PROJET MACHINE LEARNING

Prédiction des prix de AirBnb à Berlin

Rayann Bentounes
Clément Blanchoz-–Rhône
Juliette Cochez
Flavien Deseure

Encadrante
Myriam Tami

I - Introduction

L'étude se résume à la prédiction de prix de AirBnB dans la ville de Berlin à partir d'un dataset comportant près de 40 features. Pour cela, nous avons dans un premier temps exploré les données du dataset, avant d'effectuer un preprocessing des features. Puis dans un dernier temps, nous avons testé plusieurs modèles de prédiction afin d'isoler le modèle le plus performant selon différentes métriques.

II - Analyse

1- Analyse Globale

Une première analyse des features nous a permis d'écarter une partie des variables:

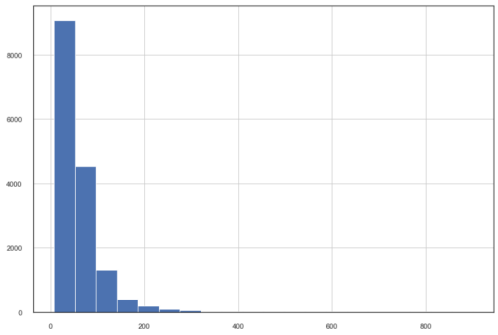
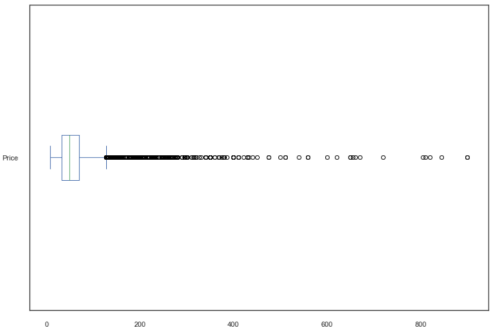
Trop de Nan	Host Response Time, Host Response Rate, Square Feet, features de Ratings
Non pertinentes	Listing ID, Listing Name, Host ID, Host Name, City, Country Code, Country, Business Travel Ready
Information redondante	Postal Code, Neighbourhood Group

Après visualisation des données, on classe les features restantes en fonction de leur corrélation avec *Price*:

Bonne corrélation (corr > 0.4)		Corrélation moyenne (0.4 > corr > 0.1)		Mauvaise corrélation (corr < 0.1)	
Accommodates	0.505	Guests Included	0.368	Is Superhost	0.081
Room Type	0.407	Bathrooms	0.257	Reviews	0.078
Beds	0.421	neighbourhood	0.218	Longitude	-0.042
Bedrooms	0.412	Property Type	0.118	Latitude	0.039
Room Type	0.407	Host Since	-0.168	Instant Bookable	0.040
				Min Night	0.017
				Is Exact Location	0.003

2- Analyse de *Price*

Nous avons ensuite analysé la distribution de la variable cible. On observe une grande concentration des prix vers 50€, avec un maximum à 900€.


```
Price
Mean: 60.34
Median: 49.0
Min: 8.0
Max: 900.0
```

3- Création de nouvelles features

Afin d'obtenir de meilleures corrélation, nous avons créé les features suivantes:

- Nombre total de pièces
- Moyenne de lits par chambre
- Distance au centre de Berlin

Ce qui amène aux corrélations suivantes :

Total Rooms	Average N° of Beds	Distance to center
0.429	0.127	0.052

Les corrélations sont intéressantes pour les variables *Total Rooms* et *Avg Beds,* mais elle est légèrement décevante pour *Distance.*
Cette étude nous permet néanmoins d'enrichir notre modèle.

III - Preprocessing

1- Valeurs manquantes

Nous avons principalement remarqué la présence de deux types de valeurs manquantes :

- Missing At Random (MAR)
- Missing Completely At Random (MCAR)

MAR

Nous avons repéré deux groupes de variables:

- Les ratings avec les dates de reviews qui sont manquantes si le nombre de reviews est égal à 0
- *"Host Response Time"* et "*Host Response Rate*" (même raisonnement)

Dans le set de train, nous avons sélectionné uniquement les variables "*Reviews"* (*pas de valeurs manquantes*) et "*Last Review".*
Pour la variable "*Last Review"*, nous avons imputé une valeur correspondant à une date précise (01-01-01) symbolisant l'absence de reviews, afin d'éviter la perte d'information.

MCAR

Pour les autres variables ainsi que la target, les valeurs manquantes sont peu nombreuses et semblent être simplement des erreurs.
Nous les avons donc simplement supprimées ces valeurs du dataset.

2- Distribution jeux d'entraînement et de test

Nous avons découpé de manière aléatoire notre dataset entre jeu d'entraînement (70% des valeurs) et de test (30% des valeurs). Nous avons ensuite comparé les distributions de la target et de chaque variables en fonction de leur nature :

- Variables numériques discrètes ou variable catégoriques
  + Test de Student (on a supposé la distribution gaussienne des variables)
  + Graphique en bar des deux distributions
- Variables numériques continues
  + Test de Student (on a supposé la distribution gaussienne des variables) en prenant un risque de 5%
  + Histogramme des deux distributions
  + Q-Q plot (graphique représente les quantiles des deux distributions l'un par rapport à l'autre)

Aucune des variables et de la target ne semble différer en distribution entre le jeu d'entraînement et de test.

3- Encodage

Nous avons appliqué cinq encodages différents en fonction des variables :

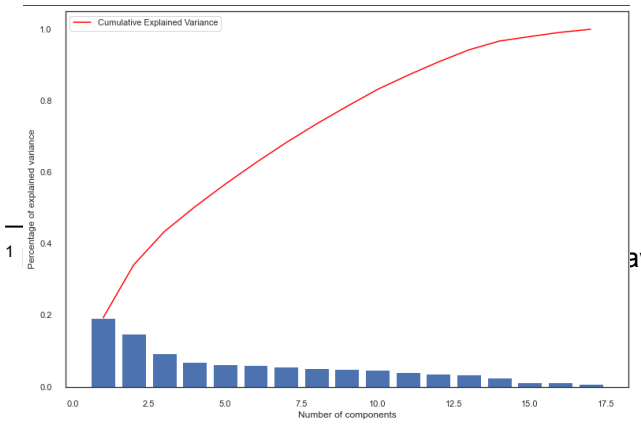
- Un **encodage ordinal** pour les variables catégoriques ordinales ("*Is Superhost"*),
- Un **encodage One Hot** pour les variables catégoriques nominales avec peu de valeurs différentes ("*Room Type"*),
- Un **encodage Leave One Out**[1] avec ajout d'un bruit gaussien pour les variables catégoriques nominales avec beaucoup de valeurs différentes (*"neighbourhood"*, "*Property Type"*),
- Un **encodage numérique** pour les variables numériques qui ont été encodées dans le dataset original comme des objets (*"Accomodates"*, "*Bathrooms"*, "*Bedrooms"*, "*Beds"*, "*Guests Included"*),
- Un **encodage sur les dates** en prenant uniquement l'année (*"Last Review"*).

4- Mise à l'échelle

Nous avons effectué une normalisation de toutes les variables.

5- Analyse PCA

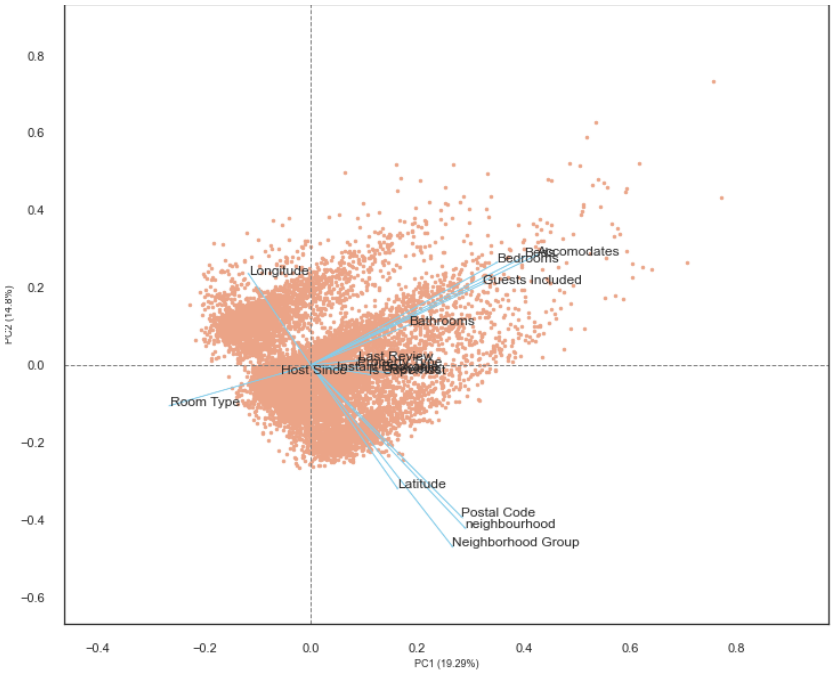
Nous avons alors pu lancer une analyse PCA sur les données mises à l'échelle et observer des résultats intéressants qui ont poussé rétrospectivement à modifier notre dataset.



Nous pouvons voir que nos données sont relativement "éparpillées", le maximum de variance expliquée est de 19% et il nous faut près de 12 composantes pour expliquer 90% de la variance.

1 aveoneout.html

Cependant, en projetant les features sur le plan généré par les deux premières composantes de la PCA, on peut se rendre compte que certaines données sont extrêmement corrélées :



Par exemple on peut voir que les données de “Structure de l'appartement", c’est à dire le nombre de pièce, de salles de bain, de lits etc. sont fortement liées, de même que celles de la “localisation”, donc la latitude, le code postal, ou le neighborhood (qui est identique au code postale). Celà nous confirme le choix de supprimer ces données redondantes.

IV - Modeling

Afin d’obtenir le modèle le plus performant, nous avons utilisé plusieurs approches vues en cours et détaillées dans les parties suivantes. Etant donné que notre modèle se référait à un problème de régression, nous avons choisi la MSE comme étant la principale métrique afin de comparer les performances des différents modèles. Nous avons également affiché d’autres métriques comme le R2, le R2 adjusted et la MAE.

1- Analyse et régression par PCA

L’analyse PCA étant faite, nous n’avions pas de raison de ne pas la tester. Cependant les résultats ne sont pas exceptionnels (mse: entre 1700 et 1500 selon le nombre de features), ce qui n’est pas surprenant puisque il y a très peu de variables par rapport au nombre de

Airbnb, alors que c'est dans le cas contraire que cette approche devient réellement pertinente. Une analyse de performance nous montre même que la réduction de dimension n'a pas d'intérêt puisque le modèle s'améliore systématiquement en augmentant le nombre de features dans notre cas.
Nous avons également essayé d'utiliser le dataset fourni par la PCA pour entraîner les autres modèles, ce qui n'a finalement pas eu un grand intérêt car les résultats obtenus restaient sensiblement identiques.

2- Régression linéaire

L'un des modèles les plus simples, mais qui reste souvent très efficace. C'est pourquoi, il s'agit du premier modèle que nous avons implémenté. A notre grande surprise, les performances étaient plutôt bonnes: la MSE était légèrement inférieure à 1030.

3- SVM

Nous avons entraîné une SVM, mais elle ne donne pas des résultats très satisfaisants, la MSE étant proche de 1200.

4- Decision Tree

Un de nos modèles les plus rapides à entraîner et évaluer (~1.3 sec, soit près de 10x plus rapide que le random forest), qui fournit des résultats corrects (MSE = 1080).

5- Random Forest (et ExtraTrees)

Modèle le plus performant, l'utilisation du grid search nous a donné une estimations des meilleurs hyperparamètres(résultat grid search : max_depth=8, n_estimators=150), mais en tâtonnant nous avons pu voir que nous arrivions à diminuer encore la MSE en en choisissant de légèrement différents (max_depth=10, n_estimators=800).

6- XGBoost

Les performances trouvées se rapprochaient de celles obtenues avec le Random Forest. Néanmoins, même en utilisant de GridSearch pour trouver les meilleurs paramètres aux modèles, nous ne sommes pas parvenus à surpasser la performance du Random Forest.

7- CatBoost

Idem que pour XGBoost.

V - Conclusion

Tableau récapitulatif de nos scores pour le dataset processed:

Modèle	**MSE**	**MAE**	**R2**	**Adj. R2**
Régression Linéaire	1029	20.20	0.434	0.434
SVM	1195	19.88	0.343	0.342

Decision Tree	1080	20.07	0.405	0.405
Random Forest	971	19.42	0.466	0.466
XGBoost	991.7	19.16	0.462	0.449
Gradient boosting	1010	19.50	0.445	0.444
CatBoost	1006	19.41	0.453	0.439
PCA	1774	23.37	0.292	0.291