

Projet Apprentissage Automatique : Prédiction des prix des Airbnb à Berlin

Amine Larhchim, Daniel Lévy, Juliette Leorat, Pierre-Alexandre Klein, Agathe Gioan

Sommaire

- 1. Exploration des données et Visualisation
- 2. Processing
 - 2.1. Pre-processing
 - 2.2. Feature Engineering
 - 2.3. Missing values et Outliers
 - 2.4. Scaling
- 3. Réduction de dimension
- 4. Modèles
 - 4.1. Régression Linéaire
 - 4.2. XGBoost
 - 4.3. Random Forest
- 5. Résultats et conclusions

1. Exploration des données et Visualisation

Nous avons commencé le projet par importer les données et les explorer pour les comprendre et découvrir les différentes features. Nous avons regardé la liste des colonnes du dataset et nous avons d'abord listé les features qui n'avaient pas d'intérêt pour le problème pour pouvoir les supprimer (Listing Name, Host Name, City, Country, User ID). Nous sommes intéressés aux colonnes restantes à Host ID, City, Country, User ID, qui sont les plus pertinentes pour la prédiction du prix.

Les visualisations suivantes vont faire apparaître des informations pertinentes sur les features :

Grâce à une fonction que nous avons créée, nous avons d'abord pu afficher le nombre de données manquantes par colonne et le pourcentage correspondant.

```
In [5]: missing(df)
host_since => 21 [0.13%]
host_response_time => 7075 [45.11%]
host_response_rate => 7249 [46.22%]
is_superhost => 23 [0.14%]
property_type => 15 [0.48%]
accommodates => 19 [0.12%]
bathrooms => 25 [0.16%]
beds => 20 [0.13%]
square_feet => 15380 [98.07%]
guests_included => 1 [0.06%]
min_nights => 1 [1.59%]
first_review => 2698 [17.28%]
last_review => 2699 [17.21%]
overall_rating => 2955 [18.84%]
accuracy_rating => 2964 [18.9%]
cleaning_rating => 2963 [18.89%]
checkin_rating => 2966 [18.91%]
communication_rating => 2963 [18.89%]
location_rating => 2964 [18.9%]
value_rating => 2965 [18.91%]
```

Certaines colonnes ont des données manquantes, en particulier l'information sur la surface des appartements qui aurait été intéressante, mais qui montre plus de 98% de données manquantes.

Voici ensuite la corrélation entre les différentes variables :

```
In [7]: plt.figure(figsize=(14,8))
sns.heatmap(corrMatrices, annot=True)

Out[7]: <AxesSubplot: x=ticks, y=ticks>

```

Pour la corrélation, on identifie deux grands groupes dont les variables sont très corrélées entre elles. Le premier groupe est celui des variables relatives à la capacité (nombre de chambre, surface, nombre de salles de bain...) qui sont assez bien corrélées au prix. L'autre groupe est celui des variables relatifs au jugement des occupants (accuracy_rating, cleaning_rating, value_rating...) qui elles ne sont pas corrélées au prix.

Pour vérifier que les données faisaient sens, nous avons également visualisé la répartition des types de Airbnb proposés et des super_host, la moyenne des prix par type de chambre, la moyenne des prix selon le quartier de Berlin.

```
In [8]: viz(df)
```

Sans surprise, un appartement est en moyenne plus cher qu'une chambre privée ou partagée. Certains quartiers sont plus chers en moyenne, comme Mitte, Pankow ou Charlottenburg-Wilm. Mitte est le centre de la ville, Pankow est un quartier résidentiel et Charlottenburg est connu pour ses jardins et ses monuments. Cette observation nous a amené à créer de nouvelles features (cf section feature engineering) pour prendre en compte la distance des Airbnb aux principaux quartiers.

Le type de propriété n'est pas très réparti puisqu'une grande majorité des biens sont des appartements. Ce serait donc intéressant de ne garder que les propriétés les plus représentatives : Appartement, Condominium, maison et loft, et de labelliser les autres comme 'others', ce qui diminuerait la complexité du problème.

En moyenne, un super_host est plus cher, ce qui pourrait être du au fait que les gens font plus confiance à ces hôtes et acceptent donc de payer plus, ou le fait que ces hôtes ont présenté des biens de meilleure qualité et donc plus cher pour devenir super_host.

Nous avons visualisé la répartition des prix par nuit des Airbnb pour détecter d'éventuels outliers. En effet, grâce à ces graphes, on voit que 75% des valeurs de prix sont en dessous de 70€ la nuit, et qu'au dessus de 300€ par nuit il y a peu de valeurs, que nous pouvons considérer comme des outliers.

```
In [10]: plot_mse(df)
```

Number of listings for price range

Nous avons voulu voir si la différence de prix selon la localisation était visible, ce qui n'a pas été concluant à cause du nombre de point et du large éventail de prix.

```
In [13]: df.plot.scatter(x="latitude", y="longitude", figsize=(10,10), c="price", cmap='jet')
```

```
Out[13]: <Figure>
```

Random Forest Price Prediction

Enfin, nous avons voulu voir si il y avait une saisonnalité dans les nombres d'hôtes et les revues, on voit un pic chaque année au milieu (l'été) et un creux au début d'année.

```
In [15]: decompose_time_series(ts_host_since, title='Number of hosts joining Airbnb each month')
```

Number of hosts joining Airbnb each month

2. Processing

2.1. Pre-processing

Grâce à notre exploration des données et à la visualisation, nous avons pu travailler sur les données que nous avions pour les rendre utilisables par nos modèles.

Nous avons mis au bon format toutes les colonnes pour introduire une distance à différents centres de Berlin en recherchant la localisation des emplacements importants de la ville.

2.2. Feature Engineering

Il y avait dans les features le neighbourhood group des Airbnb, mais également leur localisation (latitude et longitude). Nous avons décidé d'utiliser ces dernières données pour introduire une distance à différents centres de Berlin en recherchant la localisation des emplacements importants de la ville.

2.3. Missing values et Outliers

Nous avons commencé par réaliser une fonction afin de connaître le pourcentage de valeurs manquantes pour chaque colonnes.

Grâce à ces résultats nous avons décidé de supprimer la colonne "Square Feet" dans laquelle il manquait 98% des données. De plus, le métrage est une information rarement disponible sur Airbnb.

Nous avons également supprimé les colonnes qui avaient peu d'intérêt : Business Travel Ready, Host id, listing id et neighborhood group (redondant par rapport à neighbourhood).

Nous avons aussi décidé de supprimer les 9 lignes qui n'avaient pas de prix renseigné, ainsi que toutes les lignes où il y avait plus 3 valeurs manquantes, ce qui faisait 3045 lignes.

Grâce aux visualisations que nous avons faites, nous avons également vu qu'une très grande proportion des prix se situent en dessous de 300€ la nuit. Seules quelques valeurs étaient supérieures, nous avons donc décidé d'également supprimer les lignes dont le prix était supérieur à 200€/nuit, ce qui correspondait à 66 lignes.

Pour les autres colonnes, nous avons commencé par utiliser la médiane sur les valeurs numériques et la valeur la plus fréquente pour les colonnes catégorielles. En testant des modèles avec ces imputations, nous nous sommes rendus compte rapidement que ce n'était pas les méthodes les plus efficaces (sur ce modèle).

Il a été donc décidé de réaliser un KNN imputer afin d'utiliser les plus proches voisins des valeurs manquantes pour compléter ces dernières. Pour déterminer le paramètre k, nous avons testé un RandomForestRegressor sur des datasets imputé avec différentes valeurs de k afin de pouvoir choisir celle qui optimise la rmse.

2.4. Scaling

Nous avons également voulu scalar les données pour essayer d'améliorer la performance des modèles que nous avons testés. Pour les valeurs numériques, nous avons testé le MinMaxScaler ainsi que le StandardScaler. Nous avons appliqués nos modèles avec les différents Scaler (voir section modèles), et nous nous sommes rendus compte que le choix du scaler avait peu d'impact sur la rmse et le r2 des modèles. L'intérêt du scaling dépendait également des modèles utilisés. Il s'est avéré utile pour l'amélioration de la performance de la linear regression par exemple, mais inutile pour les modèles de type Decision Trees, XGBoost ou autres modèles ensemble.

Finalement nous avons testé un StandardScaler et MinMaxScaler pour les modèles nécessitant un scaling sans vraiment de différences notables au niveau de la performance.

En ce qui concerne les données catégorielles nous avons l'embaras du choix pour encoder nos variables. On a choisi le simple OneHotEncoding car le nombre de colonnes est raisonnable. On spécifie bien l'argument drop_first=True dans la fonction get_dummies de python pour n'avoir que n-1 variables qui encodent les n catégories différentes.

3. Réduction de dimension

Nous avons fait une Subset Selection pour essayer de réduire les dimensions de notre dataset et réduire la complexité du problème. Cependant, la taille de notre dataset ne pose pas vraiment de problèmes de dimensions, nous avons donc fourni le code dans notre script de processing mais avons décidé de ne pas l'utiliser par la suite.

4. Modèles

Nous avons testé plusieurs modèles de régression disponibles dans la bibliothèque Scikit-Learn et nous les avons classés selon la rmse. Une comparaison par GridSearch Validation nous a permis d'évaluer les scores et de sélectionner les meilleurs modèles.

Les résultats sont les suivants :

```
In [20]: print ('mse = {}, rmse = {} \n mae = {} \n r2 = {}'.format(mse,math.sqrt(mse), mae, r2))
```

```
mse = 819.3711044765383, rmse = 28.624659028127102
mae = 19.0555140609367 r2 = 0.5093809517796777
```

```
In [21]: plotGraph(y_test, y_pred_lr, "Linear Regression Price Prediction")
```

```
Out[21]: <Figure>
```

Linear Regression Price Prediction

Nous avons voulu voir si la différence de prix selon la localisation était visible, ce qui n'a pas été concluant à cause du nombre de point et du large éventail de prix.

```
In [13]: df.plot.scatter(x="latitude", y="longitude", figsize=(10,10), c="price", cmap='jet')
```

```
Out[13]: <Figure>
```

```
<Figure>
```

En utilisant ces résultats, nous avons obtenu les résultats suivants :

```
mse = 804.825496216108, rmse = 28.39946526432412
mae = 18.69619070870873 r2 = 0.5180905004097544
```

```
In [22]: df.plot.scatter(x="latitude", y="longitude", figsize=(10,10), c="price", cmap='jet')
```

```
Out[22]: <Figure>
```

```
<Figure>
```

Nous avons réalisé un Random Forest avec ces meilleures paramètres pour éviter un GridSearch coûteux en temps et en calculs. Voici les résultats obtenus :

```
mse = 806.2002672145373, rmse = 28.39366596997537
mae = 18.709841057427 r2 = 0.517267321719881
```

```
In [23]: Image(path+path_image2, width=500, height=500)
```

```
Out[23]: <Figure>
```

XGBoost Price Prediction

Nous avons également obtenu le graphique des valeurs prédictives vs les valeurs réelles :

```
In [24]: Image(path+path_image3, width=500, height=500)
```

```
Out[24]: <Figure>
```

```
<Figure>
```

Nous obtenons alors d'après le meilleur score rmse :

```
mse = 804.825496216108, rmse = 28.39946526432412
mae = 18.69619070870873 r2 = 0.5180905004097544
```

```
In [25]: Image(path+path_image4, width=500, height=500)
```

```
Out[25]: <Figure>
```

```
<Figure>
```

En utilisant ces résultats, nous avons obtenu les résultats suivants :

```
mse = 806.2002672145373, rmse = 28.39366596997537
mae = 18.709841057427 r2 = 0.517267321719881
```

```
In [26]: Image(path+path_image5, width=800, height=800)
```

```
Out[26]: <Figure>
```

```
<Figure>
```

Enfin, nous avons voulu voir si il y avait une saisonnalité dans les nombres d'hôtes et les revues, on voit un pic chaque année au milieu (l'été) et un creux au début d'année.

```
In [15]: decompose_time_series(ts_host_since, title='Number of hosts joining Airbnb each month')
```

Number of hosts joining Airbnb each month

2. Processing

2.1. Pre-processing

Grâce à notre exploration des données et à la visualisation, nous avons pu travailler sur les données que nous avions pour les rendre utilisables par nos modèles.

Nous avons mis au bon format toutes les colonnes (changement de virgules par des points, suppression des espaces, convertir les dates, remplacer les * par des NaN). Nous avons également encodé les valeurs catégorielles que nous avions.

2.2. Feature Engineering

Il y avait dans les features le neighbourhood group des Airbnb, mais également leur localisation (latitude et longitude). Nous avons décidé d'utiliser ces dernières données pour introduire une distance à différents centres de Berlin en recherchant la localisation des emplacements importants de la ville.

2.3. Missing values et Outliers

Nous avons commencé par réaliser une fonction afin de connaître le pourcentage de valeurs manquantes pour chaque colonnes.

Grâce à ces résultats nous avons décidé de supprimer la colonne "Square Feet" dans laquelle il manquait 98% des données. De plus, le métrage est une information rarement disponible sur Airbnb.

Nous avons également supprimé les colonnes qui avaient peu d'intérêt : Business Travel Ready, Host id, listing id et neighborhood group (redondant par rapport à neighbourhood).

Nous avons aussi décidé de supprimer les 9 lignes qui n'avaient pas de prix renseigné, ainsi que toutes les lignes où il y avait plus 3 valeurs manquantes, ce qui faisait 3045 lignes.

Grâce aux visualisations que nous avons faites, nous avons également vu qu'une très grande proportion des prix se situent en dessous de 300€ la nuit. Seules quelques valeurs étaient supérieures, nous avons donc décidé d'également supprimer les lignes dont le prix était supérieur à 200€/nuit, ce qui correspondait à 66 lignes.

Pour les autres colonnes, nous avons commencé par utiliser la médiane sur les valeurs numériques et la valeur la plus fréquente pour les colonnes catégorielles. En testant des modèles avec ces imputations, nous nous sommes rendus compte rapidement que ce n'était pas les méthodes les plus efficaces (sur ce modèle).

Il a été donc décidé de réaliser un KNN imputer afin d