PREPARED BY
EHCHCHERQAOUI OUSSAMA
FEDDOUL YASSINE

ENCADRE PAR
EL OSMANI MUSTAPHA

DATA-SCIENCE PROJECT

INCOME CLASSIFICATION

PRETRAITEMENT DES DONNEES

Table of Contents

A. Description et but de la base de données

1. Recherche de la Base de Données

Avant le prétraitement des données on devait choisir une base de données adéquate sur laquelle on pouvait réaliser des opérations de cleaning.

Parmi les critères qu'on a pris en considération :

1-Doit avoir des données catégoriques et numériques.

2- La Présence des données manquantes dans notre base choisit

3- Un Minimum nombre d'instances pour construire un bon modèle.

2. LE BUSINESS DU DATA ET SON CONTENU

Le data-set choisit a fourni des caractéristiques prédictives comme l'éducation, le statut d'emploi, l'état civil pour prédire si le salaire est supérieur à 50 000 $.

Elle peut être utilisé pour pratiquer des problèmes d'apprentissage automatique comme la classification, on peut l'utiliser pour prédire si un

La data choisi contient

Nombre d'instances : 43957

Nombre d'attribut : 15 dont le dernier est le Target [Income >50k].

Dans le tableau suivant vous trouverez l'ensemble des attributs et leurs types ainsi que le contenu des attributs catégoriques.

ATTRIBUT	DESCRIPTION
Age	Type : Numerique Age of the person
workclass	Type: Categorique Categorical variable indicating the type of work workclass ['Private' 'State-gov' 'Self-emp-not-inc' 'Federal-gov' 'Local-gov' 'Self-emp-inc' 'Without-pay']
Fnlwgt	Type: Numerique Final weight
education	Type: Categorique education ['Doctorate' '12th' 'Bachelors' '7th-8th' 'Some-college' 'HS-grad' '9th' '10th' '11th' 'Masters' 'Preschool' '5th-6th' 'Prof-school' 'Assoc-voc' 'Assoc-acdm' '1st-4th']
educational-num	Type: Numerique Education as Integer
marital-status	Type: Categorique marital-status ['Divorced' 'Never-married' 'Married-civ-spouse' 'Widowed' 'Separated' 'Married-spouse-absent' 'Married-AF-spouse']
occupation	Type: Categorique occupation ['Exec-managerial' 'Other-service' 'Transport-moving' 'Adm-clerical' 'Machine-op-inspct' 'Sales' 'Handlers-cleaners' 'Farming-fishing' 'Protective-serv' 'Prof-specialty' 'Craft-repair' 'Tech-support' 'Priv-house-serv' 'Armed-Forces']
relationship	Type: Categorique relationship ['Not-in-family' 'Own-child' 'Husband' 'Wife' 'Unmarried' 'Other-relative']
race	Type: Categorique race ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']
gender	Type: Categorique gender ['Male' 'Female']
capital-gain	Type: Numerique
capital-loss	Type: Numerique
hours-per-week	Type: Numerique
native-country	Type: Categorique native-country ['United-States' 'Japan' 'South' 'Portugal' 'Italy' 'Mexico' 'Ecuador' 'England' 'Philippines' 'China' 'Germany' 'Dominican-Republic' 'Jamaica' 'Vietnam' 'Thailand' 'Puerto-Rico' 'Cuba' 'India' 'Cambodia' 'Yugoslavia' 'Iran' 'El-Salvador' 'Poland' 'Greece' 'Ireland' 'Canada' 'Guatemala' 'Scotland' 'Columbia' 'Outlying-US(Guam-USVI-etc)' 'Haiti' 'Peru' 'Nicaragua' 'Trinadad&Tobago' 'Laos' 'Taiwan' 'France' 'Hungary' 'Honduras' 'Hong' 'Holland-Netherlands']
Income >50k	Type: Numerique Target column

B. Data Preprocessing: Prétraitemet des données

1. Importation et visualisation des données :

On a importé les librairies de python NUMPY, PANDAS, MATPLOTLIB ET SEABORN. On a effectué la lecture de la base de donnée par pandas.read_csv.

Puis on afficher data par

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

In [2]: df = pd.read_csv("Downloads/archive (10)/train.csv")

In [3]: df
```

Out[3]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income_>50K
0	67	Private	366425	Doctorate	16	Divorced	Exec-managerial	Not-in-family	White	Male	99999	0	60	United-States	1
1	17	Private	244602	12th	8	Never-married	Other-service	Own-child	White	Male	0	0	15	United-States	0
2	31	Private	174201	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States	1
3	58	State-gov	110199	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	40	United-States	0
4	25	State-gov	149248	Some-college	10	Never-married	Other-service	Not-in-family	Black	Male	0	0	40	United-States	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
43952	52	Private	68982	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	50	United-States	1
43953	19	Private	116562	HS-grad	9	Never-married	Other-service	Own-child	White	Female	0	0	40	United-States	0
43954	30	Private	197947	Some-college	10	Divorced	Sales	Not-in-family	White	Male	0	0	58	United-States	0
43955	46	Private	97883	Bachelors	13	Never-married	Sales	Not-in-family	White	Female	0	0	35	United-States	0
43956	30	Private	375827	HS-grad	9	Never-married	Handlers-cleaners	Other-relative	White	Male	0	0	40	United-States	0

43957 rows × 15 columns

2. Visualisation de la répartition des types des attributs

```
In [78]: data.dtypes

Out[78]: age               int64
         workclass        object
         fnlwgt            int64
         education        object
         educational-num   int64
         marital-status   object
         occupation       object
         relationship     object
         race             object
         gender           object
         capital-gain      int64
         capital-loss      int64
         hours-per-week    int64
         native-country   object
         income_>50K       int64
         dtype: object
```

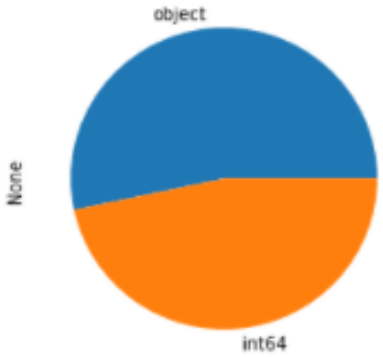
Nos attributs sont de types soit integer(numérique) et catégorique

```
In [79]: data.dtypes.value_counts()

Out[79]: object    8
         int64     7
         dtype: int64
```

```
data.dtypes.value_counts().plot.pie()

<AxesSubplot:ylabel='None'>
```



On a visualise les types des attributs de notre data

3. Détection et Suppression des lignes ou on détecte les valeurs manquantes

Dans cette partie on cherche à éliminer les lignes ou on a des valeurs manquantes .

On a commencé par une recherche des valeurs manquantes par attributs

```
In [82]: data.isnull().mean(axis=0).sort_values()*len(data)

Out[82]: age               0.0
         fnlwgt            0.0
         education         0.0
         educational-num   0.0
         marital-status    0.0
         relationship      0.0
         race              0.0
         gender            0.0
         capital-gain      0.0
         capital-loss      0.0
         hours-per-week    0.0
         income_>50K       0.0
         native-country    763.0
         workclass         2498.0
         occupation        2506.0
         dtype: float64
```

L'attribut 'native-contry' contient 763 valeur manquantes

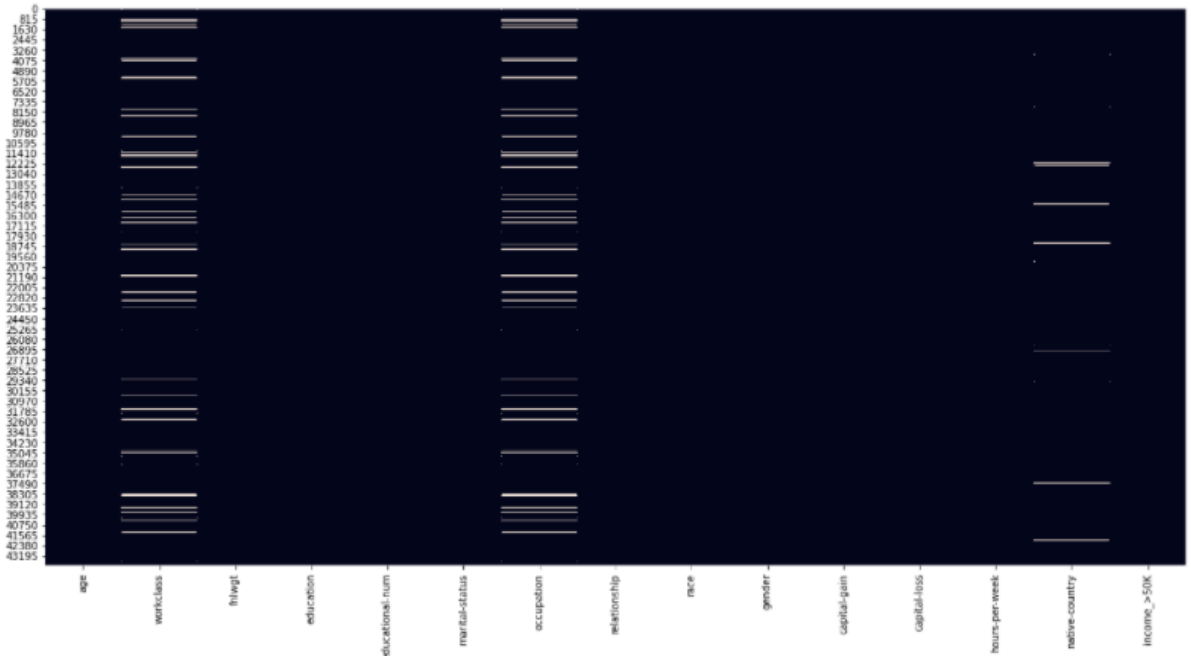
L'attribut 'workclass' contient 2498 valeur manquantes

L'attribut 'occupation' contient 2506 valeur manquantes

On remarque que les attributs *native-country* et *workclass*ainsi que*occupation* on des valeurs manquantes ,on va procéder à éliminer ces lignes

```
plt.figure(figsize=(20,10))
sns.heatmap(data.isna(), cbar=False)
```

```
<AxesSubplot:>
```



HEAT MAP : Visualisation des valeurs manquantes

```
data = data.dropna(subset=['native-country'])
```

On élimine les données manquantes de l'attribut 'native-country'

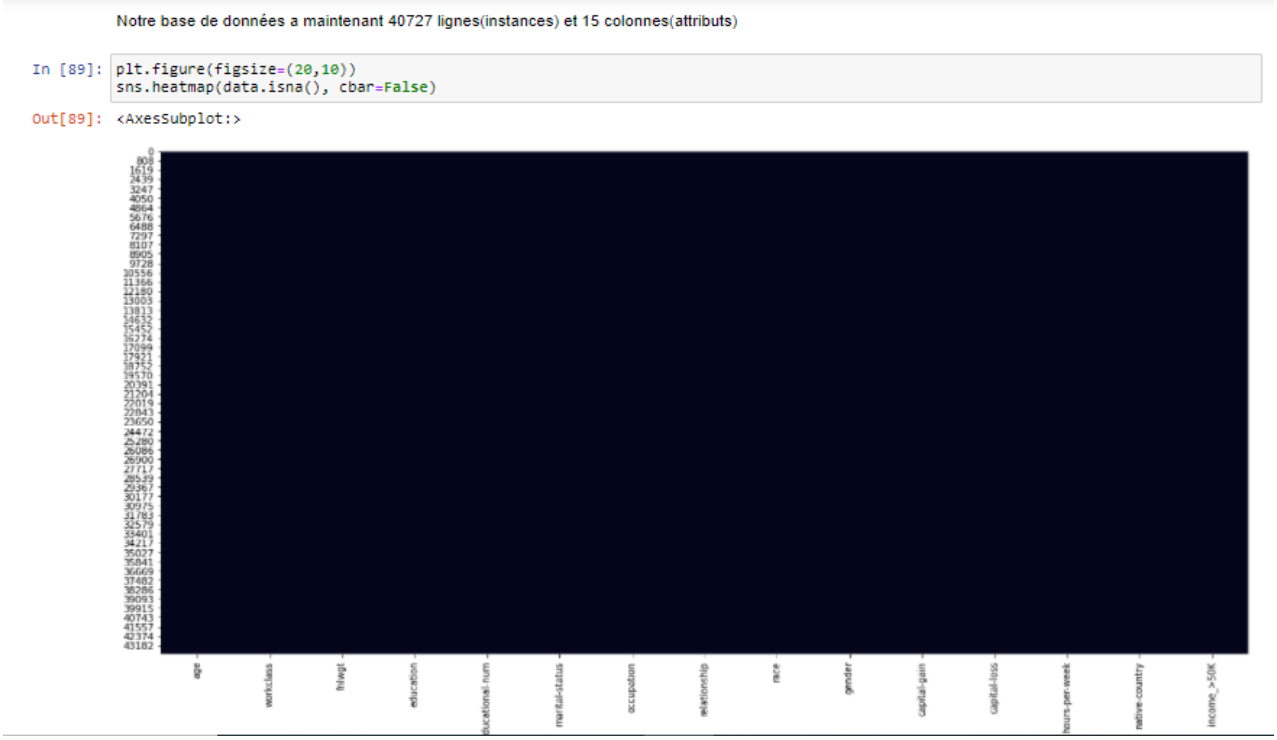
```
data = data.dropna(subset=['occupation'])
```

On élimine les données manquantes de l'attribut 'occupation'

```
data = data.dropna(subset=['workclass'])
```

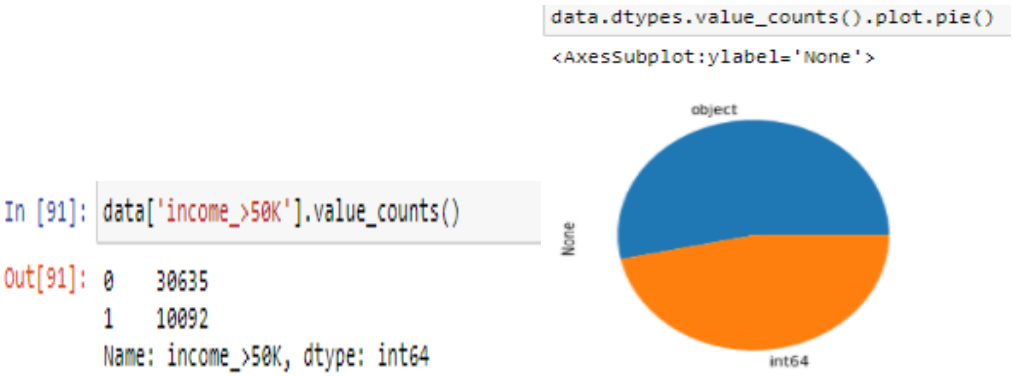
On élimine les données manquantes de l'attribut 'workclass'

On va faire une visualisation des valeurs après la suppression des lignes avec des valeurs manquantes



Notre base de données a maintenant 40727 lignes(instances) et 15 colonnes(attributs)

4. Analyse de fond : analyse de l'attribut Target



On a trouvé que :

On dispose de 30635 de personnes ayant un revenu inférieur à 50K 75,22%

On dispose de 10092 de personnes ayant un revenu supérieur à 50K 24,77%

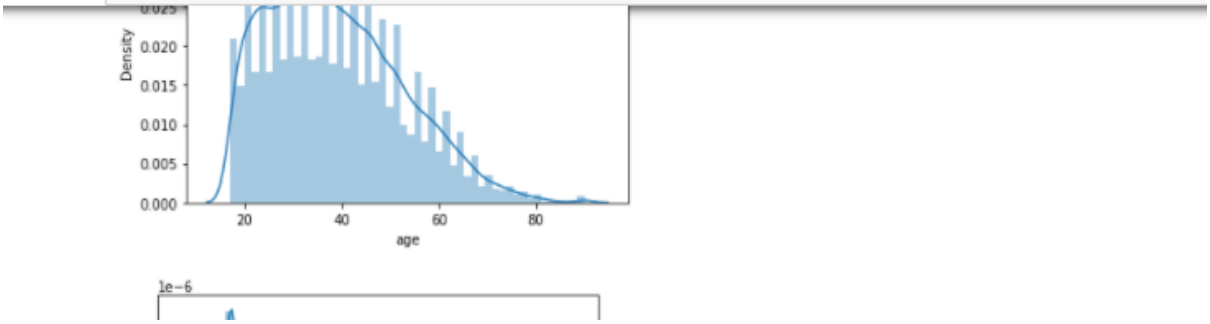
5. Analyse de fond : Histogramme des valeurs numériques

```
In [94]: for col in data.select_dtypes('int64'):
             print(col)

age
fnlwgt
educational-num
capital-gain
capital-loss
hours-per-week
income_>50K
```

L'ensemble des attributs ayant des données numérique

```
In [95]: for col in data.select_dtypes('int64'):
             plt.figure()
             sns.distplot(df[col])
```



On a utilisé : for col in data. Select_dtypes(‘int64’) car l’ensembles de tous les valeurs numeriques est de type « int 64 »

On a fait la visualisation des histogrammes on a pas trouvé un résonnement pour éliminer aucun attribut.

6. Analyse de fond pour les variables catégoriques

Variables Qualitatives

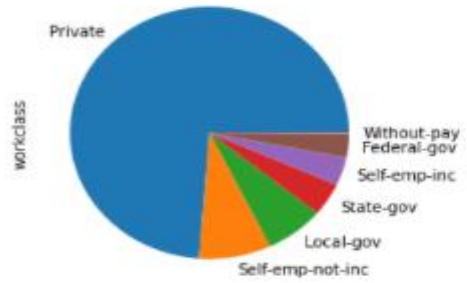
```
for col in data.select_dtypes('object'):
    print(col,data[col].unique())
```

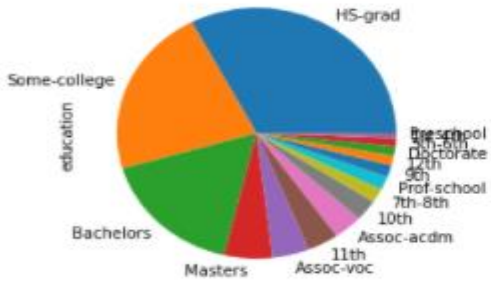
```
workclass ['Private' 'State-gov' 'Self-emp-not-inc' 'Federal-gov' 'Local-gov'
 'Self-emp-inc' 'Without-pay']
education ['Doctorate' '12th' 'Bachelors' '7th-8th' 'Some-college' 'HS-grad' '9th'
 '10th' '11th' 'Masters' 'Preschool' '5th-6th' 'Prof-school' 'Assoc-voc'
 'Assoc-acdm' '1st-4th']
marital-status ['Divorced' 'Never-married' 'Married-civ-spouse' 'Widowed' 'Separated'
 'Married-spouse-absent' 'Married-AF-spouse']
occupation ['Exec-managerial' 'Other-service' 'Transport-moving' 'Adm-clerical'
 'Machine-op-inspct' 'Sales' 'Handlers-cleaners' 'Farming-fishing'
 'Protective-serv' 'Prof-specialty' 'Craft-repair' 'Tech-support'
 'Priv-house-serv' 'Armed-Forces']
relationship ['Not-in-family' 'Own-child' 'Husband' 'Wife' 'Unmarried' 'Other-relative']
race ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']
gender ['Male' 'Female']
native-country ['United-States' 'Japan' 'South' 'Portugal' 'Italy' 'Mexico' 'Ecuador'
 'England' 'Philippines' 'China' 'Germany' 'Dominican-Republic' 'Jamaica'
 'Vietnam' 'Thailand' 'Puerto-Rico' 'Cuba' 'India' 'Cambodia' 'Yugoslavia'
 'Iran' 'El-Salvador' 'Poland' 'Greece' 'Ireland' 'Canada' 'Guatemala'
 'Scotland' 'Columbia' 'Outlying-US(Guam-USVI-etc)' 'Haiti' 'Peru'
 'Nicaragua' 'Trinadad&Tobago' 'Laos' 'Taiwan' 'France' 'Hungary'
 'Honduras' 'Hong' 'Holand-Netherlands']
```

On a fait la visualisation des attributs en utilisant pie diagramme

```
plt.figure(figsize=(30,15))
for col in data.select_dtypes('object'):
    plt.figure()
    data[col].value_counts().plot.pie()
```

```
<Figure size 2160x1080 with 0 Axes>
```





On a trouvé quelque chose d'intéressant :

Pour l'attribut '**native-country**'

```
In [98]: ((data['native-country'].value_counts())/len(data))*100
```

```
Out[98]: United-States                 91.261325
         Mexico                         2.057603
         Philippines                    0.640853
         Germany                        0.432146
         Puerto-Rico                    0.390404
         Canada                         0.341297
         El-Salvador                    0.336386
         India                          0.319199
         Cuba                           0.294645
         China                          0.260270
         England                        0.250448
         Jamaica                        0.230805
         South                          0.223439
         Dominican-Republic             0.223439
         Italy                          0.218528
         Japan                          0.198885
         Guatemala                      0.189064
         Vietnam                        0.184153
         Columbia                       0.176787
         Poland                         0.162055
         Haiti                          0.159599
         Portugal                       0.135046
         Iran                           0.122769
         Taiwan                         0.120313
         Nicaragua                      0.110492
         Greece                         0.108036
         Ecuador                        0.098215
         Peru                           0.095760
         Ireland                        0.076117
         France                         0.073661
         Thailand                       0.068750
         Hong                           0.066295
         Cambodia                       0.054018
         Trinadad&Tobago                0.051563
         Honduras                       0.046652
         Yugoslavia                     0.046652
         Scotland                       0.044197
```

On a remarqué qu'il y a un grand pourcentage de la population américaine.

Et nous sommes intéressés à faire une étude qu'on peut la généraliser sur le monde entier

DONC on a décidé d'éliminer l'attributs native country puisque nous ne sommes pas intéressés au origines des gens pour prédire leurs revenus.

```
In [99]: data = data.drop('native-country',axis=1,inplace=False)
```

7. Séparation des variables en fonction du résultat de la classification :

Création de sous-ensembles 1 et 0

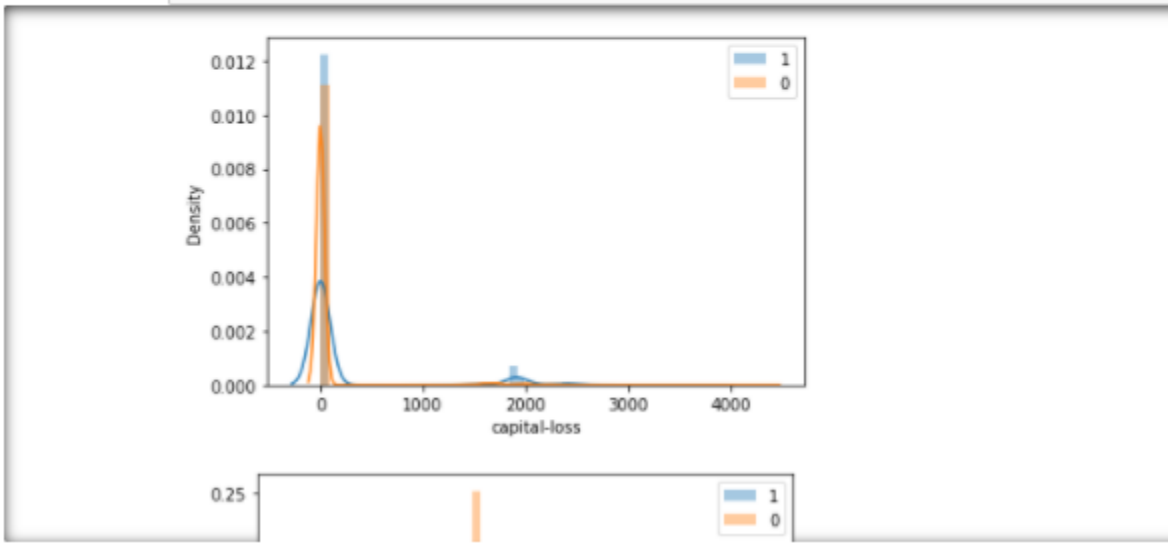
```
data1 = data[data['income_>50K'] == 1]
data1
```

```
In [102]: data0 = data[data['income_>50K'] == 0]
          data0
```

Après on a fait une visualisation

Target/variables

```
In [103]: for col in data.select_dtypes('int64'):
              plt.figure()
              sns.distplot(data1[col], label='1')
              sns.distplot(data0[col], label='0')
              plt.legend()
```



On a pu déduire que l'attribut **capital loss** et **capital gain** sont non significatifs donc on va les éliminer de notre data-set

Puisque la majorité de leurs contenus sont des zéros.

Pour capital gain 91.632087% sont des zéros.

Pour capital loss 91.632087% sont des zéros.

```
In [104]: ((data['capital-gain'].value_counts())/len(data))*100
```

```
Out[104]: 0        91.632087
          15024     1.075454
          7688      0.866747
          7298      0.775898
          99999     0.527905
                     ...
          2993      0.004911
          1639      0.002455
          7262      0.002455
          1731      0.002455
          22040     0.002455
          Name: capital-gain, Length: 120, dtype: float64
```

Cette colonne n'est signifiante vue que plus que 90% des valeurs ont une valeur égale à 0.

```
In [105]: data = data.drop('capital-gain',axis=1,inplace=False)
```

La même chose a été faite pour capital loss.

8. DIVISER LA DATA EN CATEGORICAL ET NUMERIQUES :

Diviser en variables catégoriques et numériques

```
In [114]: cat_data=[]
          num_data=[]
          for i, c in enumerate(data.dtypes):
              if c==object:
                  cat_data.append(data.iloc[:,i])
              else:
                  num_data.append(data.iloc[:,i])
          cat_data=pd.DataFrame(cat_data).transpose()
          num_data=pd.DataFrame(num_data).transpose()
```

```
In [115]: cat_data
```

Out[115]:

	workclass	education	marital-status	occupation	relationship	race	gender
0	Private	Doctorate	Divorced	Exec-managerial	Not-in-family	White	Male
1	Private	12th	Never-married	Other-service	Own-child	White	Male
2	Private	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male
3	State-gov	7th-8th	Married-civ-spouse	Transport-moving	Husband	White	Male
4	State-gov	Some-college	Never-married	Other-service	Not-in-family	Black	Male
...	...	...	...	...	...	...	...
43952	Private	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male
43953	Private	HS-grad	Never-married	Other-service	Own-child	White	Female
43954	Private	Some-college	Divorced	Sales	Not-in-family	White	Male
43955	Private	Bachelors	Never-married	Sales	Not-in-family	White	Female
43956	Private	HS-grad	Never-married	Handlers-cleaners	Other-relative	White	Male

40727 rows × 7 columns

9. Transformation des valeurs catégoriques en numériques et concaténation des deux sous-data

```
In [116]: for i in cat_data:
              cat_data[i]=le.fit_transform(cat_data[i])
          cat_data
```

Out[116]:

	workclass	education	marital-status	occupation	relationship	race	gender
0	2	10	0	3	1	4	1
1	2	2	4	7	3	4	1
2	2	9	2	3	0	4	1
3	5	5	2	13	0	4	1
4	5	15	4	7	1	2	1
...	...	...	...	...	...	...	...
43952	2	9	2	3	0	4	1
43953	2	11	4	7	3	4	0
43954	2	15	0	11	1	4	1
43955	2	9	4	11	1	4	0
43956	2	11	4	5	2	4	1

40727 rows × 7 columns

Apres la transformations des données catégoriques en numeriques nous allons les concaténer

```
In [117]: X=pd.concat([cat_data,num_data],axis=1)
```

```
In [118]: X
```

Out[118]:

	workclass	education	marital-status	occupation	relationship	race	gender	age	fnlwgt	educational-num	hours-per-week	income_>50K
0	2	10	0	3	1	4	1	67	366425	16	60	1
1	2	2	4	7	3	4	1	17	244602	8	15	0
2	2	9	2	3	0	4	1	31	174201	13	40	1
3	5	5	2	13	0	4	1	58	110199	4	40	0
4	5	15	4	7	1	2	1	25	149248	10	40	0
...	...	...	...	...	...	...	...	...	...	...	...	...
43952	2	9	2	3	0	4	1	52	68982	13	50	1
43953	2	11	4	7	3	4	0	19	116562	9	40	0
43954	2	15	0	11	1	4	1	30	197947	10	58	0
43955	2	9	4	11	1	4	0	46	97883	13	35	0
43956	2	11	4	5	2	4	1	30	375827	9	40	0

40727 rows × 12 columns

Maintenant nous avons une data prête a être utilise dans le training et la création du model

C. Analyse des donnees : Entrainment & prediction

1. Importation et séparation de la data en train et test :

Model

```
In [54]: from sklearn.model_selection import train_test_split

In [55]: X_train, X_test, y_train, y_test = train_test_split(I, Y, test_size=0.3, random_state=40)

In [56]: from sklearn.linear_model import SGDClassifier
         from sklearn.preprocessing import StandardScaler
         from sklearn.pipeline import make_pipeline
         from sklearn.model_selection import GridSearchCV

In [57]: scaler = StandardScaler()

In [58]: X_train = scaler.fit_transform(X_train)

In [59]: X_test = scaler.fit_transform(X_test)
```

On va utiliser la décision tree pour faire la classification puisque notre data est créer pour classifier les gens selon leurs revenus.

```
from sklearn.tree import DecisionTreeClassifier
```

```
pgrid={"splitter":["best","random"],
             "max_depth":range(2,20,1),
             "min_samples_leaf":range(1,15,1),
             "min_samples_split":range(2,20,1)
            }
```

```
grid_search = GridSearchCV(DecisionTreeClassifier(), param_grid=pgrid, cv=5)
grid_search.fit(X_train, y_train)
grid_search.best_estimator_.score(X_test, y_test)
```

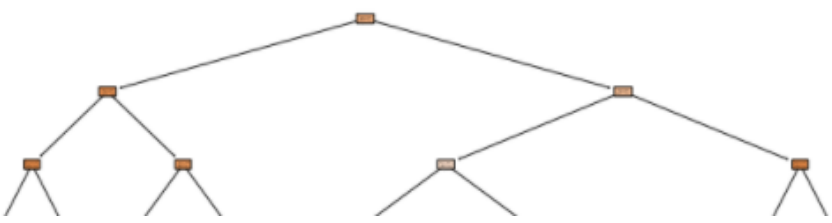
```
0.8194614943939766
```

```
grid_search.best_estimator_
```

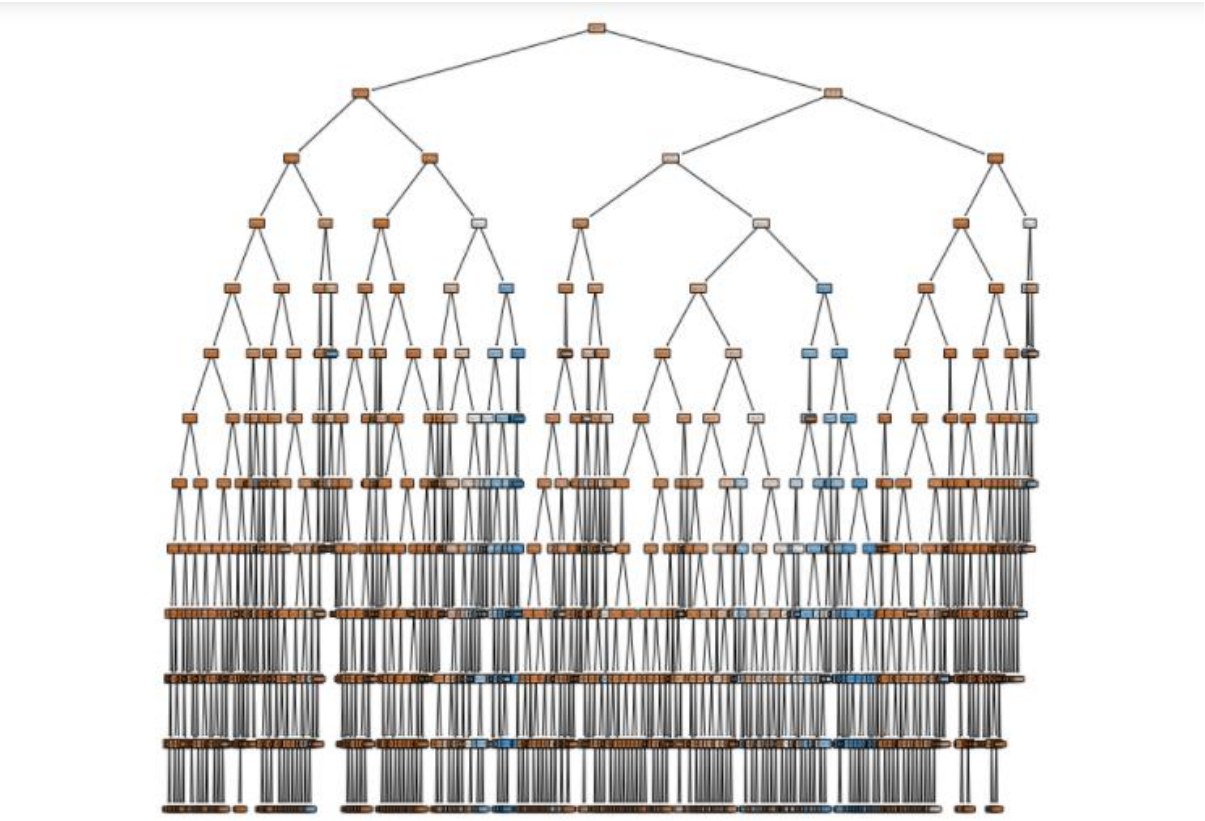
```
DecisionTreeClassifier(max_depth=12, min_samples_leaf=9, min_samples_split=16,
                       splitter='random')
```

```
from sklearn import tree
```

```
plt.figure(figsize=(12,12))
tree.plot_tree(grid_search.best_estimator_,rounded=True,filled=True)
plt.show()
```



Notre model



D. Conclusion

Dans de ce projet de prétraitement des données, on a travaillé sur une base de données qui consiste à classifier les gens selon leurs revenus >50 K ou pas.

Cette data-set nous permet de prédire selon le niveau universitaire, le genre, l'occupation et bien d'autres attributs si une personne gagne plus que 50 K ou non.

Premièrement nous avons nettoyer la data des valeurs aberrantes et des valeurs manquantes.

Puis on a appliqué différentes visualisations afin de bien comprendre les attributs et pouvoir trouver les relations entre eux afin de déduire leurs signifiances.

Tous cela nous a permet de créer une clean data, qu'on a utilisé après pour créer un modèle de classification en utilisant le modèle du décision tree pour prédire si une personne est dans la classe des gens qui gagnent plus que 50 K ou non.