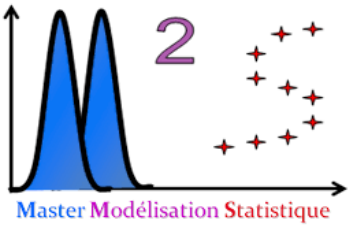


MASTER 2 MODÉLISATION STATISTIQUE

Rapport du projet de Python

Prédiction location d'un nouvel appartement sur Airbnb à Beijing

---

Présenté par : Delwende Noaga Damien Massimbo

Sarra Trabelsi

Ariane Ndong Mba

Supervisé par : **M. Nicolas ASIN-HILAIRE**

Année : 2020 - 2021

Table des matières

1 Introduction **3**

2 Traitement des données **3**
2.1 Présentaton du jeu de données 3
2.2 Traitement des données manquantes 4
2.3 Recodage des variables 5
2.3.1 Statistique descriptive 6

3 Module Cartographie **9**
3.1 Geopandas 9
3.2 Map de la variable room_type 12

4 Modélisation **12**
4.1 Séparation de la base de données 12
4.2 Implémentation des méthodes 12
4.2.1 Choix du meilleur modèle 13
4.3 Prédiction 13
4.4 Variables Influentes 14
4.5 Amélioration des points 14

5 Conclusion **15**

Références **16**

1 Introduction

Durant ce projet,nous allons étudier un jeu de donnée sur les prix de location d'appartements à Beijing mis en ligne par les utilisateurs sur AIRBNB. En effet, cette base associe aux différentes caractéristiques d'un appartement, un prix. L'objectif principal ici est de prédire dans quelle classe de prix va se trouver l'appartement à louer et non pas directement le prix. Il s'agit ainsi d'une problématique de classification. Pour cela, différentes méthodes de classification, abordées en cours de python seront mises en avant. Et par la suite, nous déciderons quel modèle à retenir grâce à des critères de sélection et d'ajustement.

2 Traitement des données

2.1 Présentaton du jeu de données

Notre jeu de données contient **27439 individus** pour **12 variables**. Ci-dessous, nous en avons une idée :

	id	host_id	room_type	neighbourhood	accommodates	bedrooms	bathrooms	beds	price	name	latitude	longitude
0	114384	533062	Entire home/apt	NaN	2	1.0	NaN	1.0	628.00	CBD Luxury 1-bedroom suite with a 30m2 terrace	39.90474	116.46372
1	114465	533062	Entire home/apt	Beijing, China	2	1.0	NaN	1.0	614.00	国贸CBD中国尊梵悦108附近豪华总统套房睡3人/步行五分钟至地铁站万达广场沃尔玛	39.90441	116.46524
2	128496	467520	Entire home/apt	NaN	3	1.0	NaN	2.0	388.00	Heart of Beijing: House with View 2	39.93235	116.42254
3	161902	707535	Entire home/apt	NaN	2	1.0	NaN	1.0	552.00	cozy studio in center of Beijing	39.93357	116.43577
4	162144	707535	Entire home/apt	NaN	4	1.0	NaN	2.0	600.00	nice studio near subway, sleep 4	39.93668	116.43798
5	279078	1455726	Entire home/apt	NaN	2	1.0	NaN	1.0	400.00	Nice Apartment in Beijing	39.93919	116.43440
6	287026	1456491	Entire home/apt	NaN	3	1.0	NaN	1.0	418.00	Studio in downtown Beijing #2	39.94115	116.44122

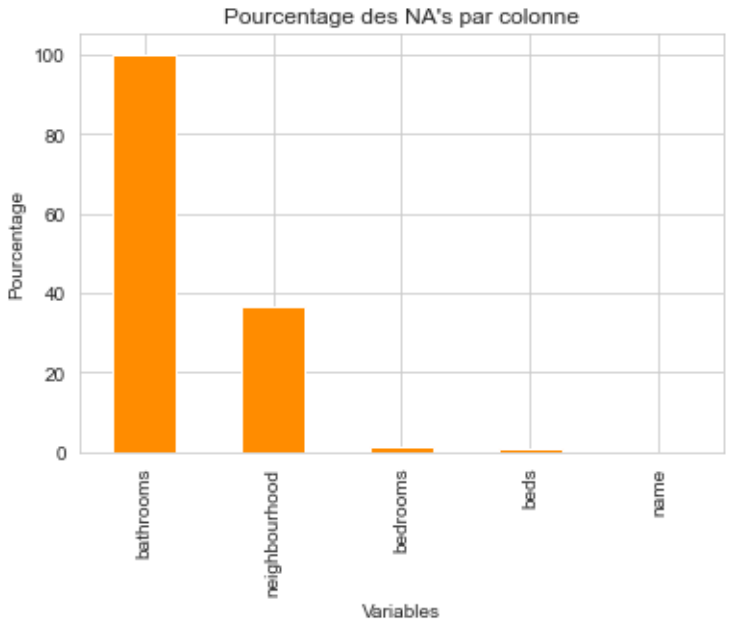
Il est important de noter les types de données que nous utilisons et nous devrons les ajuster au moment de la modélisation :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27439 entries, 0 to 27438
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             27439 non-null  int64
 1   host_id        27439 non-null  int64
 2   room_type      27439 non-null  object
 3   neighbourhood  17455 non-null  object
 4   accommodates   27439 non-null  int64
 5   bedrooms       27040 non-null  float64
 6   bathrooms      0 non-null      float64
 7   beds           27180 non-null  float64
 8   price          27439 non-null  float64
 9   name           27438 non-null  object
 10  latitude       27439 non-null  float64
 11  longitude      27439 non-null  float64
dtypes: float64(6), int64(3), object(3)
memory usage: 2.5+ MB
```

Comme nous pouvons le constater sur l'aperçu du jeu de données, on a la présence de plusieurs valeurs manquantes sur lesquelles nous allons un peu plus nous pencher.

2.2 Traitement des données manquantes

Cinq de nos dix variables contiennent des données manquantes à savoir **bathrooms, neighbourhood, bedrooms, beds** et **name**. Elles n'apportent, toutes, pas de grandes modifications dans l'information de base sauf pour la variable **bathrooms** dans laquelle il n'y a que des DM [1]. Nous allons la supprimer plus loin.



Pour les autres variables, nous allons choisir une méthode d'imputation selon leur type. En effet, nous remplacerons les DM des variables quantitatives par la médiane de chacune

1. Données manquantes

d'elle respectivement et pour une variable qualitative, nous utiliserons le mode comme méthode d'imputation avec le code ci-dessous :

```
#imputation des DM
for col in df_beijing.columns:
    print(col,': ',df_beijing[col].isnull().sum())
df_beijing.loc[:,"bedrooms"]=df_beijing.loc[:,"bedrooms"].fillna(df_beijing.
    loc[:,"bedrooms"].median())
df_beijing.loc[:,"beds"]=df_beijing.loc[:,"beds"].fillna(df_beijing.loc[:,"
    beds"].median())
df_beijing.loc[:,"neighbourhood"]=df_beijing.loc[:,"neighbourhood"].fillna(
    df_beijing.loc[:,"neighbourhood"].mode()[0])
```

2.3 Recodage des variables

Dans cette partie, nous allons effectuer plusieurs changements sur notre base de données. D'abord, nous recodons la variable **price**, qui est une variable quantitative, en classes afin de simplifier l'étape de modélisation. Le but est de prédire dans quelle classe de prix sera le prix de location de l'appartement.

La méthode la plus simple pour créer ces classes est d'utiliser les quantiles comme valeur de seuils :


```
# Creation de la variable 'Class_prix'
Class_prix = []
for prix in df_beijing2.price:
    if prix<np.quantile(df_beijing2["price"],0.20):
        classe = 'Classe0'
    elif (prix<np.quantile(df_beijing2["price"],0.50))&(prix>=np.quantile(
    df_beijing2["price"],0.20)):
        classe = 'Classe1'
    elif (prix<np.quantile(df_beijing2["price"],0.80))&(prix>=np.quantile(
    df_beijing2["price"],0.50)):
        classe = 'Classe2'
    else:
        classe = 'Classe3'
    Class_prix.append(classe)

df_beijing2["target"] = Class_prix
```

Nous allons maintenant supprimer les variables **bathrooms, id, host_id** et **names** que nous ne jugeons pas importantes pour notre analyse.

```
v=['bathrooms','id','host_id','name']
df_beijing.drop(columns= v, inplace =  True)
```

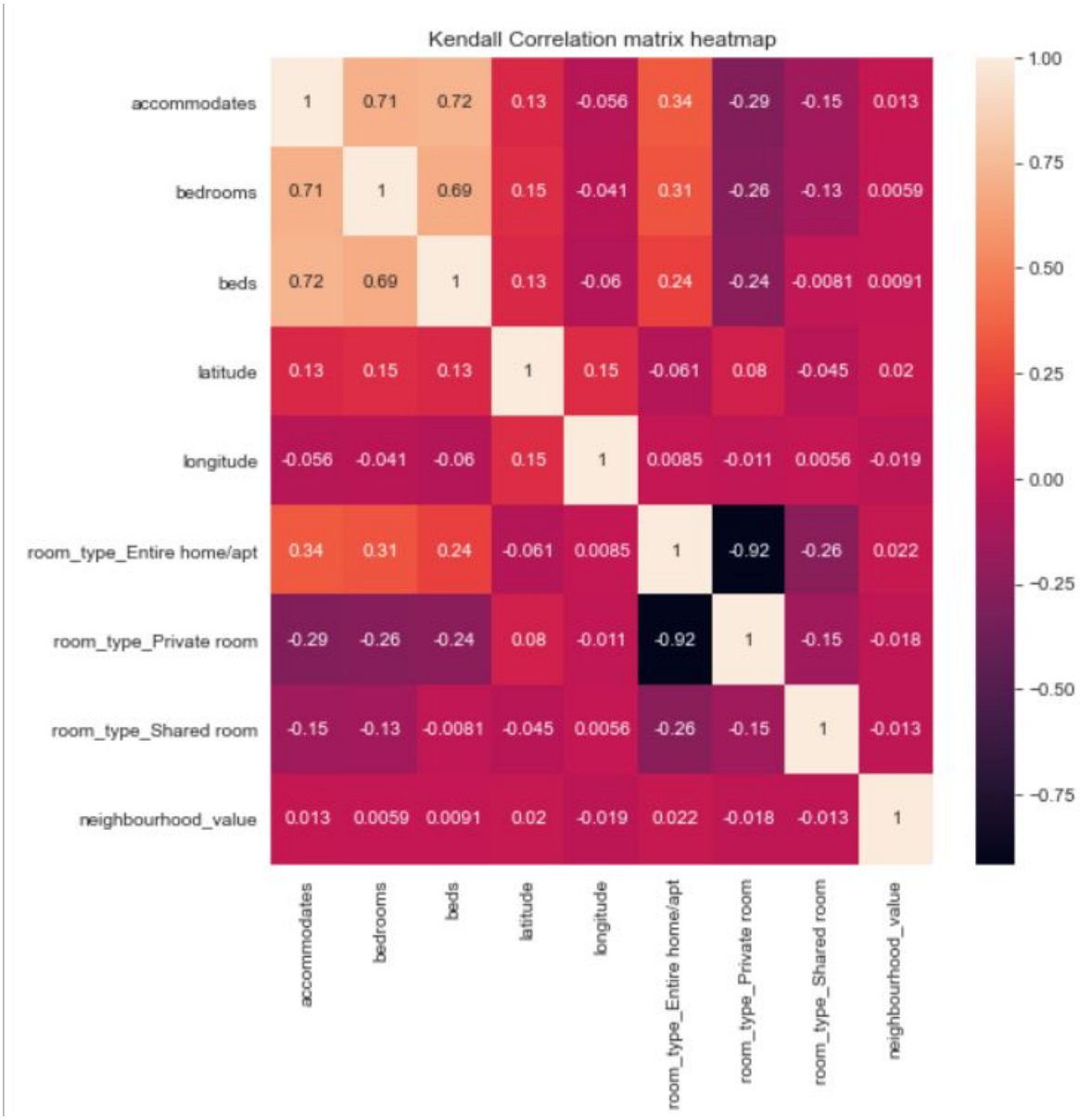
Nos dernières modifications consisteront à dummifier la variable **room_type** et à indexer la variable **neighbourhood**. Concernant cette dernière, nous pourrions normalement la supprimer car intuitivement ses informations sont essentiellement contenues dans les variables **latitude** et **longitude** mais nous allons la conserver car elles sont plus précises.

```
#dummification de la variable 'room_type'
df_beijing1=df_beijing.copy()
df_beijing1=pd.get_dummies(df_beijing, columns=['room_type'])

#indexation de la variable 'neighbourhood'
from sklearn import preprocessing
le= preprocessing.LabelEncoder()
le.fit(df_beijing2['neighbourhood'])
le.transform(df_beijing2['neighbourhood'])

df_beijing2['neighbourhood_value'] = le.transform(df_beijing2['neighbourhood
    '])
```

2.3.1 Statistique descriptive

Ci-dessous, un bref résumé descriptif des variables quantitatives de notre nouvelle table :

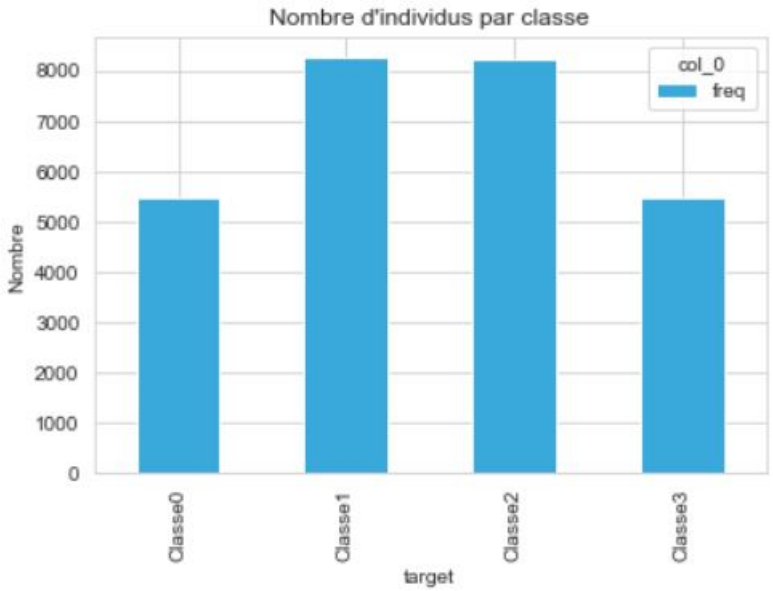
	accommodates	bedrooms	beds	latitude	longitude	room_type_Entire home/apt	room_type_Private room	room_type_Shared room	neighbourhood_value
count	27439.000	27439.000	27439.000	27439.000	27439.000	27439.000	27439.000	27439.000	27439.000
mean	4.018	1.786	2.409	40.055	116.449	0.613	0.347	0.040	2.055
std	3.473	1.688	2.960	0.264	0.293	0.487	0.476	0.196	1.000
min	1.000	1.000	0.000	39.456	115.475	0.000	0.000	0.000	0.000
25%	2.000	1.000	1.000	39.897	116.334	0.000	0.000	0.000	2.000
50%	2.000	1.000	2.000	39.942	116.434	1.000	0.000	0.000	2.000
75%	4.000	2.000	2.000	40.161	116.531	1.000	1.000	0.000	2.000
max	16.000	50.000	70.000	40.950	117.494	1.000	1.000	1.000	26.000



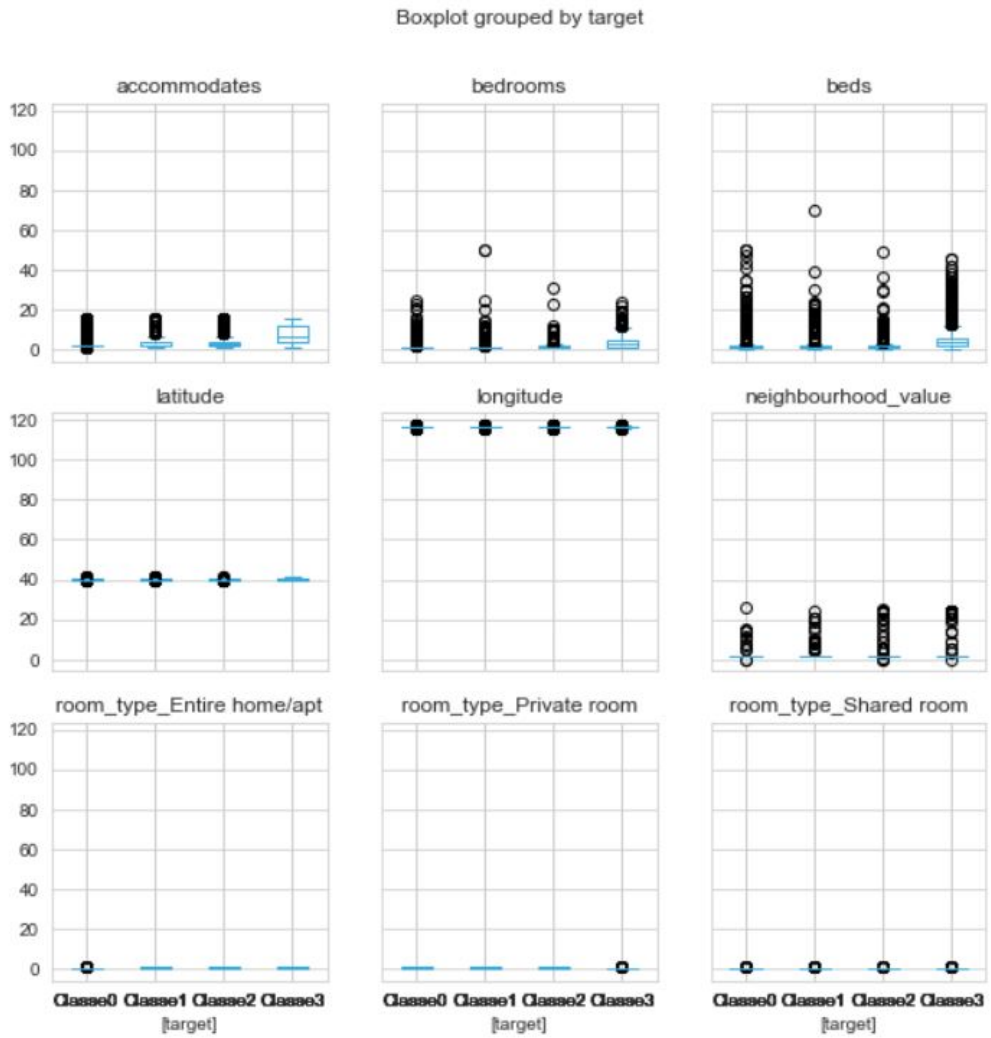
On remarque les variables **accommodates** et **bedrooms** et **beds** sont fortement corrélées positivement. Cela voudrait dire que si l'un augmente, les autres aussi et inversement. On remarque qu'on a plusiers variables corrélées négativement avec des coefficients très bas. En général,il n'y a pas beaucoup de corrélation entre nos variables.

Intéressons nous maintenant à notre variable **ClassPrix** :

target	col_0 freq
Classe0	5463
Classe1	8254
Classe2	8234
Classe3	5488



Dans notre base, on a plus d'appartements dans la classe de prix 1, suivie des classes de prix 2, 3 et 4. Visualisons maintenant les relations de notre variable **Class_prix** avec les autres variables :



On remarque les prix des chambres pouvant acceuillir plus de personnes, ayant plus de chambres et de salles de bains sont en majorité dans la classe 3. Les latitudes et longitudes tournent respectivement, essentiellement autour 40 et 118.

3 Module Cartographie

3.1 Geopandas

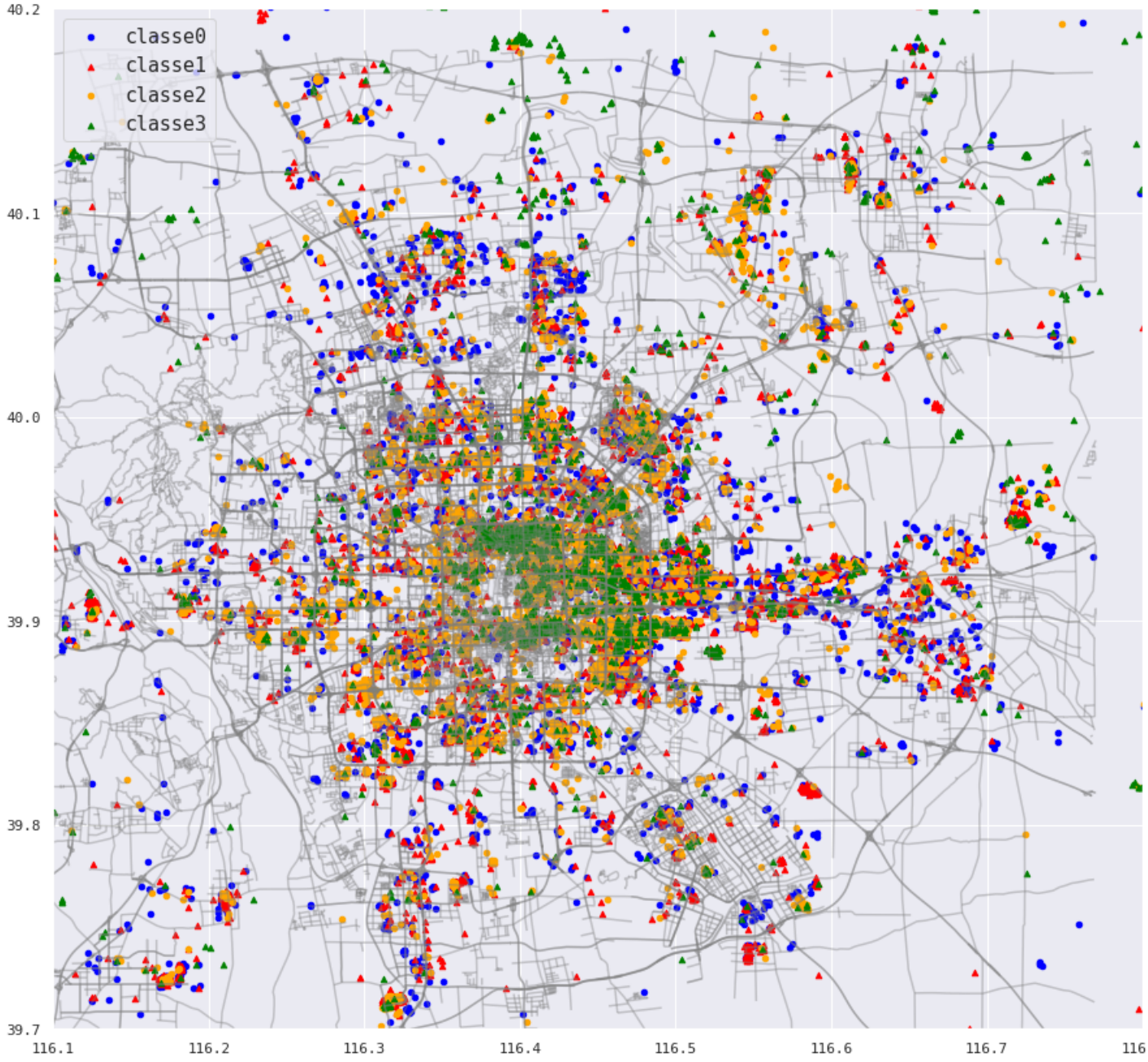
Nous pouvons visualiser à travers plusieurs cartes, la répartition des appartements en fonction de leur caractéristiques. Nous allons considérer certaines variables de notre table de base.

Nous avons téléchargé un fichier de forme (extension shp) .Il suffit de lire notre shape-file avec GeoPandas et d'utiliser matplotlib pour le tracer de cette manière :

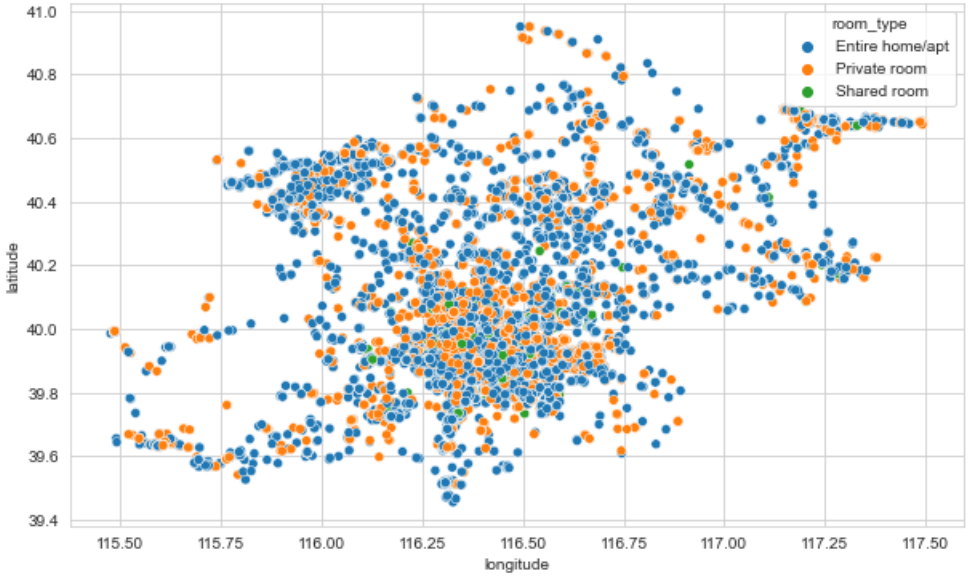
On remarque que notre carte montre les lignes centrales des rues de Beijing :

Comme nous disposons des informations sur la latitude et la longitude, nous pouvons créer des points. Un point est essentiellement un objet unique qui décrit la longitude et la latitude d'un point de données.

GéoPandas référencera automatiquement la colonne "géométrie" lorsqu'on trace nos données. Pour ce faire, il suffit de superposer nos données sur la carte que nous avons tracée ci-dessus.


classe0
classe1
classe2
classe3
40.2
40.1
40.0
39.9
39.8
39.7
116.1
116.2
116.3
116.4
116.5
116.6
116.7
116.8

3.2 Map de la variable room_type



Nous observons ci-dessus,la répartition des logements selon selon la sous-région ou le quartier de la ville considérée mais aussi selon le type de chambre (variable que nous avons dummifier). On remarque qu'avec une grande majorité, les logements se situe à Beijing et sont soit des appartements/maisons entiers et des chambres privées.On observe une zone de concentration des logements dont les prix sont dans les classes 0 et 1. Ceux des classes 3 sont un peu en périphérie.

4 Modélisation

A présent, nous allons passer à l'étape de la modélisation. C'est ici que nous allons prédire dans quelle classe de prix va se trouver l'appartement.

4.1 Séparation de la base de données

On a séparé notre base de données de la manière suivante :

```
#Division de la base de donn es en base de test et base d'entrainement
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state=1994)
```

On crée un ensemble d'apprentissage contenant 70% des observations et un ensemble de validation (ensemble de tests) contenant les observations restantes.

4.2 Implémentation des méthodes

Dans cette partie,nous avons implémenté plusieurs méthodes :**Random Forest Classifier** **Gradient Boosting Classifier**, **Regression Logistique (solver='lbfgs',multi_class='multinomial')** (Régression Multinomiale ou Polytomique) et **ExtraTreesClassifier**. Ensuite,nous avons optimisé chacun des paramètres de ces méthodes afin d'obtenir le meilleur modèle.Pour enfin, retenir

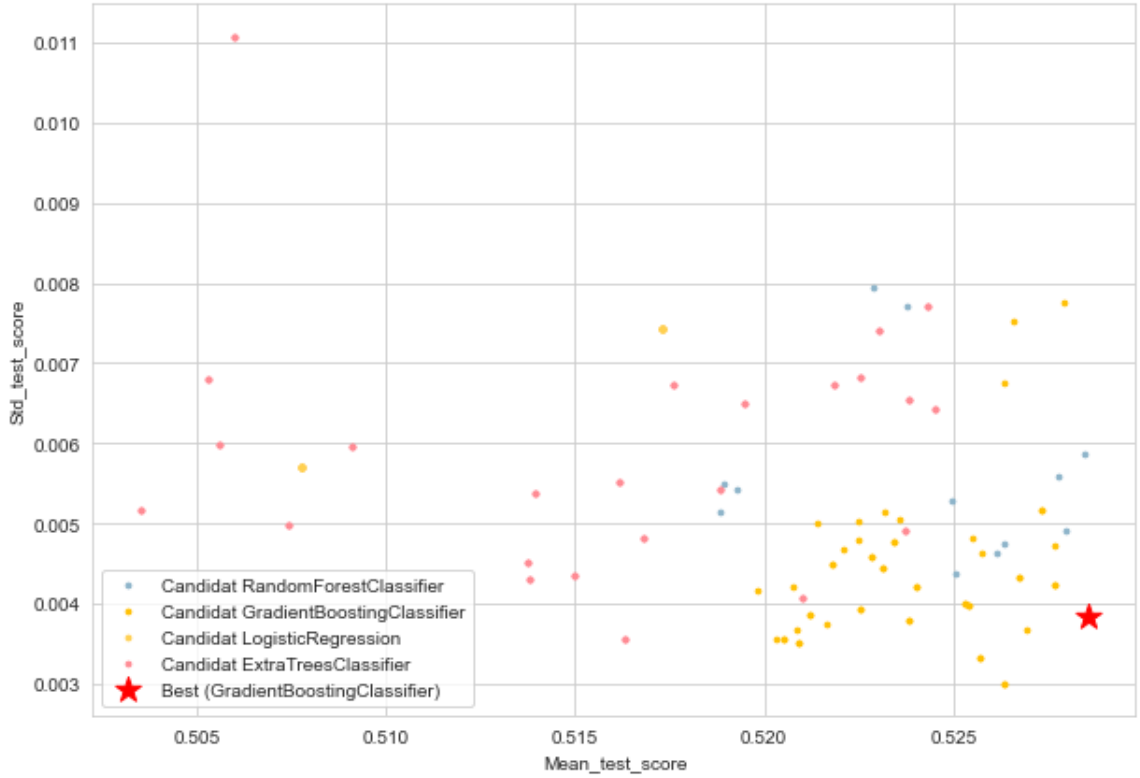
le meilleur modèle parmis d'autres modèles.La technique la plus rapide sera d'utiliser la fonction **GridSearchCV**.On détermine le meilleur candidat i.e. celui qui minimise la variance et maximise le score f1 sur des échantillons de test.

4.2.1 Choix du meilleur modèle

Ci-dessous, les cind modèles que nous avons implémenté :

	algo	mean_train_score	std_train_score	mean_test_score	std_test_score	params
17	GradientBoostingClassifier	0.538189	0.002110	0.528558	0.003847	{'max_depth': 4, 'max_features': 'sqrt', 'n_es...
20	GradientBoostingClassifier	0.538189	0.002110	0.528558	0.003847	{'max_depth': 4, 'max_features': 'log2', 'n_es...
5	RandomForestClassifier	0.538137	0.002659	0.528454	0.005872	{'class_weight': None, 'max_depth': 7, 'n_esti...
3	RandomForestClassifier	0.536862	0.002448	0.527985	0.004906	{'class_weight': None, 'max_depth': 7, 'n_esti...
12	GradientBoostingClassifier	0.532488	0.002282	0.527933	0.007759	{'max_depth': 4, 'max_features': 'auto', 'n_es...

On représente graphiquement l'ensemble de nos estimateurs avec en abscisse **mean_test_score** et en ordonnée **std_test_score** et une échelle de couleur permettant de différencier le type d'algorithme utilisé.



On voit d'après les deux dernières sorties que le modèle sélectionné est le **Gradient Boosting Classifier**.

4.3 Prédiction

Disposant du meilleur modèle nous allons prédire dans quelle classe de prix va se trouver un nouvel appartement. Afin de mesurer la performance de ce modèle nous affichons sa matrice de confusion pour voir comment est faite la classification grâce aux échantillons test :

```
#matrice de confusion
from sklearn.metrics import confusion_matrix
```

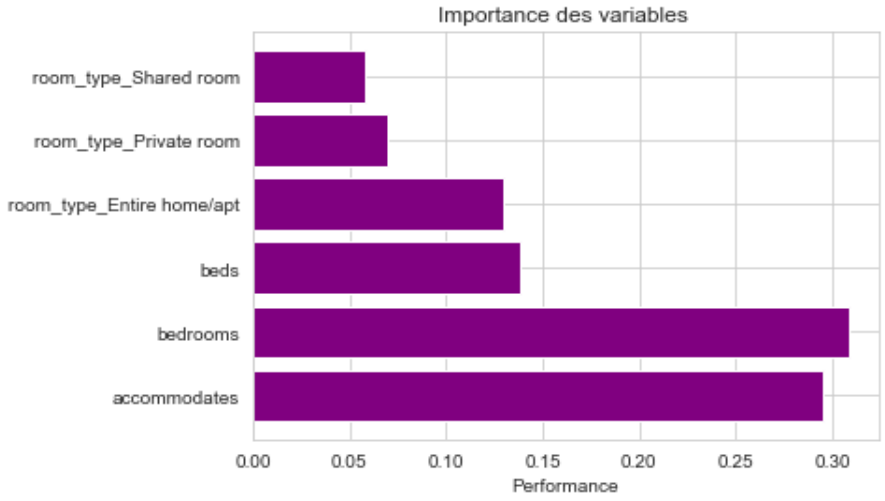
```
confusion_matrix(y_test, classifier.predict(X_test))
```

Voici la matrice de confusion trouvée :

$$\begin{pmatrix} 1179 & 291 & 118 & 45 \\ 607 & 1298 & 538 & 39 \\ 458 & 842 & 1035 & 173 \\ 173 & 153 & 383 & 900 \end{pmatrix}$$

4.4 Variables Influentes

Disposant du meilleur modèle regardons l'importance des variables.



On remarque que les variables influentes sont **accommodates**,**bedrooms**, **beds** et **room_type** pour la modalité **Entire_home/apt**.

4.5 Amélioration des points

Nous aurions pu obtenir uniquement le GradientBoosting-Classifier pour les cinqs meilleurs modèle regroupés dans le data-frame en jouant sur les paramètres que peut prendre le **make_scorer**. Le paramètre **average='weighted'** est pondéré car les scores de classe F-1 sont moyennés en utilisant le nombre d'instances dans une classe comme poids. On pourrait donc s'intéresser à **average='weighted'** car la variable target constituée des classes de prix ont des effectifs bel et bien différents.

```
from sklearn.model_selection import GridSearchCV
scorer = make_scorer(f1_score, average='weighted')
res={}
for model_class in Models_list:
    print(str(model_class['model']).split('(')[0])
    model_opt=GridSearchCV(estimator=model_class['model'],param_grid=
   model_class['parameters'], cv=5, scoring =scorer, return_train_score=True
   ).fit(X_train, y_train)
```

```
7     res.update({str(model_class['model']).split('(')[0] : model_opt.
    cv_results_})
```

	algo	mean_train_score	std_train_score	mean_test_score	std_test_score	params
67	GradientBoostingClassifier	0.542477	0.002823	0.529883	0.010843	{'max_depth': 7, 'max_features': 'auto', 'n_es...
59	GradientBoostingClassifier	0.537148	0.003014	0.529830	0.012562	{'max_depth': 5, 'max_features': 'auto', 'n_es...
66	GradientBoostingClassifier	0.541229	0.003196	0.529805	0.011444	{'max_depth': 7, 'max_features': 'auto', 'n_es...
61	GradientBoostingClassifier	0.534468	0.003938	0.529586	0.010900	{'max_depth': 5, 'max_features': 'sqrt', 'n_es...
64	GradientBoostingClassifier	0.534468	0.003938	0.529586	0.010900	{'max_depth': 5, 'max_features': 'log2', 'n_es...

On aurait pu aussi imputer les valeurs manquantes en utilisant des méthodes telles que le GradientBossting ou la méthode d'Adaboost. On aurait pu faire intervenir la variable **neighbourhoods** dans notre analyse.

5 Conclusion

Ce projet nous a bel et bien permis d'aller plus en profondeur sur les différentes notions abordées en cours et egalement, nous avons appris à bien se familiariser avec la manipulation des outils de machine learning. Dans l'optique de se ramener à un problème de classification nous avons d'abord recoder la variable d'intérêt. Ensuite, nous avions effectuer une statistique descriptive et une exploration de notre variable d'intérêt à travers les barplots et une carte qui illustre la repartition des logements en fonction du prix recodé en classe dans la ville de Beijing. Aussi, nous avons traité nos données manquantes, ce qui a abouti à la phase de modélisation en testant ainsi plusieurs algorithmes de machine learning. On constate que le meilleur modèle qui a été retenu est le GradientBoosting-Classifier. Enfin, en prédisant la classe de prix d'appartenance des logements dans la ville de Beijing, nous avons aboutit à une précision pas tellement élevée.

Références

[1] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

[2] https://scikit-learn.org/stable/modules/feature_selection.html.

[3] https://scikit-learn.org/stable/index.html.

[4] https://towardsdatascience.com/geopandas-101-plot-any-data-with-a-latitude-and-longitude