Entrée [1]:

```
import pandas as pd
import numpy as np

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

from bokeh.plotting import figure, show, output_file
import seaborn as sns
```

Entrée [2]:

```
train = pd.read_csv('data/train.csv')
train = train.replace('\\N',np.NAN)
n,p = train.shape
train.head()
```

Out[2]:

	id	STRUCTURE PRESCRIPTRICE	PLATEFORME	year	month	region	ORIENTATION	NATI
0	1550	PRESCRIPTEUR35	bancaire	2015	7	BOURGOGNE FRANCHE COMTE	Surendettement	Suren
1	2249	PRESCRIPTEUR16	bancaire	2016	2	OCCITANIE	Mediation	En
2	2670	PRESCRIPTEUR18	social	2016	4	PROVENCE-ALPES-COTE-D'AZUR	Accompagnement	En
3	1102	PRESCRIPTEUR6	bancaire	2014	11	BRETAGNE	Mediation	Multien
4	7069	PRESCRIPTEUR23	social	2018	9	NOUVELLE AQUITAINE	Accompagnement	Dif

5 rows × 44 columns

I - Traitement des types

Entrée [3]:

```
train.columns.to_series().groupby(train.dtypes.astype(str)).groups['object']
```

Out[3]:

```
Index(['STRUCTURE PRESCRIPTRICE', 'PLATEFORME', 'region', 'ORIENTATIO
N',
       'NATURE_DIFF', 'age', 'tranche_age', 'situation', 'adulte_foye
r',
       'PROF', 'LOGEMENT', 'cat_rev', 'cat_charges', 'cat_credit',
       'IMPAYES_DEBUT', 'cat_impayes', 'cat_RAV_ouverture', 'RAV_UC',
       'cat_RAV_UC', 'CRD', 'gain_mediation', 'crd_amort', 'crd_renou
v',
       'crd_immo', 'crd_rac', 'crd_autres', 'crd_decouvert', 'moy_eco_
jour',
       'cat_moy_eco_jour'],
      dtype='object')
```

• Numeric

Entrée [4]:

```
for c in train.columns:
    if 'crd_' in c or c in ['CRD','IMPAYES_DEBUT','age','adulte_foyer']:
        train[c] = pd.to_numeric(train[c], errors='coerce')
```

• Float

Entrée [5]:

```
train.moy_eco_jour = train.moy_eco_jour.str.replace(',', '.', regex=False).astype(fl
train.RAV_UC = train.RAV_UC.str.replace(',', '.', regex=False).astype(float)
```

• Date

Entrée [6]:

```
train['Date'] = pd.to_datetime(train.year.astype(str) + '-' + train.month.astype(str
train = train.drop(columns=['year','month'])
```

II - Traitement des NA

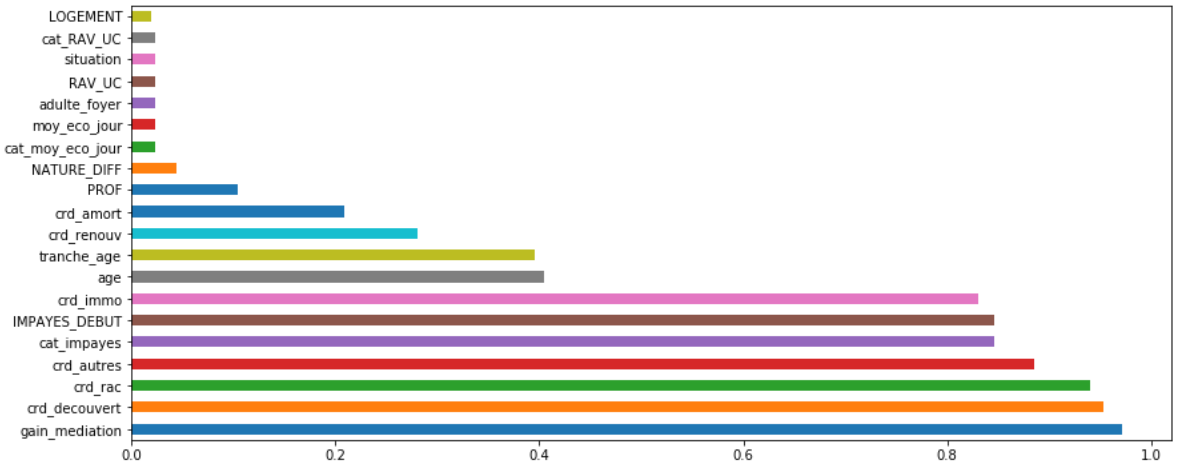
Entrée [7]:

```
train = train.replace('Non Renseigne', np.NAN)
```

Entrée [8]:

```
train_na = train.isna().sum()/train.shape[0]

plt.figure(figsize=(14,6))
train_na.sort_values(ascending=False)[:20].plot(kind='barh')
plt.show()
```



Certaines colonnes contiennent plus de **80%** de **NA** ce qui est beaucoup **trop important** pour appliquer une méthode de **fillna**.

• CRD et nb

Entrée [9]:

```
train[train.crd_decouvert.isna()][['crd_decouvert','nb_decouvert']].sum()
```

Out[9]:

```
crd_decouvert    0.0
nb_decouvert     0.0
dtype: float64
```

La colonne **crd_decouvert** contient des **NA** lorsque la colonne **nb_couvert**. Ce n'est donc pas un vrai manque de valeur. Nous pouvons remplacer ces **NA** par **0**.

Vérifions cette hypothèse pour les colonnes **CRD - nb**.

Entrée [10]:

```
for CRD in train.columns.tolist():
    if 'crd_' in CRD:
        _type = CRD.split('_')[1]
        print('\n')
        print(dict(train[train[CRD].isna()][[CRD,f'nb_{_type}']].sum()))
```

```
{'crd_amort': 0.0, 'nb_amort': 0.0}


{'crd_renouv': 0.0, 'nb_renouv': 0.0}


{'crd_immo': 0.0, 'nb_immo': 0.0}


{'crd_rac': 0.0, 'nb_rac': 0.0}


{'crd_autres': 0.0, 'nb_autres': 0.0}


{'crd_decouvert': 0.0, 'nb_decouvert': 0.0}
```

Entrée [11]:

```
for CRD in train.columns.tolist():
    if 'crd_' in CRD:
        train[CRD] = train[CRD].fillna(0)
train.head()
```

Out[11]:

	id	STRUCTURE PRESCRIPTRICE	PLATEFORME	region	ORIENTATION	NATURE_DIFF	age
0	1550	PRESCRIPTEUR35	bancaire	BOURGOGNE FRANCHE COMTE	Surendettement	Surendettement	NaN
1	2249	PRESCRIPTEUR16	bancaire	OCCITANIE	Mediation	Endettement	NaN
2	2670	PRESCRIPTEUR18	social	PROVENCE-ALPES-COTE-D'AZUR	Accompagnement	Endettement	NaN
3	1102	PRESCRIPTEUR6	bancaire	BRETAGNE	Mediation	Multiendettement	NaN
4	7069	PRESCRIPTEUR23	social	NOUVELLE AQUITAINE	Accompagnement	Difficultés de Gestion	24.0

5 rows × 43 columns

• Age

Entrée [12]:

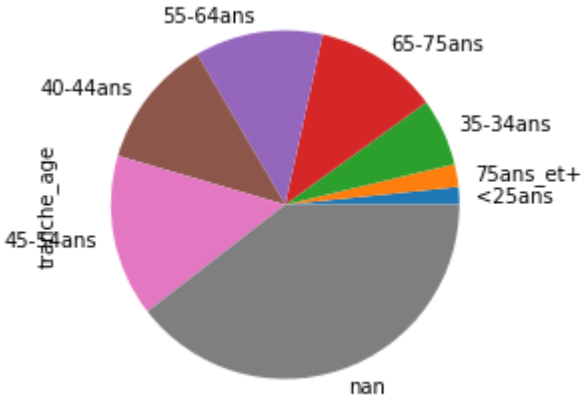
```
#train.tranche_age = train.tranche_age.replace('Non Renseigne', np.NAN)
```

Entrée [13]:

```
train.tranche_age.value_counts(ascending=True,dropna=False,normalize=True).plot(kind
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a217389e8>
```

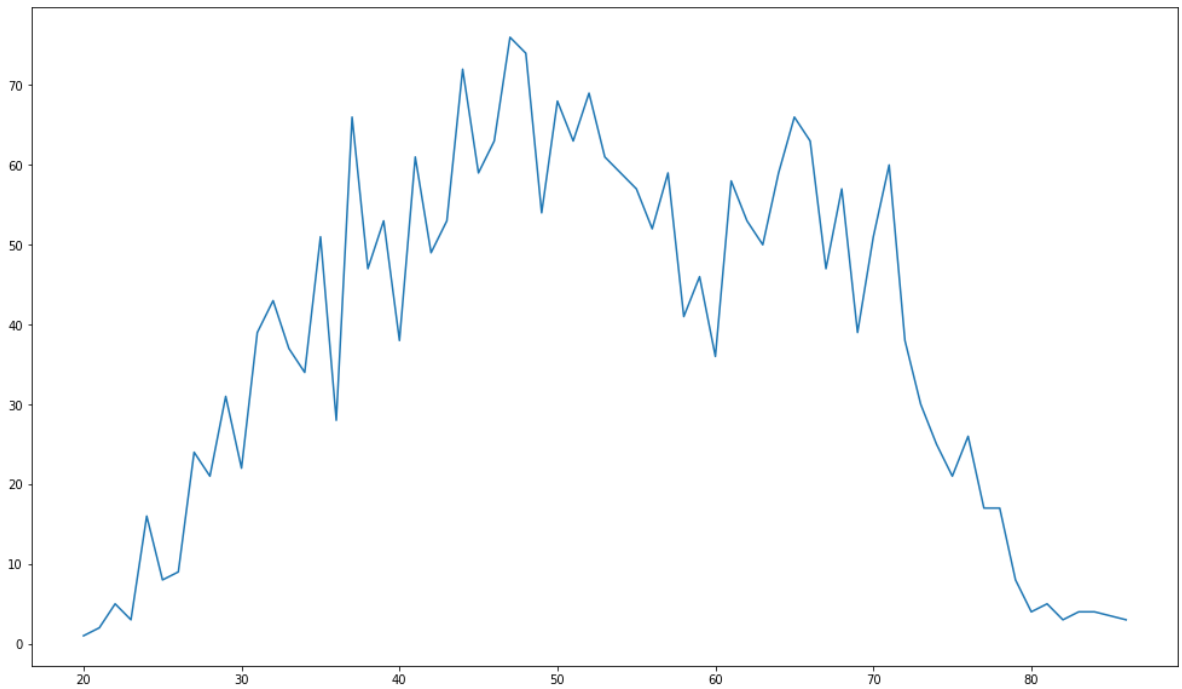


Entrée [14]:

```
ar = pd.DataFrame({'age':train.age.value_counts().index,'nb':train.age.value_counts(
fig = plt.figure(figsize=(17, 10))
plt.plot(ar.age,ar.nb)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x1a21776d30>]
```



Entrée [15]:

```
round(train.age.isna().sum()/train.age.shape[0],ndigits=3)
```

Out[15]:

```
0.405
```

La répartition de l'**âge** est intéressante, on semble distinguer une **hyperbole centrée en 52**. Cependant il y a tout de même trop de **NA** (**40%**) pour les remplacer par la **mediane** de la colonne **âge**. On pourrait cependant utiliser la **median/moyenne** d'âge en groupant par **region, profession ou personne à charge**.

Entrée [16]:

```
train.age.mean()
```

Out[16]:

```
52.0992963252541
```

Entrée [17]:

```
for p in train.PROF.unique():
    med = train.loc[train.PROF == p].age.median()
    train.loc[train.PROF == p, 'age'] = train.loc[train.PROF == p, 'age'].fillna(med
train.age.mean()
```

Out[17]:

```
52.28862077123255
```

Entrée [18]:

```
train.tranche_age.unique().tolist()
```

Out[18]:

```
[nan,
 '<25ans',
 '35-34ans',
 '55-64ans',
 '75ans_et+',
 '45-54ans',
 '40-44ans',
 '65-75ans']
```

Entrée [19]:

```
def trancheAge(x):
    age = x.age
    if age < 25:
        return '<25ans'

    elif age >= 25 and age <= 34:
        return  '25-34ans'

    elif age >= 35 and age <= 44:
        return  '35-44ans'

    elif age >= 45 and age <= 54:
        return  '45-54ans'

    elif age >= 55 and age <= 64:
        return  '55-64ans'

    elif age >= 65 and age <= 74:
        return  '65-74ans'

    elif age >= 75:
        return '>75ans'
```

Entrée [20]:

```
train.tranche_age = train.apply(lambda x: trancheAge(x), axis=1)
```

• IMPAYES DEBUT

Entrée [21]:

```
print(f'MAX: {pd.to_numeric(train.IMPAYES_DEBUT, errors="coerce").max()}')
print(f'MIN: {pd.to_numeric(train.IMPAYES_DEBUT, errors="coerce").min()}')
print(f'Median: {pd.to_numeric(train.IMPAYES_DEBUT, errors="coerce").median()}')
print(f'Moyenne: {round(pd.to_numeric(train.IMPAYES_DEBUT, errors="coerce").mean())}')
print(f'NA: {round(train.IMPAYES_DEBUT.isna().sum()/train.shape[0],ndigits=2)}')
```

```
MAX: 10889.0
MIN: 35.0
Median: 1107.5
Moyenne: 1199
NA: 0.85
```

Cette colonne contient beaucoup beaucoup **trop de NA**. Je ne vois pas, **pour l'instant**, de méthode pour la combler.

Nous la supprimons **pour l'instant**.

Entrée [22]:

```
train = train.drop(columns=['IMPAYES_DEBUT'])
```

• GAIN MEDIATION

Entrée [23]:

```
print(f'MAX: {pd.to_numeric(train.gain_mediation, errors="coerce").max()}')
print(f'MIN: {pd.to_numeric(train.gain_mediation, errors="coerce").min()}')
print(f'Median: {pd.to_numeric(train.gain_mediation, errors="coerce").median()}')
print(f'Moyenne: {round(pd.to_numeric(train.gain_mediation, errors="coerce").mean()}
print(f'NA: {round(train.gain_mediation.isna().sum()/train.shape[0],ndigits=2)}')
```

```
MAX: 4359.0
MIN: -16.0
Median: 514.0
Moyenne: 686
NA: 0.97
```

Entrée [24]:

```
train.gain_mediation.unique()
```

Out[24]:

```
array([nan, '32', '456', '309', '508', '613', '562', '-16', '1320', '5
54',
       '514', '1243', '1382', '181', '4359', '885', '555', '236', '132
1',
       '321', '252', '267', '552', '238', '733', '434', '952', '126',
       '1375', '956', '0', '394', '139', '297', '116', '338', '275',
       '1211', '292', '595', '1567', '543', '1168', '1169', '1924', '4
95',
       '114', '384', '2154', '1634', '1391', '843', '156', '213', '172
7',
       '720', '545', '1307', '90', '2328', '825', '975', '902', '106
4',
       '306', '459', '1051', '232', '754', '326', '433', '25', '656',
       '122', '308', '391', '1405', '198', '903', '425', '986', '999',
       '168', '476', '591', '322', '779', '1424', '891', '60', '553',
       '1359', '173', '343', '302', '1012', '96', '673', '1908', '72
5',
       '405', '94', '151', '671', '341', '453', '274', '24', '770', '1
67',
       '2419', '1723', '333', '526', '688', '74', '485', '544', '47
5'],
      dtype=object)
```

Entrée [25]:

```
train.gain_mediation = train.gain_mediation.fillna(0).astype(int)
```

- **cat_impayes**

Entrée [26]:

```
train.cat_impayes.value_counts(dropna=False)
```

Out[26]:

```
NaN              3635
1000€-1500€       246
500€-1000€        236
1501€-2000€       108
1€-499€            36
2001€-3000€        31
3001€-4000€         4
6000€ et plus       2
5001€-6000€         1
Name: cat_impayes, dtype: int64
```

Le nombre de **NA** pour cette colonne est **trop important** pour les supprimer. Cette colonnes étant catégorielle, **pouvons les conserver**.

Entrée [27]:

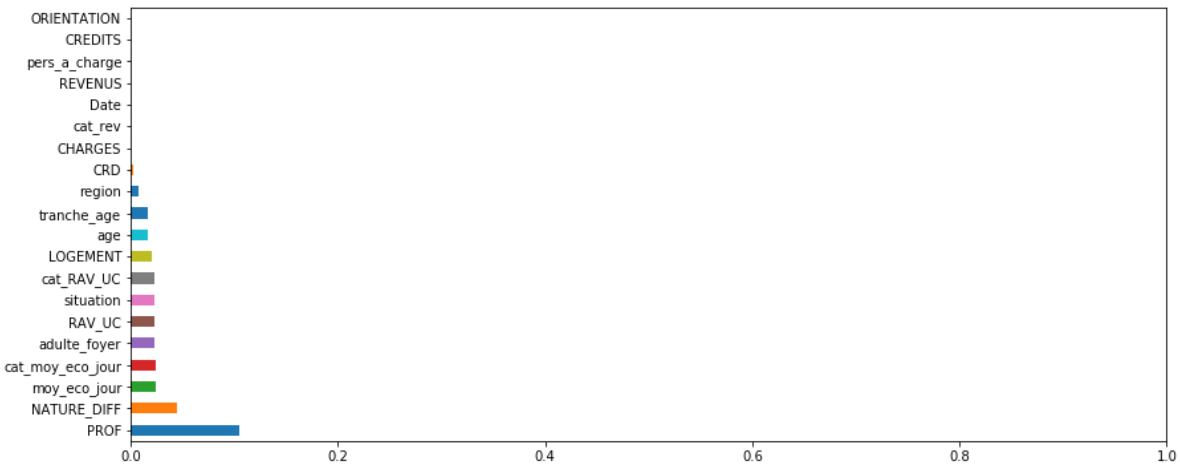
```
train.cat_impayes = train.cat_impayes.fillna('Inconnu')
```

-> Résultat du nettoyage des NA

Entrée [28]:

```
train_na = train.isna().sum()/train.shape[0]

fig, ax = plt.subplots()
fig.set_size_inches(14,6)
ax.set_xlim([0,1])
train_na.sort_values(ascending=False)[:20].plot(kind='barh')
plt.show()
```



Entrée [29]:

```
round(train.dropna().shape[0]/n,ndigits=2)
```

Out[29]:

```
0.86
```

Nous pouvons **supprimer** les **NA restants**, soit **14%**.

Entrée [30]:

```
train = train.dropna()
```

III - Traitement des données catégorielles

Entrée [31]:

```
cols_categorielles = list(train.dtypes[train.dtypes == 'object'].to_dict().keys())
train[cols_categorielles].head()
```

Out[31]:

	STRUCTURE PRESCRIPTRICE	PLATEFORME	region	ORIENTATION	NATURE_DIFF	tranche_a
0	PRESCRIPTEUR35	bancaire	BOURGOGNE FRANCHE COMTE	Surendettement	Surendettement	65-74a
1	PRESCRIPTEUR16	bancaire	OCCITANIE	Mediation	Endettement	65-74a
2	PRESCRIPTEUR18	social	PROVENCE-ALPES-COTE-D'AZUR	Accompagnement	Endettement	45-54a
3	PRESCRIPTEUR6	bancaire	BRETAGNE	Mediation	Multiendettement	65-74a
4	PRESCRIPTEUR23	social	NOUVELLE AQUITAINE	Accompagnement	Difficultés de Gestion	<25a

Entrée [32]:

```
for cc in cols_categorielles:
    cat_dtype = pd.api.types.CategoricalDtype(categories=train[cc].unique().tolist()
    train[cc] = train[cc].astype(cat_dtype)
```

Entrée [33]:

```
train.dtypes.head()
```

Out[33]:

```
id                          int64
STRUCTURE PRESCRIPTRICE    category
PLATEFORME                 category
region                     category
ORIENTATION                category
dtype: object
```

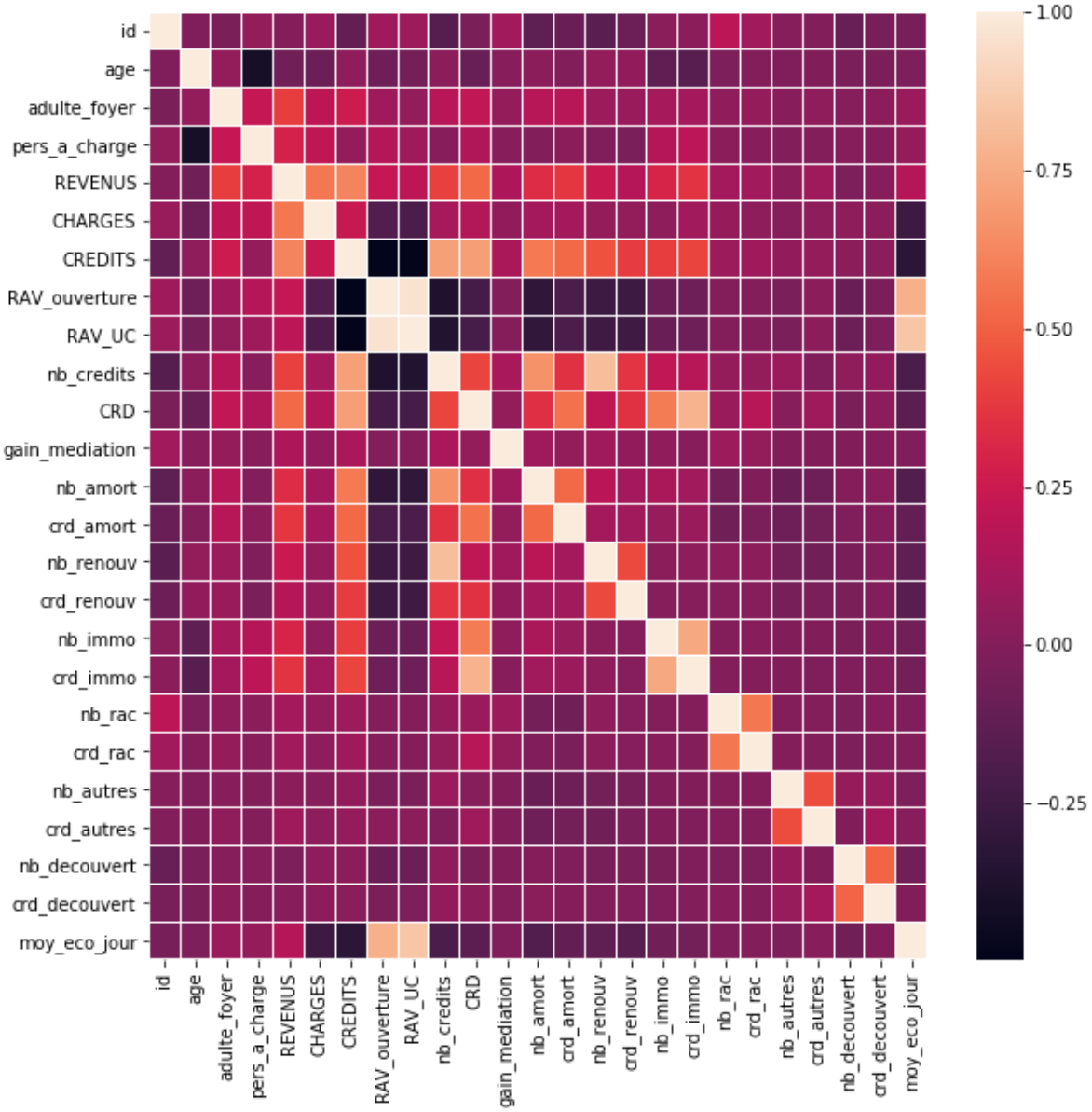
Entrée [34]:

```
corr = train.corr()
print(corr.shape)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr, xticklabels=corr.columns.values, yticklabels=corr.columns.values,
```

```
(25, 25)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a21ee2ba8>
```



Entrée [35]:

```
train.shape
```

Out[35]:

```
(3677, 42)
```