18-24
65+
45-54
55-64
25-34


My test page
127.0.0.1:8000/polls/apply_Model?model=4
Benzos
Caff
Cannabis
Choc
Coke
Crack
Australia
Canada
New Zealand
Other
Republic of Ireland
UK
USA
Pays
Fréquence moyenne

Conclusion

- We can’t have an excellent prediction model for the gender of a person with only the consumption parameters.
- The gender of a person can be well predicted with a randomForestClassifier if we take into account all the parameters of the study
- The probability of a person to be a drug user is not determined by their personnality. We have to take into account that on top of personnality traits, a person's surroundings and environment plays a large role in their decision. With more data on the user's background we could make the models more accurate.