01_presentation_donnees

January 16, 2020

1 Caractériser les adhérents de la Médiathèque de Roubaix selon leur lieu d'habitation - partie 1 : Présentation et exploration des données

1.1 Le jeu de données

On part de données mises en ligne sur la plateforme open data de la Ville de Roubaix.

Ces données ont vocation à permettre de caractériser les adhérents de la Médiathèque de Roubaix, pour une année donnée (2018 ici).

[1]:
```
import pandas as pd

adh = pd.read_csv("data/caracteristiques_adherents_2018.csv", sep=";")
adh.head(5)
```

[1]:
```
   date_extraction      activite activite_emprunteur activite_emprunteur_bus  \
0             2018          prêt          Emprunteur    Non emprunteur Zèbre
1             2018          prêt          Emprunteur    Non emprunteur Zèbre
2             2018  aucune trace      Non emprunteur    Non emprunteur Zèbre
3             2018          prêt          Emprunteur    Non emprunteur Zèbre
4             2018  aucune trace      Non emprunteur    Non emprunteur Zèbre

      activite_emprunteur_med           activite_salle_etude  \
0      Emprunteur Médiathèque  Non utilisateur Salle d'étude
1      Emprunteur Médiathèque  Non utilisateur Salle d'étude
2  Non emprunteur Médiathèque  Non utilisateur Salle d'étude
3      Emprunteur Médiathèque  Non utilisateur Salle d'étude
4  Non emprunteur Médiathèque  Non utilisateur Salle d'étude

  activite_utilisateur_postes_informatiques activite_utilisateur_wifi  \
0      Non utilisateur postes informatiques      Non utilisateur Wifi
1      Non utilisateur postes informatiques      Non utilisateur Wifi
2      Non utilisateur postes informatiques      Non utilisateur Wifi
3      Non utilisateur postes informatiques      Non utilisateur Wifi
4      Non utilisateur postes informatiques      Non utilisateur Wifi

  Tranches d'âge (1) Tranches d'âge (2)  … type_inscription  nb_venues  \
```

```
0        25 - 64 ans        30 - 39 ans  …         gratuite        1.0
1         0 - 14 ans        06 - 10 ans  …         gratuite        1.0
2         0 - 14 ans        06 - 10 ans  …         gratuite        0.0
3     65 ans et plus        65 - 79 ans  …         gratuite       13.0
4        15 - 24 ans        15 - 17 ans  …         gratuite        0.0

  nb_venues_postes_informatiques nb_venues_prets nb_venues_prets_bus  \
0                            0.0            1.0                 0.0
1                            0.0            1.0                 0.0
2                            0.0            0.0                 0.0
3                            0.0           13.0                 0.0
4                            0.0            0.0                 0.0

  nb_venues_prets_mediatheque nb_venues_salle_etude  nb_venues_wifi   sexe  \
0                         1.0                  0.0             0.0  Homme
1                         1.0                  0.0             0.0  Femme
2                         0.0                  0.0             0.0  Femme
3                        13.0                  0.0             0.0  Homme
4                         0.0                  0.0             0.0  Femme

                 geo_point_2d
0  50.6873356245,3.16102905857
1  50.6873356245,3.16102905857
2  50.6873356245,3.16102905857
3  50.6873356245,3.16102905857
4  50.6873356245,3.16102905857

[5 rows x 28 columns]
```

1.2 Individus et variables

Chaque ligne correspond à un individu (la taille de l'échantillon est de 13288 individus), pour lequel on dispose à la fois de variables qui permettent de connaître : - ses caractéristiques propres : - tranche d'âge, - sexe, - commune de résidence, - quartier (IRIS) de résidence si l'adhérent habite Roubaix, - son utilisation de l'équipement : - utilise-t-il ou non tel ou tel service ? combien de fois dans l'année ? - depuis combien de temps fréquent-il la Médiathèque ? - selon quelles modalités s'est-il inscrit ?

On dispose donc de variables : - qualitatives - nominales : - activite - activite_emprunteur - activite_emprunteur_bus - activite_emprunteur_med - activite_salle_etude - activite_utilisateur_postes_informatiques - activite_utilisateur_wifi - Roubaisien ou non - Code IRIS de Roubaix - Nom de l'IRIS à Roubaix - Commune de résidence - inscription_attribut_action - inscription_attribut_zebre - inscription_carte - type_inscription - sexe - ordinales : - Tranches d'âge (1) - Tranches d'âge (2) - quantitatives - nombre d'années d'adhésion - nb_venues - nb_venues_postes_informatiques - nb_venues_prets - nb_venues_prets_bus - nb_venues_prets_mediatheque - nb_venues_salle_etude - nb_venues_wifi

1.3 Exploration des variables et filtrage des données

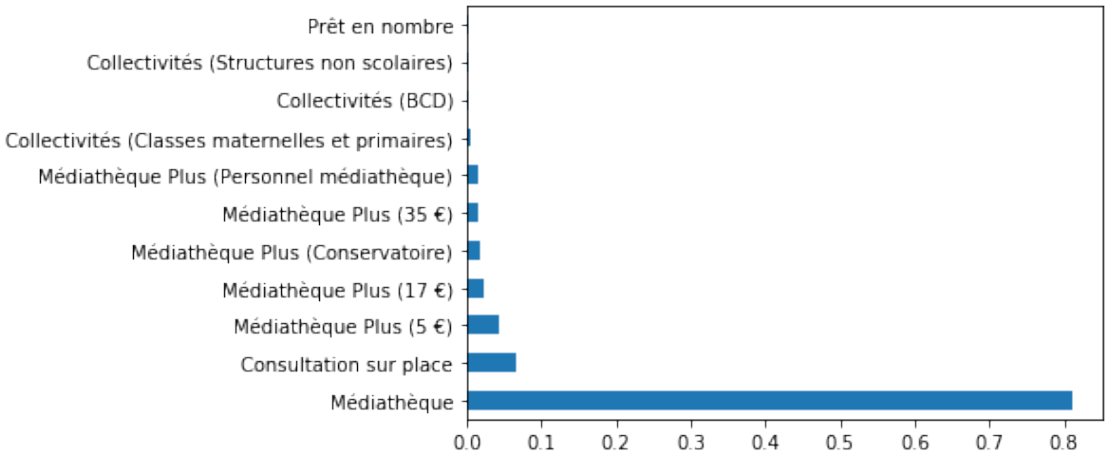
On procède à la vérification des différentes valeurs (avec pour objectif de mieux les comprendre, repérer d'éventuelles aberrations et aussi de ne garder que les informations qui nous intéressent).

1.3.1 Exploration des vairables : typologie des personnes

On ne souhaite garder que les personnes physiques, on commence donc par examiner le type de cartes et à filtrer sur les cartes attribuées uniquement à des personnes physiques :

[2]:
```
adh['inscription_carte'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

[2]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5acb4f3590>
```



[3]:
```
cartes_ok = ['Médiathèque', 'Consultation sur place', 'Médiathèque Plus (5 €)',␣
↪'Médiathèque Plus (17 €)','Médiathèque Plus (Conservatoire)', 'Médiathèque␣
↪Plus (35 €)']
adh = adh[adh['inscription_carte'].isin(cartes_ok)]
adh['date_extraction'].count()
```

[3]:
```
12964
```

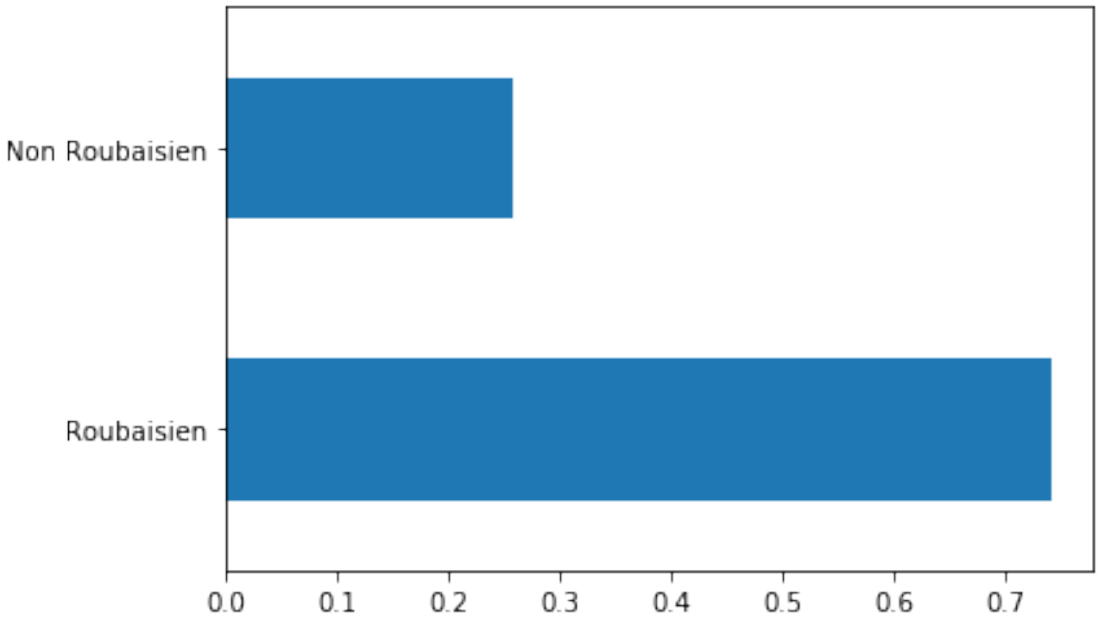
On réduit donc l'échantillon de 13288 à 12964.

1.3.2 Exploration des variables : information géographique

On poursuit par les informations de type géographique, sachant qu'on ne gardera que les lignes concernant des adhérents dont la commune de résidence est Roubaix et pour lesquels on dispose d'un code IRIS. - Roubaisien ou non :

[4]:
```
adh['Roubaisien ou non'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

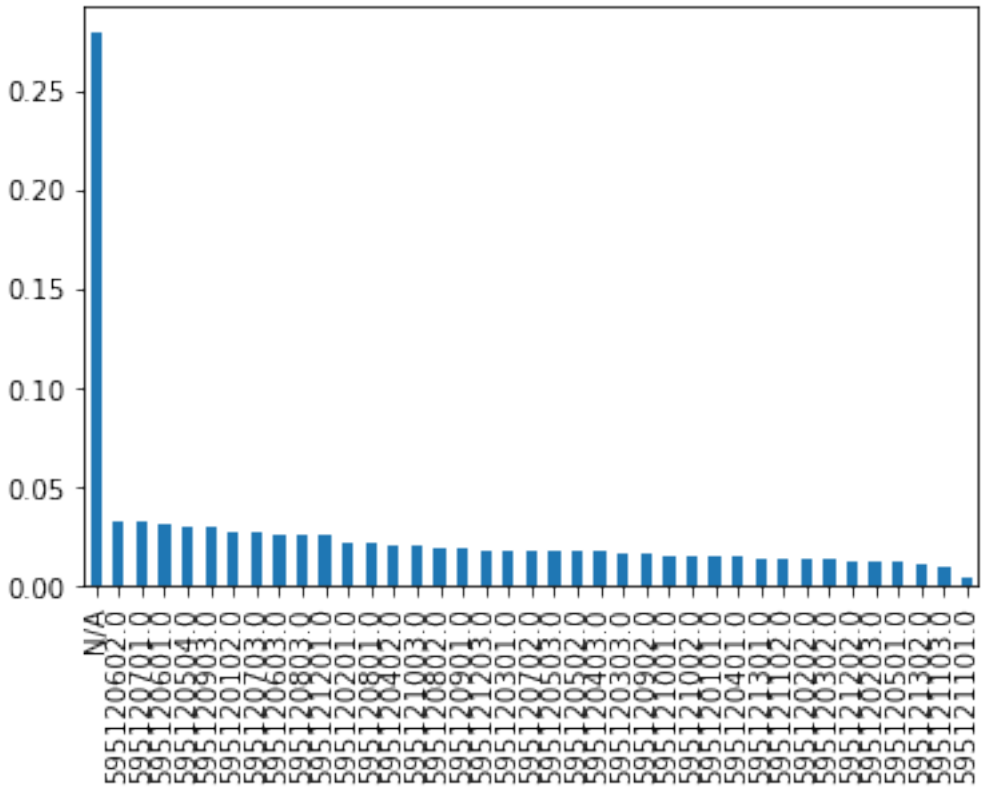
[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5aca4bca90>



- Code IRIS de Roubaix :

[5]:
```
adh['Code IRIS de Roubaix'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='bar')
```

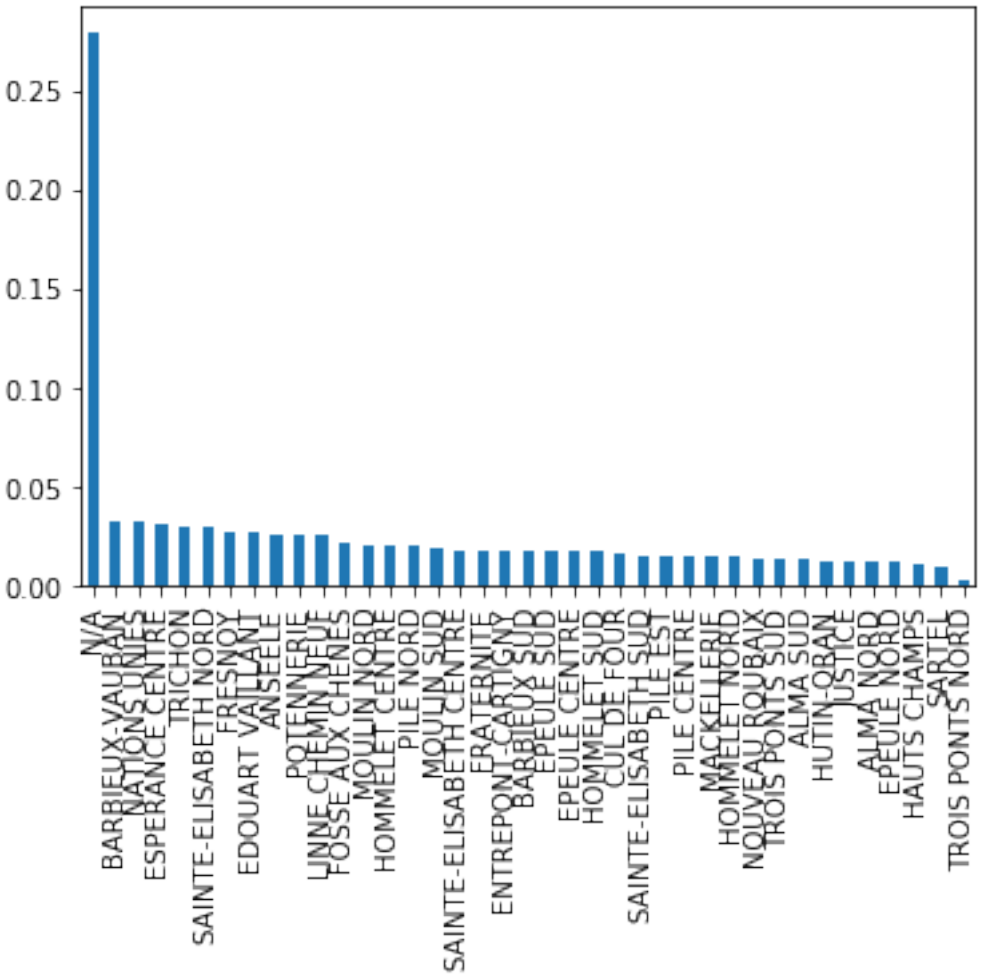
[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5aca41f2d0>



- Nom de l'IRIS à Roubaix :

[6]:
```
adh['Nom de l'IRIS à Roubaix'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='bar')
```

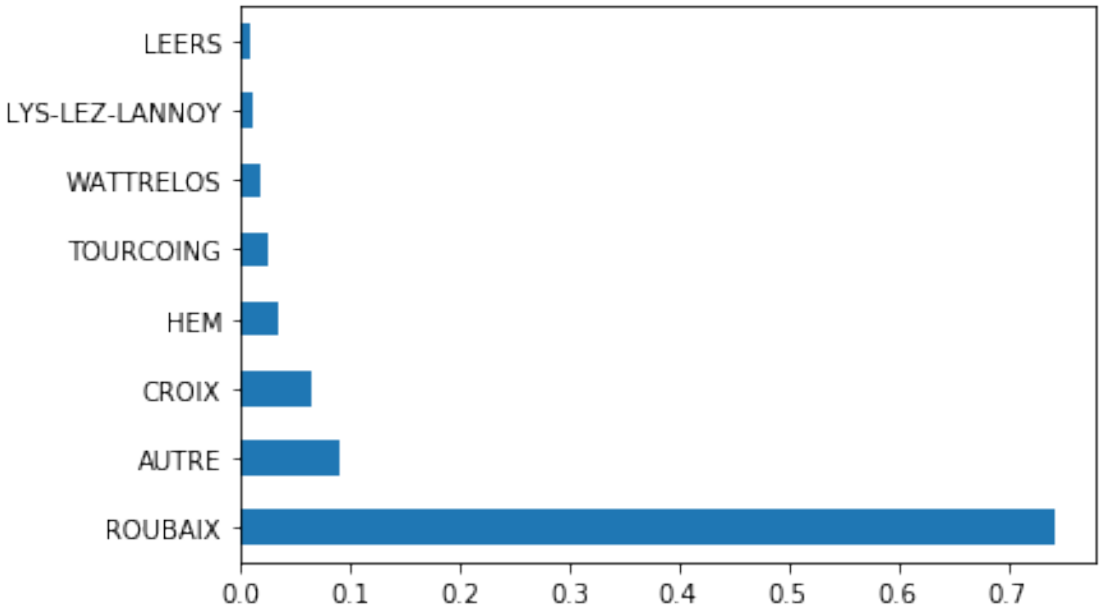
[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5aca0a1f10>



- Commune de résidence :

[7]:
```
adh['Commune de résidence'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9f90b50>
```

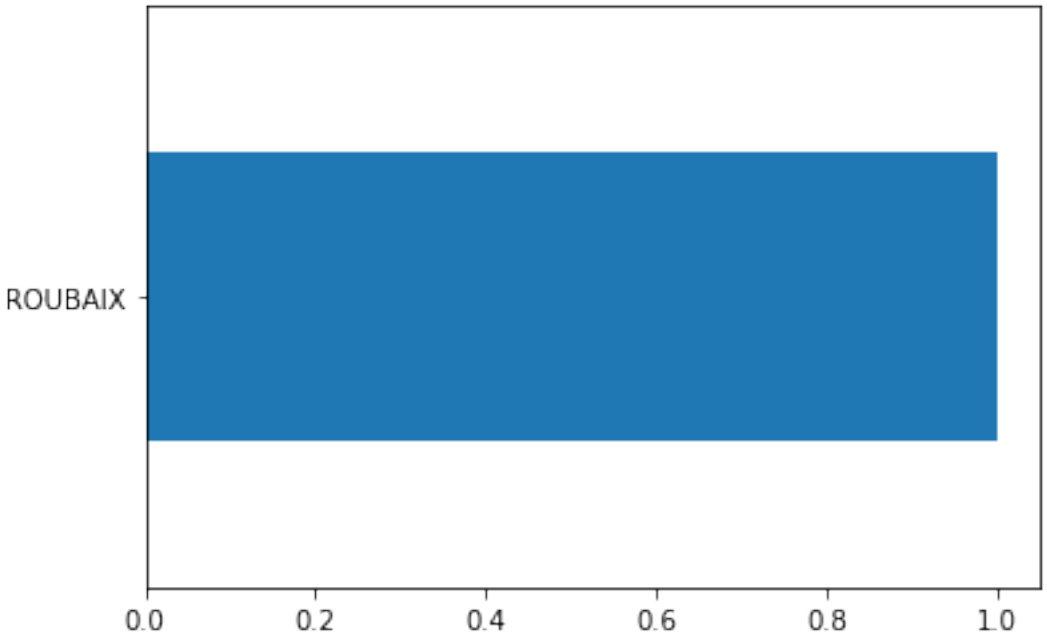


On effectue quelques tests de cohérence, puis on filtre les lignes en ne retenant que les Roubaisiens pour lesquels on dispose d'un IRIS.

On vérifie que les données des variables "Commune de résidence" et "Roubaisien ou non" sont cohérentes entre elles.

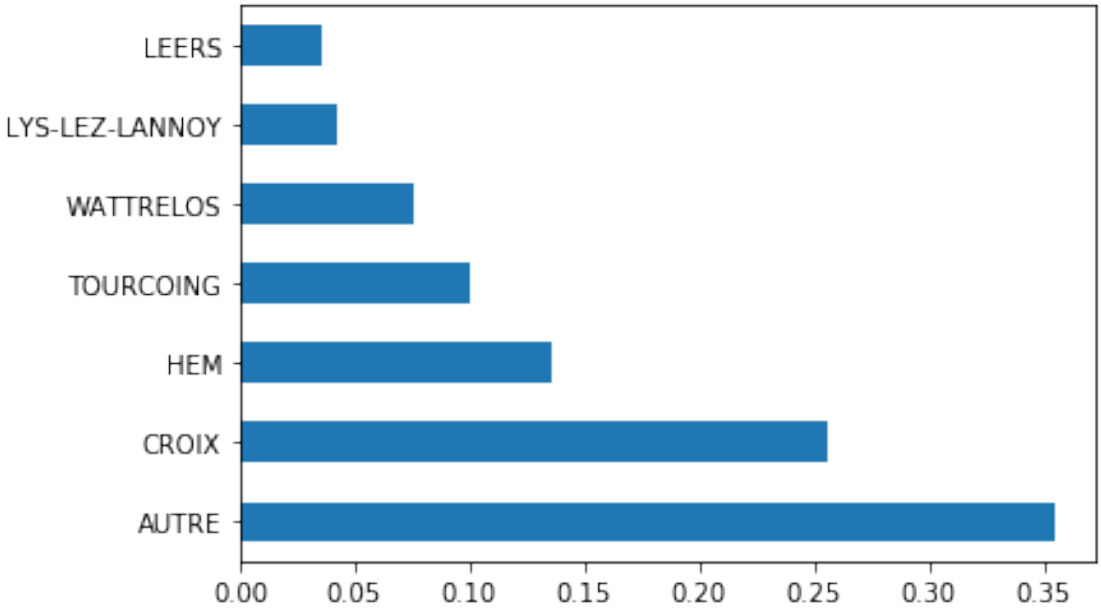
[8]:
```
adh['Commune de résidence'][adh['Roubaisien ou non'] == 'Roubaisien'].
↪value_counts(normalize=True).plot(kind='barh')
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9ebcd50>
```



[9]:
```
adh['Commune de résidence'][adh['Roubaisien ou non'] == 'Non Roubaisien'].
↪value_counts(normalize=True).plot(kind='barh')
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9ec47d0>



Pas d'incohérence, on peut ne garder que les habitants de Roubaix par exemple en se basant sur la variable 'Roubaisien ou non'.

[10]:
```
adh = adh[adh['Roubaisien ou non'] == 'Roubaisien']
adh['date_extraction'].count()
```

[10]:
```
9624
```

La taille de l'échantillon passe donc de 12964 à 9624 individus. On vérifie ensuite que tous les Roubaisiens disposent d'un code IRIS, dans le cas contraire on ne garde que les lignes en disposant et on calcule le taux d'erreur.

[11]:
```
nb_roubaisiens = adh['date_extraction'].count()
adh['Code IRIS de Roubaix'] =  adh['Code IRIS de Roubaix'].fillna('N/A')
nb_roubaisiens_iris_ko = adh['date_extraction'][adh['Code IRIS de Roubaix'] ==␣
↪'N/A'].count()
tx_iris_ko = (nb_roubaisiens_iris_ko / nb_roubaisiens).round(2)
print(f"Le taux d'adhérents habitant à Roubaix et pour lesquels on ne dispose␣
↪pas d'un code IRIS est : {tx_iris_ko}")
```

```
Le taux d'adhérents habitant à Roubaix et pour lesquels on ne dispose pas d'un
code IRIS est : 0.03
```

On supprime les lignes pour lesquelles le code IRIS est absent :

[12]:
```
adh = adh[adh['Code IRIS de Roubaix'] != 'N/A']
adh['date_extraction'].count()
```

[12]:
```
9341
```

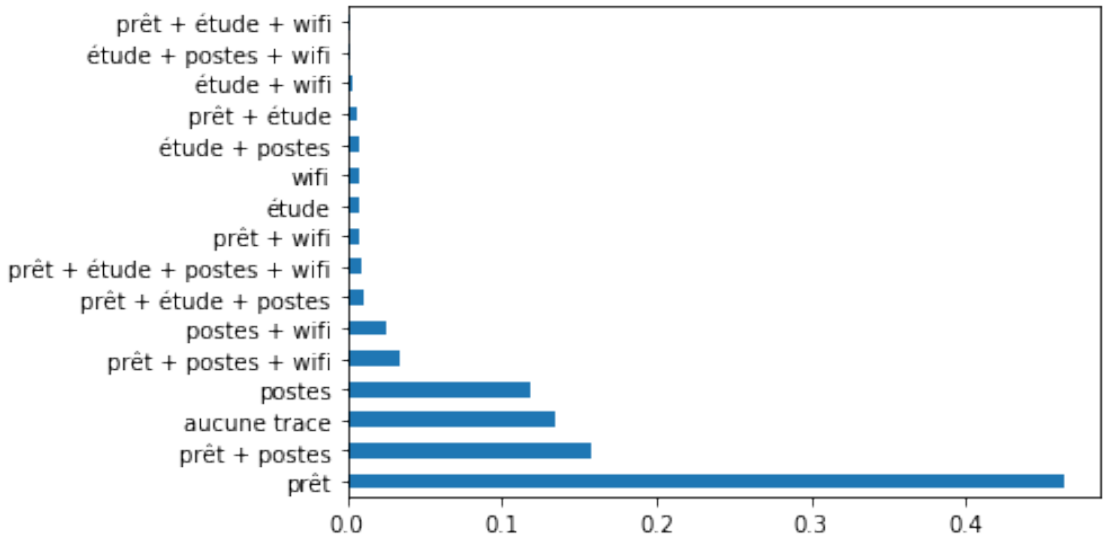
La taille de l'échantillon passe de 9624 à 9341 indvidus.

1.3.3 Exploration des variables : pratiques des inscrits

Une fois que l'on n'a conservé que les lignes nous intéressant, on regarde les autres variables qualitatives : - activité :

[13]:
```
adh['activite'].fillna('N/A').value_counts(normalize=True).plot(kind='barh')
```

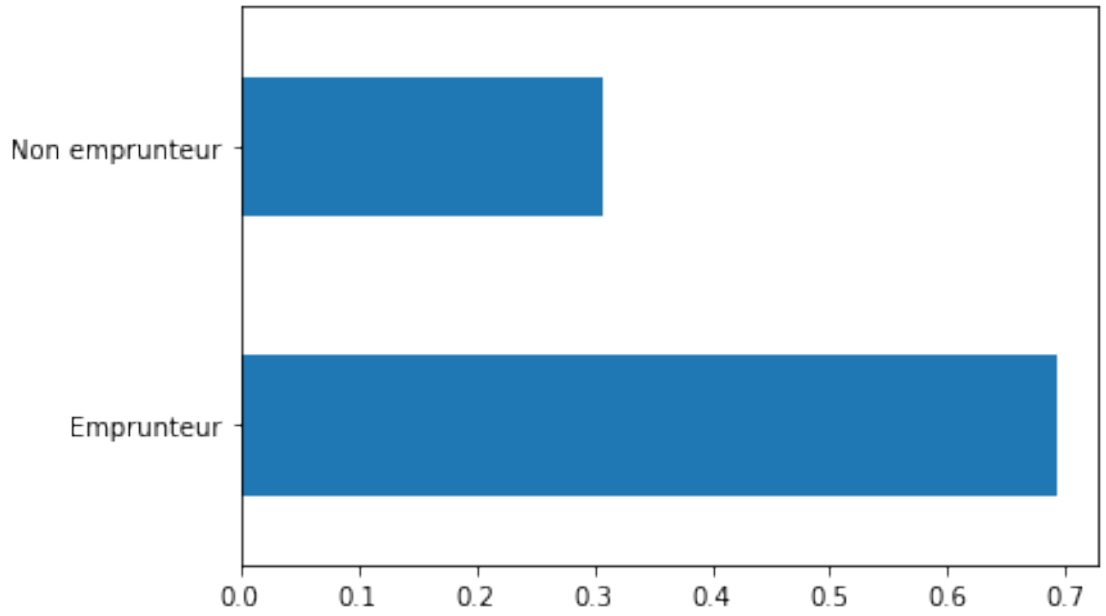
[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9fb25d0>
```



- activite_emprunteur :

[14]:
```
adh['activite_emprunteur'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

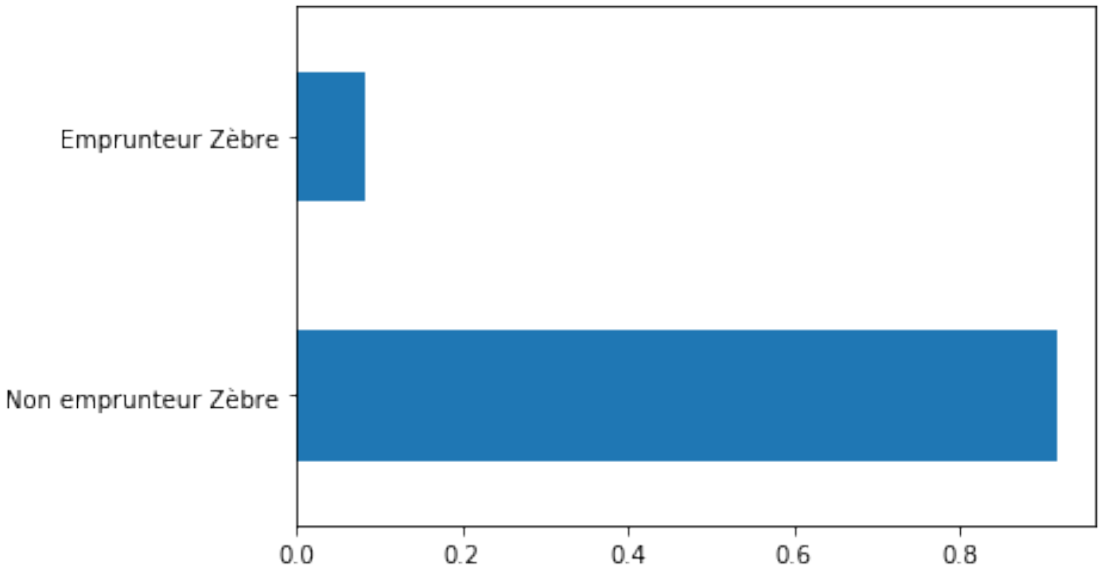
[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5aca3227d0>
```



- activite_emprunteur_bus :

[15]:
```
adh['activite_emprunteur_bus'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

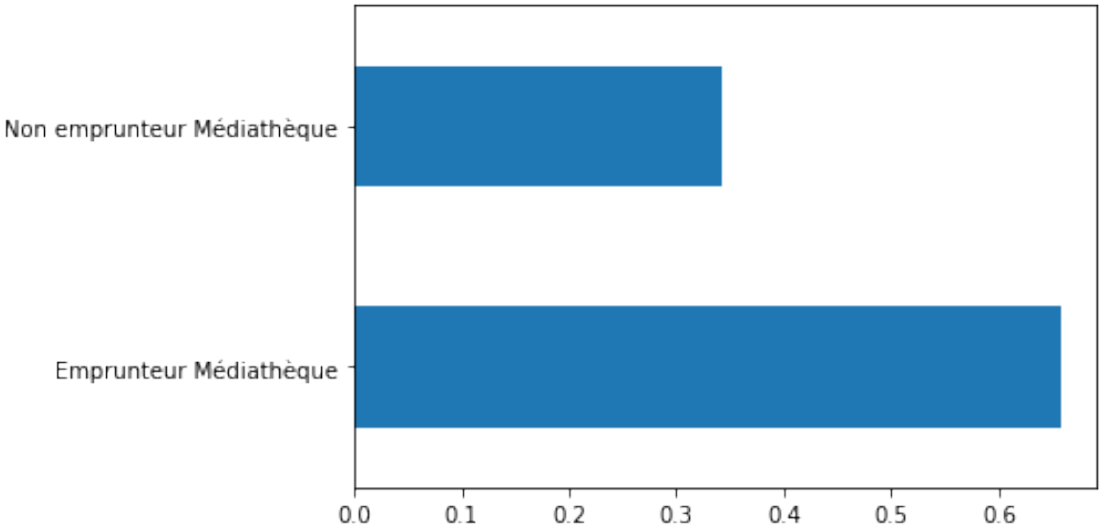
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5aca291e50>



- activite_emprunteur_med :

[16]:
```
adh['activite_emprunteur_med'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

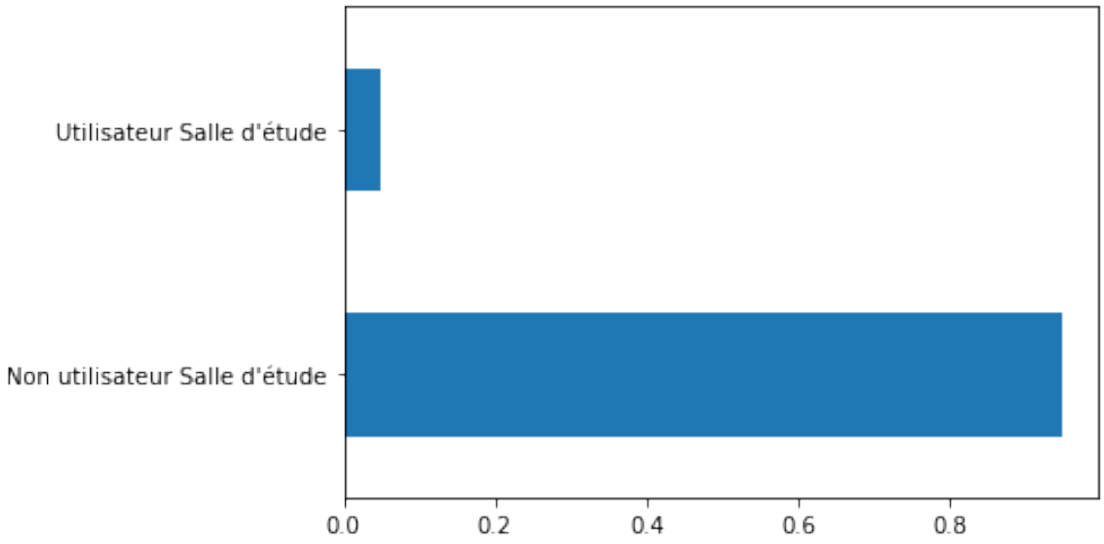
[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5aca27d5d0>



- activite_salle_etude :

[17]:
```
adh['activite_salle_etude'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

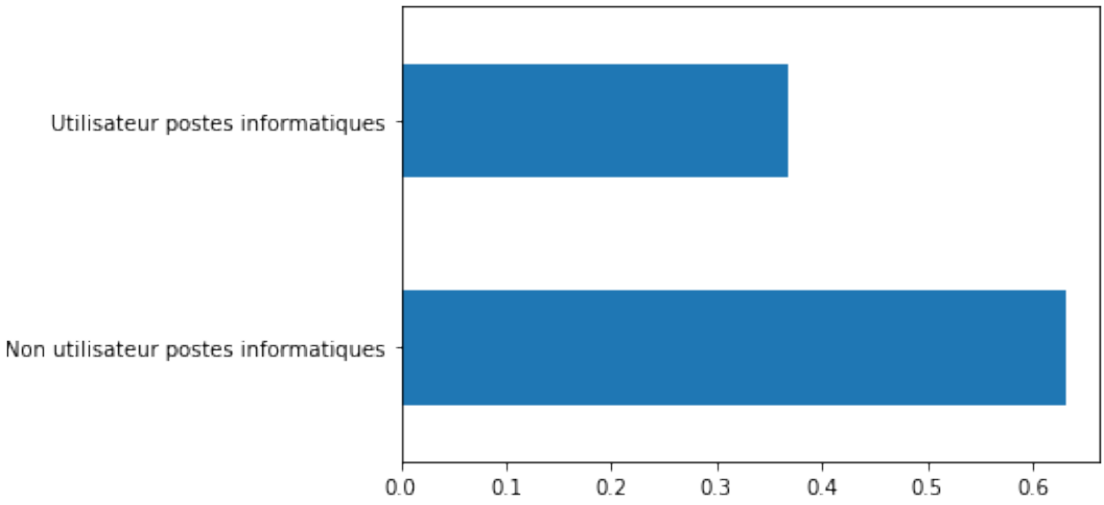
[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9d9cbd0>
```



- activite_utilisateur_postes_informatiques :

[18]:
```
adh['activite_utilisateur_postes_informatiques'].fillna('N/A').
↪value_counts(normalize=True).plot(kind='barh')
```

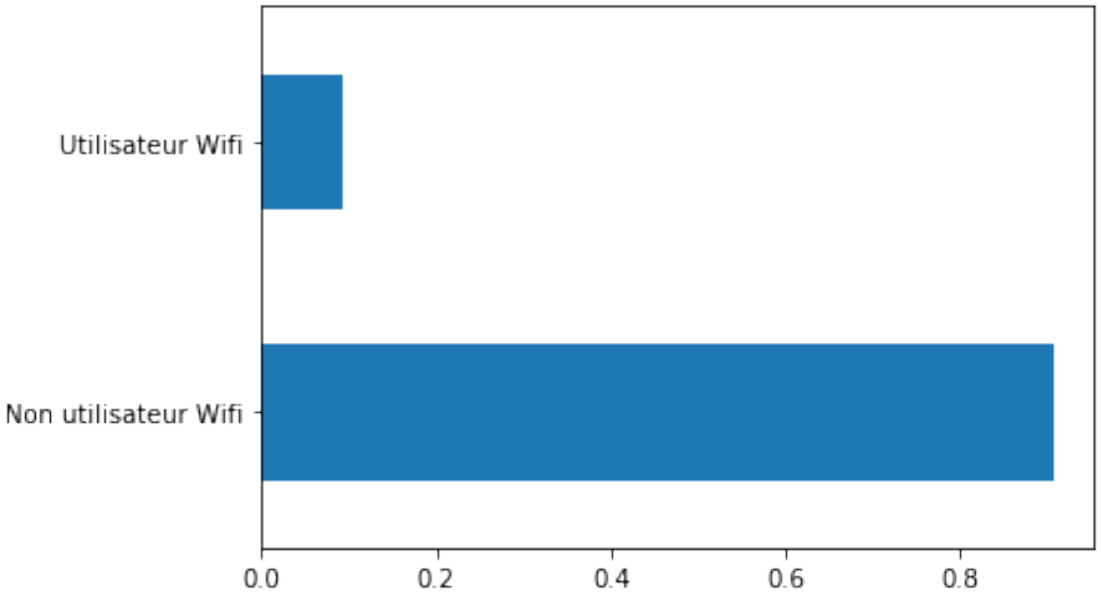
[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9d6c750>
```



- activite_utilisateur_wifi :

[19]:
```
adh['activite_utilisateur_wifi'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

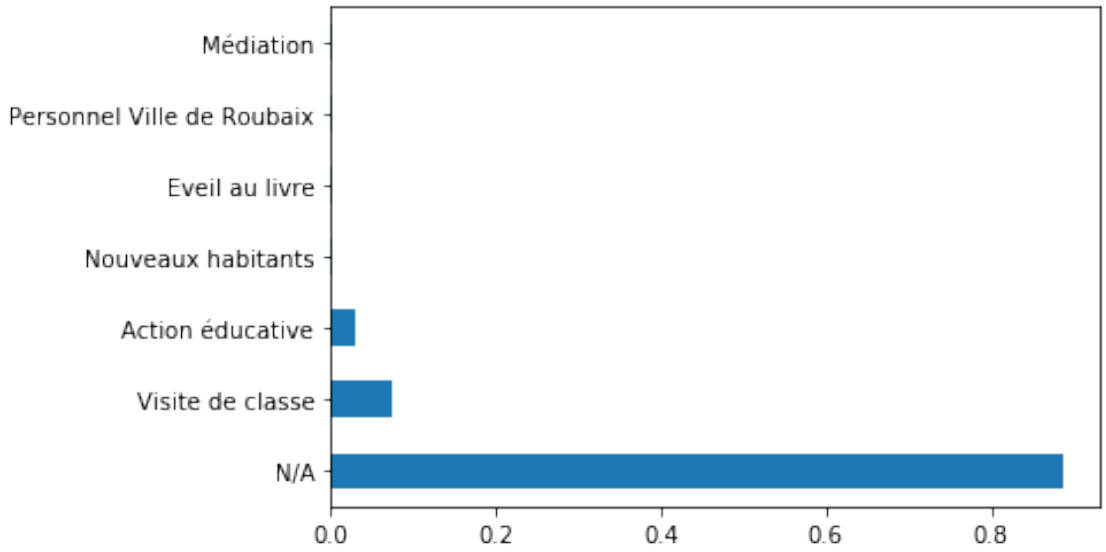
[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9cea3d0>
```



- inscription_attribut_action :

[20]:
```
adh['inscription_attribut_action'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

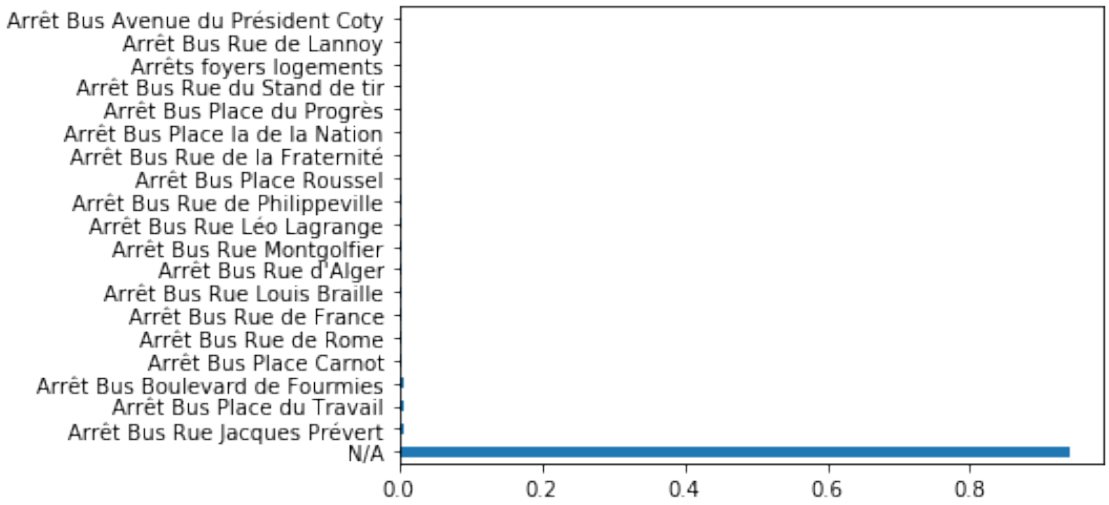
[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9c52590>
```



- inscription_attribut_zebre :

[21]:
```
adh['inscription_attribut_zebre'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9bc5dd0>
```

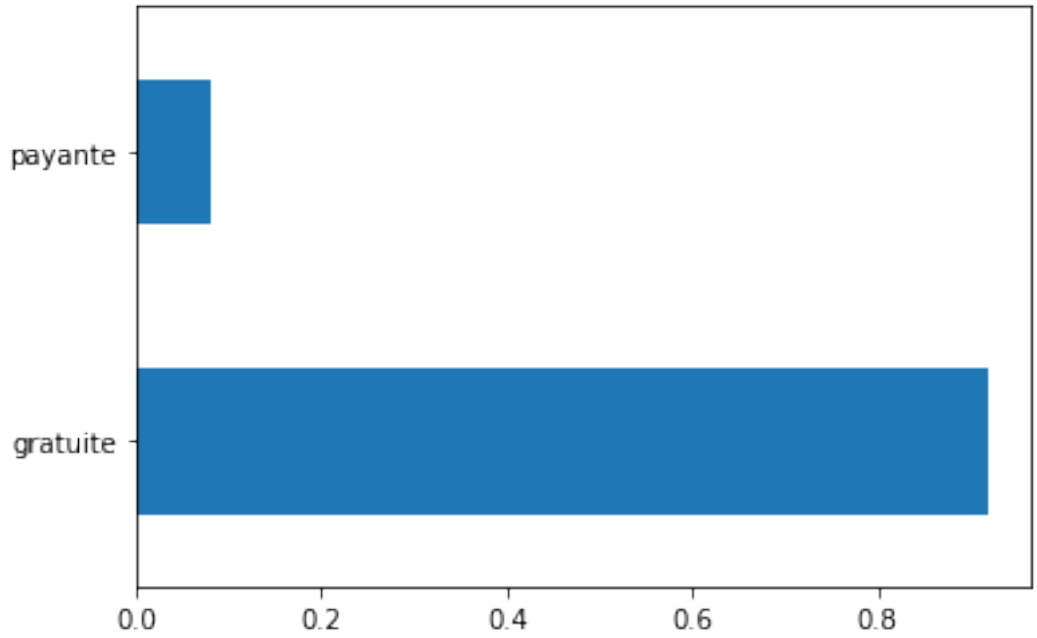


Cette variable paraît offrir peu d'informations, on ne la retiendra donc pas.

- type-inscription

[22]:
```
adh['type_inscription'].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

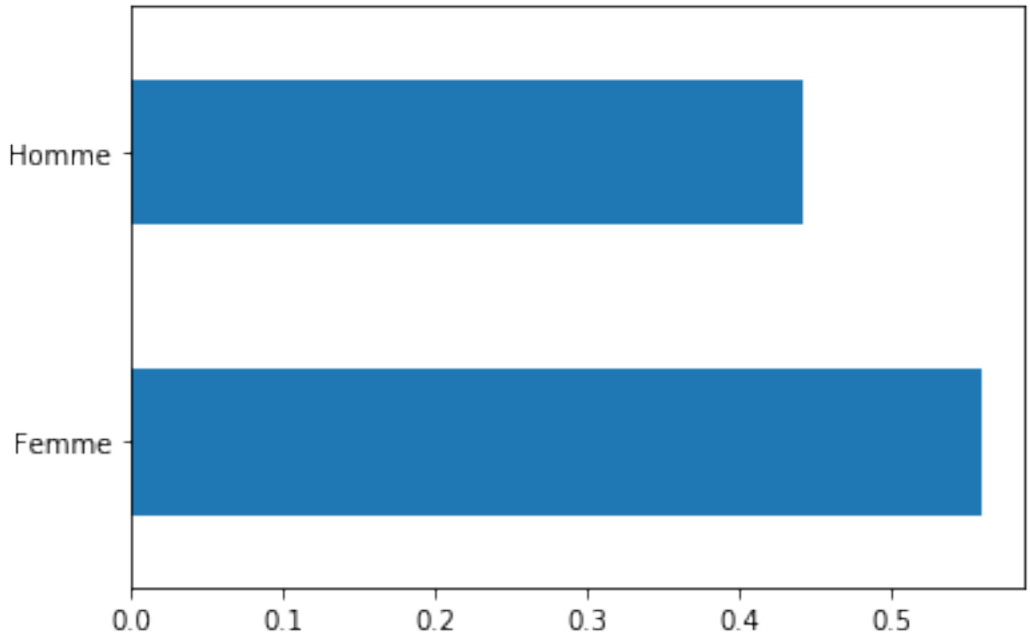
[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9b00a10>



- sexe :

[23]:
```
adh['sexe'].fillna('N/A').value_counts(normalize=True).plot(kind='barh')
```

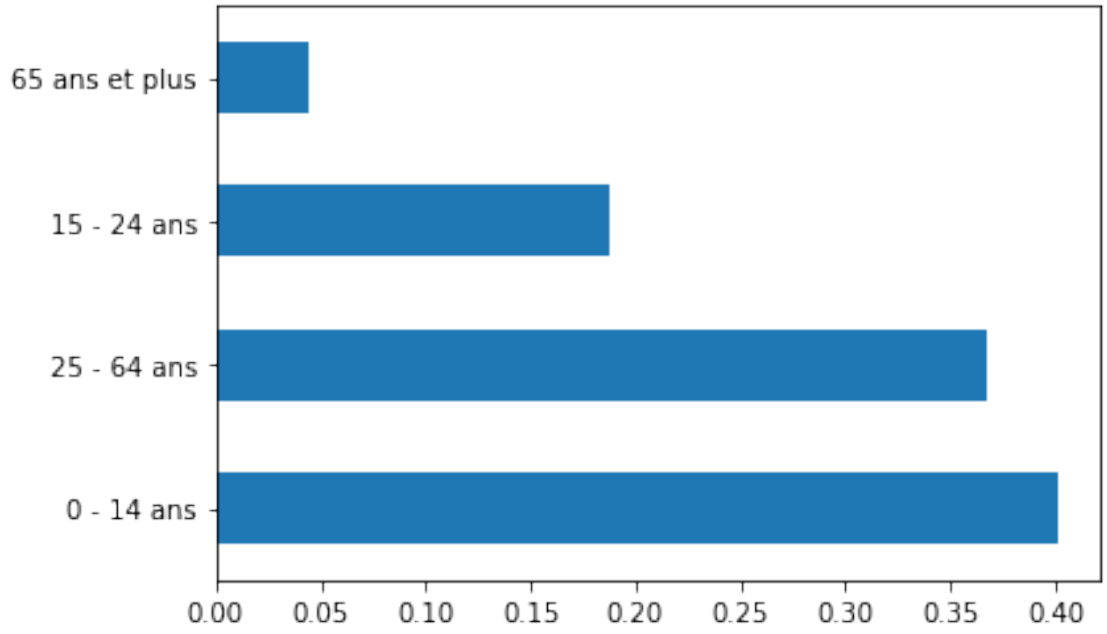
[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9ac9dd0>



On poursuit par les variables qualitatives ordinales : - Tranches d'âge (1)

[24]:
```
adh["Tranches d'âge (1)"].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

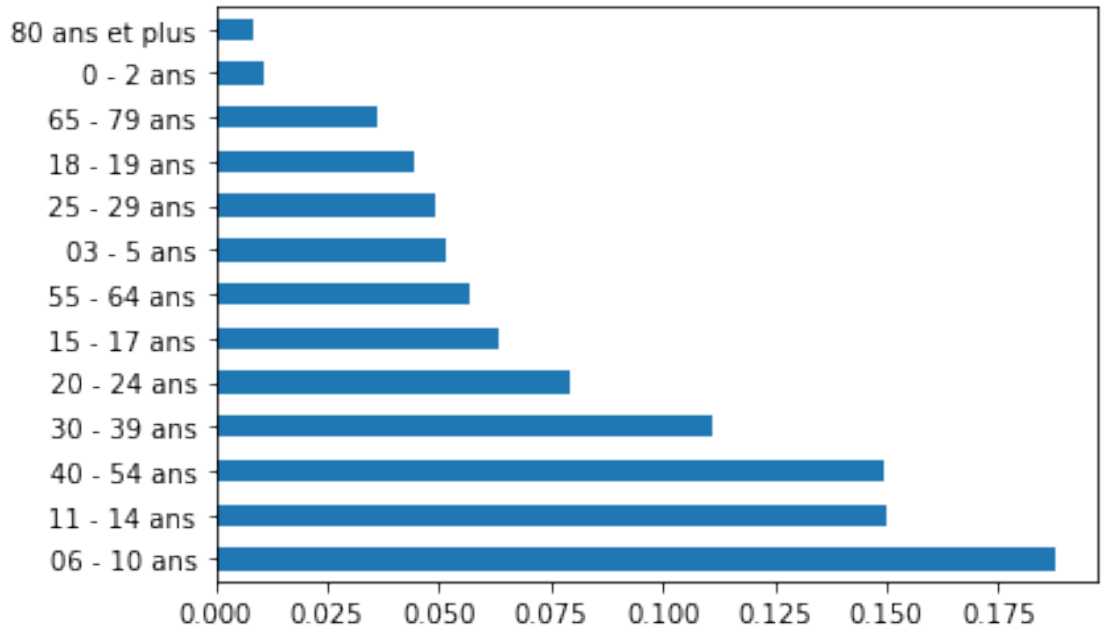
[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9a53a10>
```



- Tranches d'âge (2)

[25]:
```
adh["Tranches d'âge (2)"].fillna('N/A').value_counts(normalize=True).
↪plot(kind='barh')
```

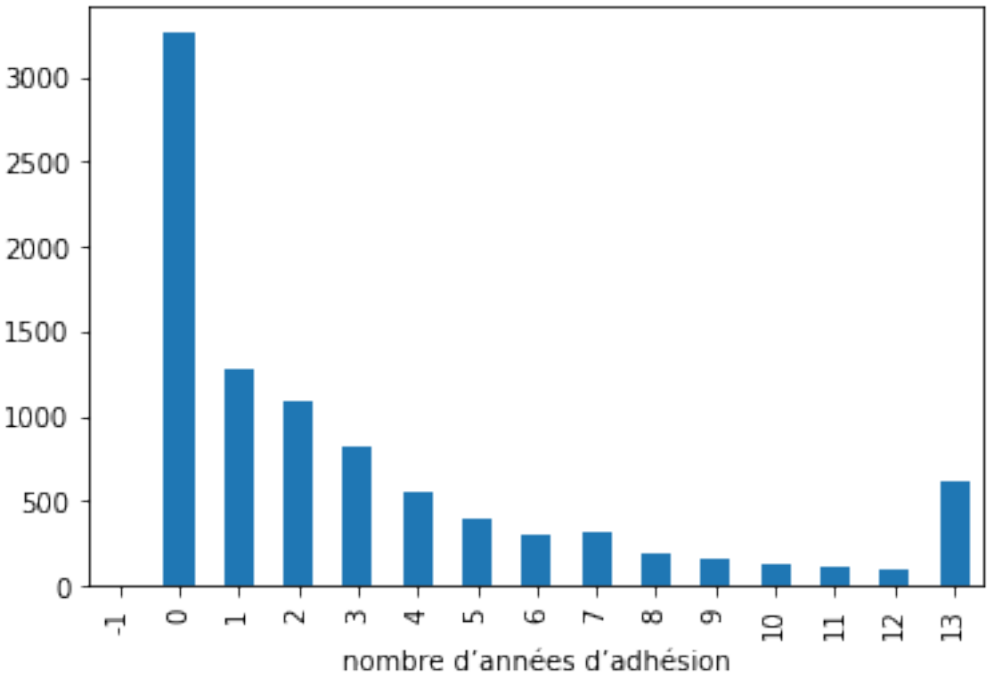
[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac9a53290>
```



On regarde enfin les variables quantitatives : - nombre d'années d'adhésion

[26]:
```
adh.groupby('nombre d'années d'adhésion')['nombre d'années d'adhésion'].count().
↪plot.bar()
```

[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ac99645d0>
```



On trouve une valeur aberrante (-1), on la corrige en 0 :

[27]:
```
adh.loc[adh['nombre d'années d'adhésion'] == -1, 'nombre d'années d'adhésion']␣
↪= 0
```

[28]:
```
adh['nombre d'années d'adhésion'].describe()
```

[28]:
```
count    9341.000000
mean        3.099882
std         3.867946
min         0.000000
25%         0.000000
50%         2.000000
75%         4.000000
max        13.000000
Name: nombre d'années d'adhésion, dtype: float64
```

- nb_venues

[29]:
```
adh['nb_venues'].describe()
```

[29]:
```
count    9341.000000
mean        7.723798
std        15.678937
min         0.000000
```

```
25%         1.000000
50%         3.000000
75%         8.000000
max       251.000000
Name: nb_venues, dtype: float64
```

- nb_venues_postes_informatiques

[30]:
```
adh['nb_venues_postes_informatiques'].describe()
```

[30]:
```
count    9341.000000
mean        3.551975
std        13.341006
min         0.000000
25%         0.000000
50%         0.000000
75%         2.000000
max       251.000000
Name: nb_venues_postes_informatiques, dtype: float64
```

- nb_venues_prets

[31]:
```
adh['nb_venues_prets'].describe()
```

[31]:
```
count    9341.000000
mean        4.193769
std         7.971209
min         0.000000
25%         0.000000
50%         2.000000
75%         5.000000
max       211.000000
Name: nb_venues_prets, dtype: float64
```

- nb_venues_prets_bus

[32]:
```
adh['nb_venues_prets_bus'].describe()
```

[32]:
```
count    9341.000000
mean        0.356921
std         1.886956
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max        52.000000
Name: nb_venues_prets_bus, dtype: float64
```

- nb_venues_prets_mediatheque

[33]:
```
adh['nb_venues_prets_mediatheque'].describe()
```

[33]:
```
count    9341.000000
mean        3.836848
std         7.805460
min         0.000000
25%         0.000000
50%         1.000000
75%         4.000000
max       211.000000
Name: nb_venues_prets_mediatheque, dtype: float64
```

- nb_venues_salle_etude

[34]:
```
adh['nb_venues_salle_etude'].describe()
```

[34]:
```
count    9341.000000
mean        0.335938
std         4.318674
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       215.000000
Name: nb_venues_salle_etude, dtype: float64
```

- nb_venues_wifi

[35]:
```
adh['nb_venues_wifi'].describe()
```

[35]:
```
count    9341.000000
mean        0.488813
std         4.543981
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       218.000000
Name: nb_venues_wifi, dtype: float64
```

Sur l'ensemble de ces données quantitatives, on constate deux choses, sachant que l'on va ensuite regrouper toutes ces données selon les IRIS : - le recours à la médiane est inopérant pour la plupart des variables, car celle-ci est souvent égale à 0. On va donc devoir recourir à la moyenne, - on a pour chaque variable des valeurs maximales aberrantes, i.e. très éloignées des autres valeurs de l'échantillon. Si on doit utiliser des moyennes, on va devoir supprimer ces valeurs aberrantes. On propose de supprimer toutes les valeurs supérieures à plus de 1,5 fois l'écart interquartile + le troisième quartile. On évalue auparavant le nombre de lignes qui serait concernées :

[36]:
```
quanti = ['nb_venues', 'nb_venues_postes_informatiques', 'nb_venues_prets',␣
 ↪'nb_venues_prets_bus', 'nb_venues_prets_mediatheque',␣
 ↪'nb_venues_salle_etude', 'nb_venues_wifi']
rejet = pd.DataFrame()
for q in quanti:
    q1 = adh[q].quantile(0.25)
    q3 = adh[q].quantile(0.75)
    eiq = q3 - q1
    r = adh[adh[q] > (q3 + 1.5 * eiq)]
    rejet = pd.concat([rejet, r])
rejet = rejet.drop_duplicates()
rejet['date_extraction'].count()
```

[36]:
```
3255
```

On obtient beaucoup trop de lignes à rejeter (près d'un tiers de l'échantillon). On refait un test en se contentant d'éliminer les valeurs supérieures au centile 99 :

[37]:
```
quanti = ['nb_venues', 'nb_venues_postes_informatiques', 'nb_venues_prets',␣
 ↪'nb_venues_prets_bus', 'nb_venues_prets_mediatheque',␣
 ↪'nb_venues_salle_etude', 'nb_venues_wifi']
rejet = pd.DataFrame()
for q in quanti:
    r = adh[adh[q] > adh[q].quantile(0.99)]
    rejet = pd.concat([rejet, r])
rejet = rejet.drop_duplicates()
rejet['date_extraction'].count()
```

[37]:
```
392
```

On obtient 5% de l'échantillon à supprimer, on renonce également.

1.3.4 On sauvegarde le nouveau jeu de données obtenu

[38]:
```
adh.to_csv("data/adh.csv", header=True, index=False)
```

[ ]:
```

```