PROJET 2

Analyse du marché immobilier avec le projet Python :

Enquête sur les transactions immobilières et la satisfaction des clients

Partie 1 : Prétraitement des données

Nettoyage de l'ensemble de données de propriétés

1. **Créer une copie de l'ensemble de données d'origine.** Nous créons une nouvelle variable qui copie les propriétés ensemble de données d'origine. Cette approche permet de se prémunir contre les modifications accidentelles de nos données brutes. Nous n'utiliserons plus directement les données brutes, mais elles resteront accessibles pour référence.

2. **Présentation de l'ensemble de données.** Avec notre ensemble de points de contrôle, nous générons un résumé du contenu de notre ensemble de données immobilières. Ce résumé couvre le nombre, les valeurs uniques et la fréquence de chaque colonne. Nous nous efforçons notamment d'obtenir des informations sur tous les types de données, pas seulement les données numériques.

3. **Évaluer les valeurs manquantes.** Un examen initial indique que toutes les valeurs de notre ensemble de données sont présentes. Mais un examen plus approfondi de la ID colonne suggère qu'elle ne contient peut-être que des données numériques, ce qui justifie une enquête plus approfondie.

4. **Vérification des types de données.** Nous examinons les types de données de chaque colonne pour confirmer nos soupçons sur l'ID colonne. Nous convertissons son type de données en chaîne pour éviter les opérations numériques involontaires sur l'ID série.

5. **Problèmes d'encodage des adresses.** Un problème mineur survient lors du changement du ID type de données ; Pandas ne le reconnaît pas comme un nom de colonne valide. Après enquête, nous avons découvert un problème d'encodage avec une marque d'ordre d'octet (BOM) inutile attachée au nom de la colonne. Pour

remédier à cela, nous renommons la colonne, supprimant ainsi les métadonnées non pertinentes.

6. **Conversion de chaîne.** Nous avons converti avec succès l'ID colonne en type chaîne, résolvant ainsi le problème d'encodage. La conversion du type de données s'étend également aux colonnes building et nombre de propriété, que nous identifions comme des cas similaires à ID.

7. **Vérification finale.** Après la conversion, nous vérifions les nouveaux types de données pour l'ensemble du jeu de données. Mais toutes les colonnes (à l'exception d'area) sont répertoriées comme des objets. Cette découverte inhabituelle nécessite une exploration plus approfondie, que nous reprendrons lors de notre prochaine session.

Nettoyage de l'ensemble de données clients

1. **ID client.** Il y avait un problème d'encodage avec l'ID colonne dans properties. Le nom de la première colonne de customers raw_data est également compromis. C'est pourquoi nous devons renommer la colonne en supprimant la marque d'ordre d'octet (BOM) inutile qui lui est attachée.

2. **Valeurs manquantes.** Vérifiez les valeurs manquantes et observez s'il y en a. Nous les traiterons dans les étapes suivantes. Nous devons mapper les valeurs CSV #NUM aux pandas NA, avecpd.NA()

3. **Entité.** Ensuite, nous transformons la colonne d'entité de catégorie en colonne numérique. Pour y parvenir, nous l'avons renommée « Individuel » et avons mappé ses valeurs sur 0 et 1.

4. **Sexe.** Associez-le à 0 et 1 et assurez-vous que les valeurs manquantes sont correctement indiquées. Vérifiez l'impact de cette opération sur le nombre de valeurs manquantes. Dans ce cas, nous pouvons laisser la variable « sex » sous forme de chaîne ; il n'y a aucune raison pour qu'elle soit un entier, un flottant ou un booléen.

5. **Convention relative aux minuscules.** Nous veillons à ce que les valeurs « purpose » et « source » soient uniquement des minuscules.

6. **Hypothèque.** Travaillez sur la variable « hypothèque ». Associez-la à 0 et 1 et assurez-vous que les valeurs manquantes sont correctement indiquées. Vérifiez comment cela a affecté le nombre de valeurs manquantes.

7. **Nom et prénom.** Nous aimerions également créer une nouvelle colonne, Full name, qui remplacera les informations de name et surname. Essayez d'obtenir des valeurs qui ressemblent à Name Surname. Pensez à supprimer les séries nameet surname de l'ensemble de données.

8. **Date de naissance.** Dans cette étape, nous devons prétraiter birth_date. Nous utiliserons les fonctionnalités DateTime pour nous assurer que nous pourrons l'utiliser ultérieurement comme une date plutôt qu'une chaîne, ce qui est crucial pour notre analyse d'âge.

Vous pouvez également parcourir les variables restantes et vous assurer qu'il n'y a pas d'autres problèmes.

Combinaison des deux ensembles de données

Cette phase vise à fusionner nos deux ensembles de données nettoyés (properties et customers) en un ensemble de données complet.

1. **Vérifications préliminaires.** Notre première étape consiste à inspecter visuellement les cinq premières lignes de chaque ensemble de données à l'aide de la .head() fonction. Une observation clé est la customer_ID fonctionnalité des deux tables, offrant un moyen pratique de fusionner ces deux ensembles de données.

2. **Tentative de fusion initiale.** En tirant parti de la fonction native .merge() de Pandas, nous utiliserons customer_IDcomme colonne de jointure. Étant donné qu'une personne ne devient client qu'après avoir acheté une propriété, nous cherchons à conserver la propertiestable complète et à customersy joindre des données. Ici, nous utilisons une jointure gauche.

3. **Identification et résolution des problèmes de fusion.** Malheureusement, notre tentative de fusion initiale a entraîné l'absence de données client. Notre enquête nous a conduit à la customer_IDcolonne. Il semble que les identifiants de la propertiestable soient flanqués d'espaces vides, ce qui empêche une fusion

réussie. Pour rectifier ce problème, nous utilisons la .strip()méthode permettant de supprimer les espaces inutiles dans customer_IDles séries des deux tables.

4. **Vérifications finales et fusion.** Avant de réessayer la fusion, nous comparons le nombre d'éléments uniques dans la customer_IDcolonne des deux tables pour garantir l'alignement. Le décompte révèle que certaines propriétés ne sont pas vendues et qu'il manque un customer_ID. Un espace les représente. Nous remplaçons ensuite ces valeurs par la représentation standard des valeurs manquantes dans pandas PD, NA.

5. **Fusion réussie.** Après ces étapes de nettoyage, nous fusionnons les tables, créant ainsi une table commune nommée real estate data. Une inspection visuelle des entrées du haut et du bas à l'aide des .head() méthodes .tail() et indique une fusion réussie, avec les détails du client correctement alignés avec les détails de la propriété.

6. **Gestion des valeurs manquantes.** Comme prévu, les propriétés qui restent à vendre manquent de détails sur les clients, représentés par le type de données manquantes natif de pandas. Nous utilisons la fill.NA() méthode pour remplacer ces valeurs par « NA », ce qui facilite la gestion future des données.

7. **Vérification du type de données.** Enfin, avant de continuer, nous vérifions les types de données de notre table combinée, en observant un mélange d'entiers, de colonnes DateTime, de flottants et d'objets texte. Une fois cette vérification finale terminée, nous sommes prêts à passer à la phase suivante de notre analyse.

Partie 2 : Statistiques descriptives

8. Répartition par bâtiment

Nous commençons par créer une variable nommée `data`qui contient toutes les données pertinentes pour notre analyse. Pour obtenir un aperçu rapide, nous affichons les cinq premières lignes de l'ensemble de données. Ensuite, nous utilisons la `.describe()`méthode pour obtenir des statistiques descriptives. Étant donné que la méthode par défaut ne renvoie que des informations sur les séries numériques, nous définissons le `include`paramètre sur `all` pour afficher les détails de toutes les séries du tableau.

En examinant les statistiques, nous rencontrons un Future Warning message concernant le traitement des DateTime valeurs comme des valeurs numériques. Pour adopter le comportement futur, nous définissons le datetime_is_numeric paramètre sur True. Parmi les informations affichées, nous observons que la building colonne comporte cinq options uniques, nous permettant d'analyser l'ensemble de données en fonction du bâtiment auquel appartient chaque propriété. À l'aide d'un tableau de distribution de fréquences, nous pouvons décomposer les moyennes ou les totaux.

Pour la répartition des moyennes, nous identifions les colonnes d'intérêt, notamment building, area, price in dollarset deal satisfaction. Nous créons une averages_by_building variable et stockons les données agrégées. En examinant les moyennes, nous découvrons des informations sur les propriétés du bâtiment 4 qui sont plus grandes, plus chères et qui conduisent à une plus grande satisfaction client.

9. Répartition par pays

Pour construire un tableau de distribution de fréquence pour le pays, nous redéfinissons les colonnes de la variable d'intérêt pour inclure country, soldet mortgage. Nous créons ensuite une nouvelle variable appelée totals_by_countrypour stocker les données groupées. Nous remarquons deux entrées pour les États-Unis en raison d'un espace supplémentaire et utilisons la .strip() méthode pour supprimer le symbole en excès. Nous supprimons en outre les espaces supplémentaires potentiels de toutes les colonnes de texte pour assurer la cohérence. Après cela, nous pouvons obtenir une répartition précise des totaux et des moyennes.

10. Répartition par État

Poursuivant notre analyse statistique, nous nous concentrons maintenant sur la décomposition des données par état. Nous redéfinissons les colonnes de variable d'intérêt pour inclure state, soldet mortgage. Nous créons une nouvelle variable appelée totals_by_statepour stocker les données groupées. Mais nous rencontrons des valeurs manquantes dans la série « état », qui doivent être remplacées.

Avant de continuer, nous vérifions que chaque client avec une state valeur est bien des États-Unis. En comparant le nombre d' soldobservations avec le totals_by_country tableau, nous identifions une incohérence, indiquant la présence de clients étrangers ou de données incorrectes. Pour maintenir la cohérence, nous supposons que les clients de pays non américains ont fourni les state informations par erreur et définissons l'état sur valeurs manquantes (pd.NA) pour ces cas.

Après avoir nettoyé la statesérie, nous avons réussi à décomposer les données par État. Nous remarquons que le nombre de clients des États spécifiés correspond au nombre de clients américains du totals_by_countrytableau, ce qui garantit la cohérence.

Pour améliorer le tableau de répartition, nous supprimons la mortgagecolonne, trions les valeurs par ordre décroissant en fonction de la sold colonne et renommons la série pour refléter avec précision la distribution de fréquence.

Dans les deux dernières étapes de cette analyse, nous examinons également la fréquence relative et cumulative en ajoutant deux nouvelles colonnes au sold_by_statebloc de données : la première appelée relative_frequency, qui inclut la fréquence relative des différents états, la seconde cumulative_frequency, qui contient la fréquence cumulative des différents états. La fréquence relative est la fréquence de chaque état divisée par la fréquence totale, puis nous calculons la fréquence cumulative ; nous pouvons utiliser la .cumsum() méthode native dans pandas.

Partie 3 : Analyse des données

Analyse de l'âge des clients

Nous commençons par calculer le age_at_purchaseen trouvant la différence entre les colonnes date_of_saleet birth_date . L'objet delta de temps résultant doit être converti en jours et divisé par 365 pour obtenir l'âge en années. Il est recommandé d'arrondir vers le bas pour une interprétation plus simple.

Après avoir obtenu les age_at_purchasevaleurs, nous calculons des statistiques descriptives telles que la moyenne, l'écart type, le minimum, le maximum et les percentiles. De plus,

nous explorons la différence entre l'écart type de la population et de l'échantillon et apprenons à reproduire les statistiques obtenues à l'aide de la `.describe()` méthode.

Pour faciliter la phase suivante de notre analyse, nous créons des intervalles d'âge. En divisant les données en intervalles équidistants, vous pouvez analyser la répartition des achats de biens immobiliers entre différentes tranches d'âge. Cette répartition fournira des informations précieuses sur les préférences des clients et leurs achats en fonction de l'âge.

Analyse des prix de l'immobilier

De la même manière que pour l'analyse de la variable d'âge, nous commençons par créer des intervalles de prix. Pour plus de commodité, nous utilisons **10** bacs. Ensuite, nous trouvons le nombre de propriétés dans chaque intervalle de prix. Ensuite, nous voyons le nombre de propriétés vendues par intervalle de prix et, enfin, nous vérifions également le nombre de propriétés encore invendues.

Relation entre l'âge et le prix :

Pour analyser la relation entre l'âge des clients et les prix des biens immobiliers, nous filtrons d'abord l'ensemble de données pour n'inclure que les biens vendus, en veillant à ce que l'analyse se concentre sur les données pertinentes. Ensuite, nous calculons la covariance entre l'âge et le prix à l'aide de la `.cov()` fonction NumPy. La covariance révélera l'étendue de la relation et si l'âge a un impact sur le prix des biens immobiliers.

Le coefficient de corrélation entre l'âge et le prix peut être calculé pour examiner plus en détail la relation. Nous explorons les différences entre les corrélations entre la population et l'échantillon et validons l'interprétation des résultats.

Partie 4 : Visualisation des données

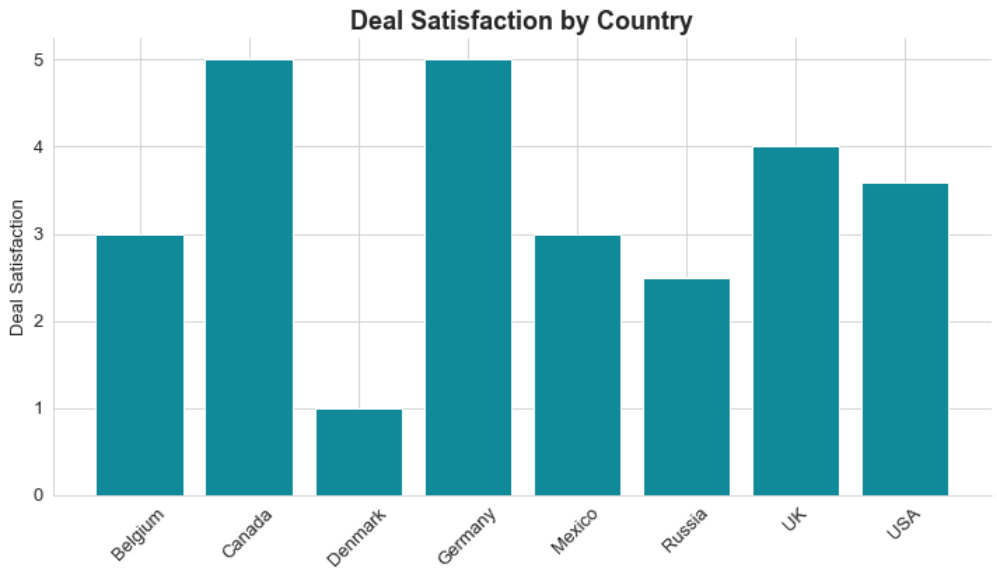
Après avoir effectué toute la préparation des données et l'analyse statistique nécessaires, nous pouvons désormais facilement créer les visualisations requises pour mieux comprendre et analyser les données.

Graphique à barres de la satisfaction des transactions par pays

Pour représenter graphiquement la satisfaction des clients par pays, nous avons besoin de données montrant une répartition de la satisfaction des clients par pays. Heureusement, nous avons déjà créé une telle répartition avec le tableau des moyennes par pays. Nous pouvons l'utiliser pour créer un graphique à barres en utilisant la méthode des barres de Matplotlib et représenter les pays sur l'axe des x et la satisfaction moyenne sur l'axe des y.

Code:

```
sns.set_style("whitegrid") # override the default matplotlib style, to avoid the grey background, but keep the grid
plt.figure(figsize = (12, 6)) #determine the size of the chart
plt.bar(x = averages_by_country.index, #specify the x axis
        height = averages_by_country['deal_satisfaction'], #specify the y axis
        color = "#108A99") # the color for the bars (365 Data Science color)
# format the ticks
plt.xticks(rotation = 45, fontsize = 13) # rotate and format the labels for the x-axis
plt.yticks(fontsize = 13) # format the y-axis
plt.title("Deal Satisfaction by Country", fontsize = 18, fontweight = "bold") #add and format the title for the chart
plt.ylabel("Deal Satisfaction", fontsize = 13 ) #add a title for the y-axis
sns.despine() # removes the top and right border of our graph
plt.savefig("deal_satisfaction_by_country_bar_chart.png") # you can export your chart as a picture
plt.show()
```



Histogramme de la répartition par âge

Pour tracer la distribution par âge, nous avons besoin de la variable d'âge que nous avons créée. La fonctionnalité de Matplotib nous permet de définir le paramètre bins pour qu'il fonctionne directement avec l'âge au lieu des intervalles d'âge que nous avons créés plus tôt dans le projet. L'histogramme ne prend qu'une seule variable : l'âge à l'achat. Segmentation par état Diagramme de Pareto

Pour cette visualisation, nous devons afficher la fréquence relative et cumulative des propriétés vendues par État. Heureusement, nous disposons d'un tableau affichant ces informations exactes provenant de la partie statistique du projet. Pour créer un Pareto, nous pouvons utiliser une figure avec deux sous-graphiques, le premier montrant la fréquence relative sous forme de graphique à barres, le second la fréquence cumulative sous forme de graphique linéaire. Les deux graphiques montrent l'État sur leur axe des x.

Code :

```
sns.set_style("whitegrid") # override the default matplotlib style, to avoid the grey background, but keep the grid
plt.figure(figsize = (12, 6)) # determine the size of the figure
plt.hist(data['age_at_purchase'], # the variable on which to create the histogram
        bins = 10, # create a histogram with 10 bins
        color = "#108A99")
plt.title("Age Distribution", fontsize = 18, weight = "bold")
plt.xlabel("Age", fontsize=13)
plt.ylabel("Number of Purchases", fontsize=13)
sns.despine() # removes the top and right border of our graph
plt.savefig("age_distribution_histogram.png") # you can export your chart as a picture
plt.show()
```

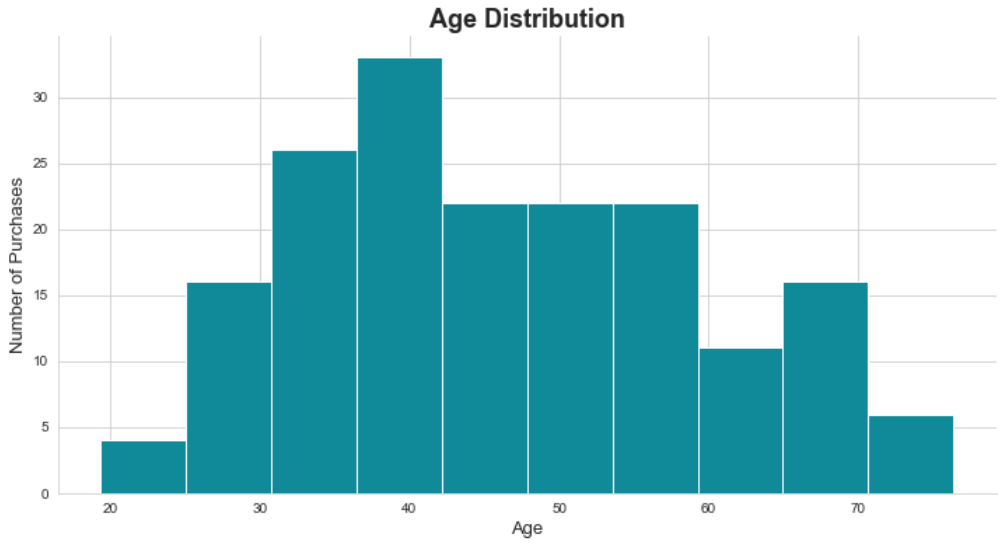


Diagramme de Pareto

Tout d'abord, nous ajoutons un arrière-plan blanc propre à l'aide de **sns.set_style("white")** . Cela remplace le style **matplotlib** par défaut pour donner un aspect plus propre à notre graphique.

Un graphique de Pareto contient un graphique à barres et un graphique linéaire. Par conséquent, l'étape suivante consiste à initialiser une figure avec deux axes à l'aide de **plt.subplots(figsize = (12, 6))** .

Dans le graphique à barres, nous représentons la fréquence des ventes d'appartements par État sur l'axe des Y et les États correspondants sur l'axe des X. En conséquence, définissez l'étiquette de l'axe des Y sur « Appartements vendus » et formatez les graduations de l'axe des Y pour qu'elles correspondent à la couleur de notre barre.

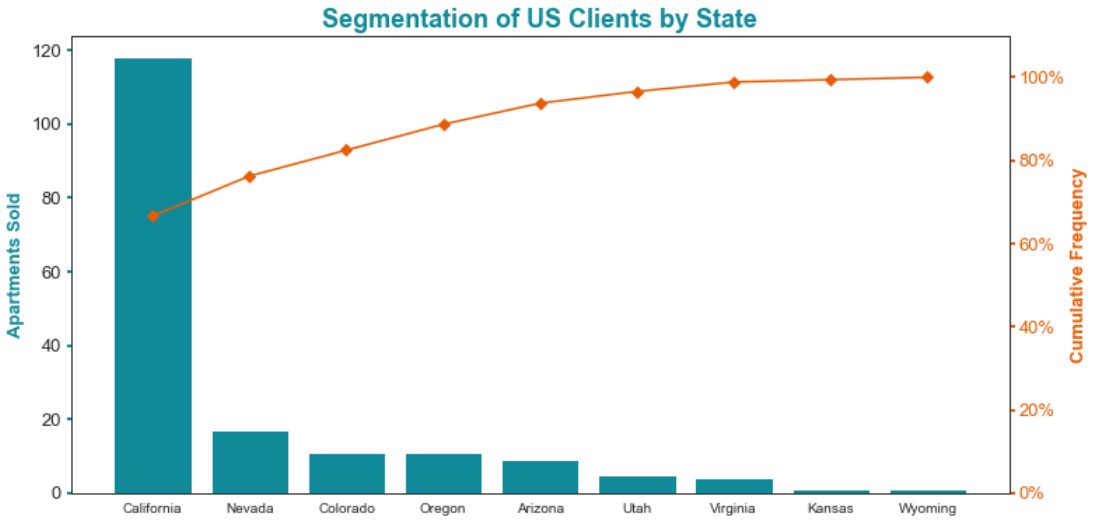
Ensuite, définissez un deuxième axe Y qui partage le même axe X que le graphique à barres. Il représentera des pourcentages jusqu'à un maximum de 100 %.

Ici, tracez un graphique linéaire pour représenter la fréquence cumulée des ventes d'appartements. Nous étiquetons cet axe Y comme « Fréquence cumulée » et formatons ses graduations pour qu'elles correspondent à la couleur de la ligne.

Enfin, le diagramme de Pareto a besoin d'un titre descriptif avant que nous puissions enregistrer notre travail. Nommez-le (« Segmentation des clients américains par État »), enregistrez-le sous forme de fichier .png à l'aide de la commande **plt.savefig()** et affichez-le avec **plt.show()** .

```
# Setting the seaborn style to "white" for aesthetic reasons - to override the default matplotlib style,
# removing the grey background and grid for a cleaner look.
sns.set_style("white")
# To create the Pareto diagram, we first initiate a figure with two axes.
fig, ax = plt.subplots(figsize = (12, 6))
# The first part of our visualization is a bar chart, featuring the 'sold_by_state' index on the x-axis
# and the frequency on the y-axis.
ax.bar(sold_by_state.index,
       sold_by_state['frequency'],
       color = "#108A99")
# Setting a descriptive y-axis label.
ax.set_ylabel("Apartments Sold",
              weight='bold',
              fontsize = 13,
              color = "#108A99")
# Formatting the y-axis ticks.
ax.tick_params(axis = "y",
               width = 2,
               labelsize = 13,
               color = "#108A99")
# Creating a second axis that shares the same x-axis as the first.
ax2 = ax.twinx()
# Setting the y-axis limit and formatting it to show percentages.
ax2.set_ylim(0, 1.1)
ax2.yaxis.set_major_formatter(PercentFormatter(xmax = 1.0))
# The second part of our visualization is a line chart sharing the x-axis with the bar chart,
```

```
# and featuring the cumulative frequency on the y-axis.
ax2.plot(sold_by_state.index,
        sold_by_state["cumulative_frequency"],
        color = "#E85D04",
        marker = "D")
# Setting a descriptive y-axis label.
ax2.set_ylabel("Cumulative Frequency",
              color = "#E85D04",
              weight = "bold",
              fontsize=13)
# Formatting the y-axis ticks.
ax2.tick_params(axis = "y",
               colors = "#E85D04",
               width = 2,
               labelsize = 13)
# Adding a descriptive and engaging title to our visualization.
ax.set_title("Segmentation of US Clients by State", fontsize = 18, weight = "bold", color =
"#108A99")
# Saving the visualization as an image file.
plt.savefig("US_segmentation_by_state_pareto_diagram.png")
# Displaying the final visualization.
plt.show()
```

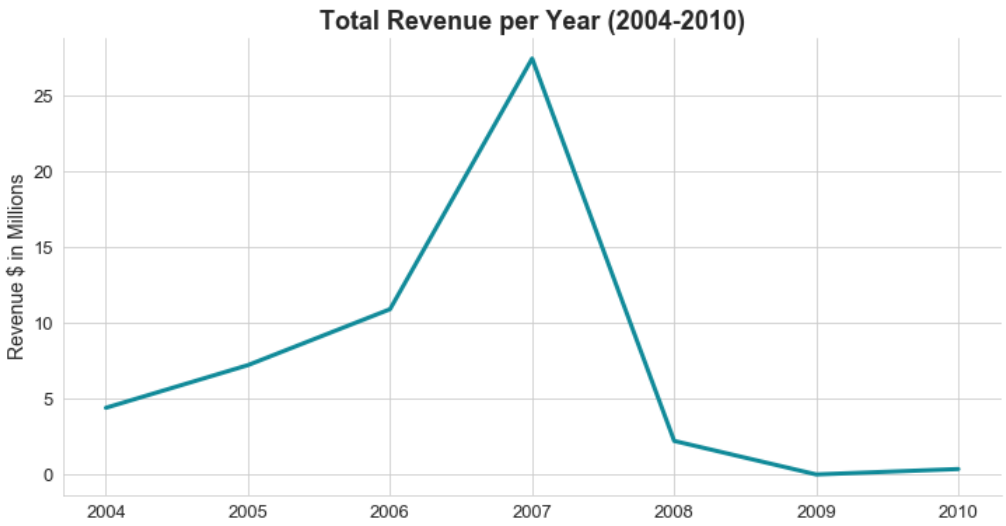


Graphique linéaire des ventes totales par an

Pour créer cette visualisation, nous devons connaître le total des ventes annuelles. Pour ce faire, nous pouvons manipuler la variable date_sale et la transformer au format correct pour l'inclure en la transformant en un entier. Après cela, nous pouvons créer la répartition pertinente de l'année de vente et du chiffre d'affaires et tracer un graphique en courbes, avec la fonction plot de Matplotlib, par exemple, en traçant le chiffre d'affaires sur l'axe des y et l'année sur l'axe des x.

```
# Let's override the default matplotlib style, opting for a white grid layout.
sns.set_style("whitegrid")
# We'll generate a figure that aligns aesthetically with our other visualizations.
plt.figure(figsize = (12, 6))
# A line chart will be constructed to represent revenue per year.
plt.plot(revenue_per_year['revenue$'],
        color='#108A99',
        linewidth=3)
# Let's ensure our visualization includes essential chart elements like title, labels, and legends.
plt.title("Total Revenue per Year (2004-2010)", fontsize = 18, fontweight = "bold")
plt.ylabel("Revenue $", fontsize = 13)
plt.xticks(fontsize = 13) # We'll format and adjust the orientation of labels on the x-axis.
```

```
plt.yticks(fontsize = 13) # y-axis label formatting
sns.despine() # We'll remove the top and right borders of the chart for a cleaner look.
# Don't forget to save the visualization as an image file.
plt.savefig("total_revenue_per_year_in_M_line_chart.png")
plt.show() # Finally, display the visualization.
```



Graphique des ventes totales par année et par zone de bâtiment

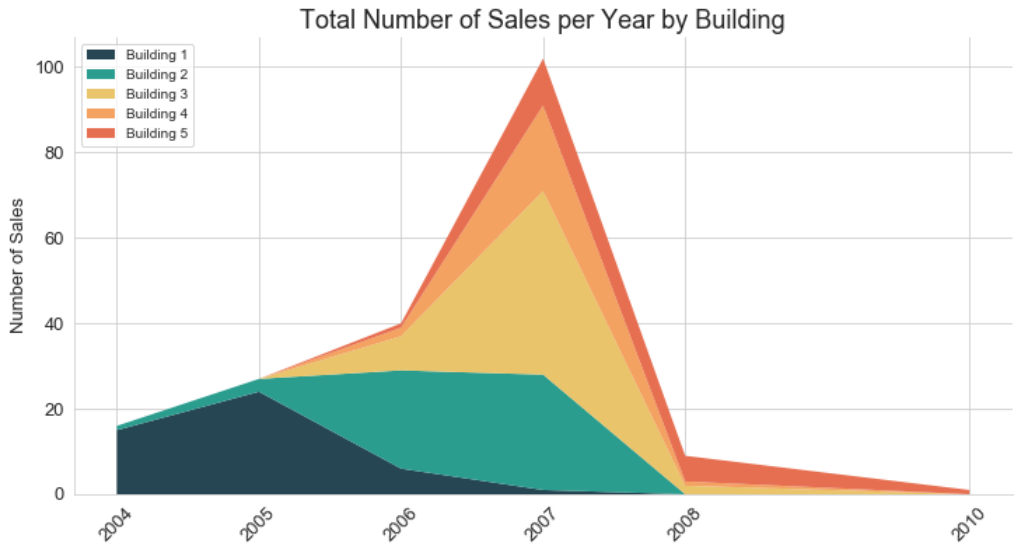
Pour représenter graphiquement le nombre total de ventes par an et par bâtiment, nous devons créer une répartition par année et par bâtiment, ce qui nécessite la création d'un nouveau bloc de données. De plus, ici, nous pouvons créer des variables indicatrices (fictives) basées sur la variable building_variable, que nous stockons directement dans notre nouveau bloc de données et que nous renommons de manière appropriée. Après avoir filtré les propriétés invendues, nous pouvons décomposer l'année des cinq variables factices de bâtiment.

En ayant la répartition par année et par bâtiment, nous pouvons créer un graphique à aires empilées en utilisant le « stackplot » de Matplotlib. L'année est représentée sur l'axe des x, l'axe des y montre le nombre de ventes et la surface est divisée par le type de bâtiment.

```
# let's choose different colors for each of the buldings
```

```
# We'll assign unique colors for each of the buildings to differentiate them on the plot.
colors = ["#264653", "#2A9D8F", "E9C46A",'F4A261','E76F51']
# List of labels for the legend.
# These should appear in the same order as the stacked area plot categories.
labels = ['Building 1','Building 2','Building 3','Building 4','Building 5',]
# Utilize seaborn's 'whitegrid' theme for a cleaner look with a white background.
sns.set_style("whitegrid")
# Initiate a figure that's in line with the dimensions of the other figures.
plt.figure(figsize = (12, 6))
# Create a stacked area plot with our data.
plt.stackplot(stacked_area.index, # The x-axis is simply the index (the year).
              stacked_area['building1'],
              stacked_area['building2'],
              stacked_area['building3'],
              stacked_area['building4'],
              stacked_area['building5'],
              colors = colors,
              edgecolor = 'none')
# Include x-axis labels for each year and rotate them by 45 degrees for better readability.
plt.xticks(stacked_area.index, rotation = 45)
# Add a legend and specify its location on the chart.
plt.legend(labels = labels, loc = "upper left")
# Label the y-axis and format the x and y tick marks for improved readability.
plt.ylabel("Number of Sales", fontsize = 13)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
# Provide a clear title for the chart.
plt.title("Total Number of Sales per Year by Building", fontsize = 18)
# Remove top and right borders of the chart for a cleaner look.
sns.despine()
# Save your chart as an image for external use.
plt.savefig("total_sales_per_year_per_building_stacked_area_chart.png")
```

```
# Display the plot.
plt.show()
```



Partie 5 : Interprétation des données

Après avoir prétraité, analysé et visualisé les données, il est temps de discuter des raisons pour lesquelles cette analyse et cette définition du problème sont pertinentes et pour qui. Les listes de clients et de propriétés aident à créer un profil du client idéal de l'entreprise et à comprendre les types de bâtiments et leurs caractéristiques.

Profil client

Les profils clients permettent aux propriétaires et promoteurs immobiliers de rechercher les acheteurs parfaits. Mais il existe une petite variable : actuellement, la demande est supérieure à l'offre de logements abordables dans la plupart des villes américaines et européennes, par exemple, ce qui rend difficile l'identification du profil client idéal. D'un autre côté, il existe des projets immobiliers plus exclusifs qui proposent des appartements de luxe à une poignée de privilégiés. Dans de tels cas, les sociétés immobilières peuvent utiliser ce profil pour cibler les clients potentiels avec des publicités sur Facebook, YouTube ou d'autres médias.

En ce qui concerne le profil du client, on peut noter que l'âge auquel la plupart des clients achètent une maison se situe entre 31 et 42 ans, plus précisément la tranche d'âge 36 à 42 ans, suivie de près par celle de 31 à 26 ans. C'est plutôt logique puisque ces clients dans cette tranche d'âge sont plus susceptibles d'être financièrement stables pour acheter un bien immobilier.

intervalle_d'âge	vendu
(19.0, 25.0]	4
(25,0, 31,0]	16
(31.0, 36.0]	26
(36,0, 42,0]	33
(42,0, 48,0]	22
(48,0, 54,0]	22
(54,0, 59,0]	22
(59,0, 65,0]	11
(65,0, 71,0]	16
(71,0, 76,0]	6

Gardez à l'esprit que l'échantillon statistique ne comprend qu'une centaine de clients environ. Il nous faudrait donc un échantillon de données plus large pour pouvoir tirer une conclusion plus concluante. Néanmoins, lorsqu'elle dispose de suffisamment d'informations, l'agence immobilière peut créer un profil d'acheteur idéal pour découvrir les clients dont les caractéristiques correspondent à celles de sa personnalité idéale. Elle peut ensuite cibler le groupe spécifique de personnes par le biais de publicités sur YouTube, Google, Facebook ou des marchés similaires.

Caractéristiques du bâtiment

La deuxième liste contient des informations plus simples : elle détaille les caractéristiques des bâtiments de notre ensemble de données, telles que :

- Les zones où le plus de propriétés sont vendues
- Les prix des biens vendus

- La période pendant laquelle la vente a eu lieu

Ces données sur les propriétés sont largement disponibles et faciles à trouver. En fonction des données, vous pourrez même trouver des informations plus détaillées sur la propriété, par exemple l'étage, l'orientation et le style des fenêtres (par exemple, à la française), etc. Vous pouvez ainsi créer un aperçu complet du marché et de ses tendances.
Voici un exemple de la façon d'interpréter cet ensemble de données :
Répartition par bâtiment
Sur la base de nos données, nous concluons que les bâtiments les plus vendus sont les types 2 et 3 :

bâtiment	vendu	hypothèque
1	46	14.0
2	54	18.0
3	53	15.0
4	23	9.0
5	19	6.0

Cependant, lorsque nous examinons le prix moyen et la satisfaction des clients, nous remarquons autre chose :

bâtiment	zone	prix$	affaire_satisfaction
1	928.038846	275143.242500	3.630435
2	943.891930	286661.848246	3.518519
3	927.852381	280451.255556	3.566038
4	974.720930	290239.515581	3.869565
5	914.298654	274557.604615	3.526316

Le type 4 affiche les prix de vente moyens les plus élevés, soit 290 000 USD. Les types 2 et 3 suivent de près. En revenant aux immeubles du type 4, nous constatons que la satisfaction moyenne des transactions est également la plus élevée, proche de 3,9.

Le type 4 est également le plus élevé en termes de superficie.

Il est probable que les immeubles de type 4 abritent des appartements plus luxueux ou plus spacieux. Les totaux montrent que ces types de biens ne se vendent pas aussi souvent que

les autres. Ils sont également plus chers en raison de leur taille, car les immeubles sont vendus au mètre carré. Les clients sont plus susceptibles d'acheter des types d'immeubles plus petits et moins chers, comme l'indiquent les tendances de notre ensemble de données. Les promoteurs immobiliers peuvent tenir compte de ces informations lors de l'élaboration de leurs stratégies. Une option consiste à s'appuyer sur des types d'immeubles plus standard, tels que les types 2 et 3, et à vendre davantage d'immeubles à un prix plus moyen. Alternativement, ils peuvent investir dans quelques appartements plus luxueux et les vendre à un prix plus élevé, créant ainsi une plus grande satisfaction commerciale.

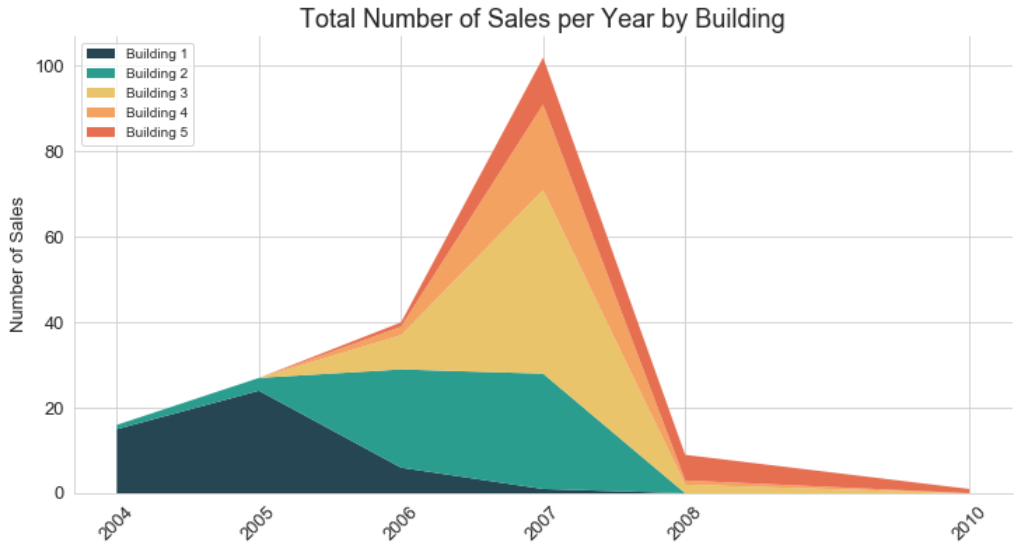
Ventes par pays

En regardant les données géographiques, on remarque que 90% des ventes proviennent des États-Unis :

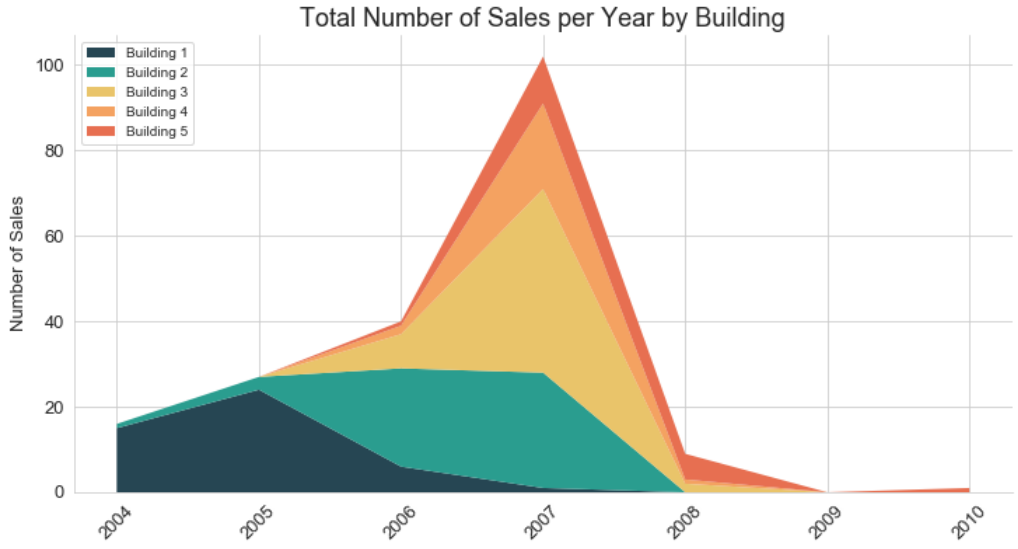
pays	vendu	hypothèque
Belgique	2	0,0
Canada	7	0,0
Danemark	1	0,0
Allemagne	1	0,0
Mexique	1	0,0
Russie	4	1.0
ROYAUME-UNI	2	0,0
cerf	177	61,0

Nombre total de ventes par an par bâtiment

Relions ces informations à la période à laquelle la plupart des ventes ont eu lieu en utilisant le graphique en aires empilées du nombre total de ventes par an par bâtiment.



Nous observons que 2007 est de loin l'année la plus populaire en termes de ventes. Compte tenu des événements historiques, cela n'est pas surprenant car cela s'est produit juste avant le krach boursier aux États-Unis. En fait, comme le montrent nos graphiques en courbes et en aires empilées, il n'y a pas eu de ventes en 2009. Comme il s'agit d'informations clés à prendre en compte, nous avons dû l'ajouter explicitement dans la dernière partie du projet :



Nous pouvons effectuer une analyse similaire des clients et des bâtiments lorsque nous obtenons des données plus récentes.