

## INTRODUCTION

Le marché immobilier canadien est un domaine en constante évolution, représentant un enjeu majeur pour les investisseurs cherchant à exploiter ou vendre une propriété. Le secteur immobilier est actuellement en difficulté au Canada et partout dans le monde entre la période de 2023 à 2024. Cela est sûrement l'un des enjeux majeurs des gouvernements à l'international.

Ce projet a pour but de créer des visualisations de modèles afin de faciliter les analyses d'investissement personnel ou professionnel, soutenues par l'apprentissage automatique. De plus, j'ai voulu aller plus loin avec des méthodes de visualisation peut-être moins agréables à l'œil nu, mais intéressantes à titre informatif.

This image is a summary page from the project. It features a large, stylized title "SOMMAIRE" in bold, dark letters. Below the title is a collage of various data visualizations, including heatmaps, bar charts, and scatter plots. To the right of the collage is a vertical list of four numbered steps: 1. Nettoyage des données, Compréhension des tendances; 2. Modélisation du jeu de données, test et prédition; 3. Validation de la prédition; 4. Conclusion résultat. Each step is accompanied by a small icon. At the bottom of the page, there is a footer with the year "2024", credit to "Crédit du DataFrame Kaggle", and the author's name "Marvin Hugues Laurac".

## LOGICIEL PRIVILÉGIÉ

- Python

J'ai utilisé Python avec ces bibliothèques, comme Pandas pour l'analyse de données, Scikit-Learn pour le machine learning, et Matplotlib pour la visualisation.

**PRÉVISION DES PRIX DE L'IMMOBILIER CANADIEN****CONTEXTE DU SUJET**

Ce projet représente des données immobilières Canadiennes avec comme informations, le prix, le nombre de salles de bains, le nombre de chambres, ainsi que la ville et la province. Ces données ont été récoltées grâce aux annonces immobilières des 45 villes les plus peuplées du Canada selon le recensement de 2021, mais une mise à jour a été faite en date du 29 octobre 2023 avec des mises à jour mensuellement.

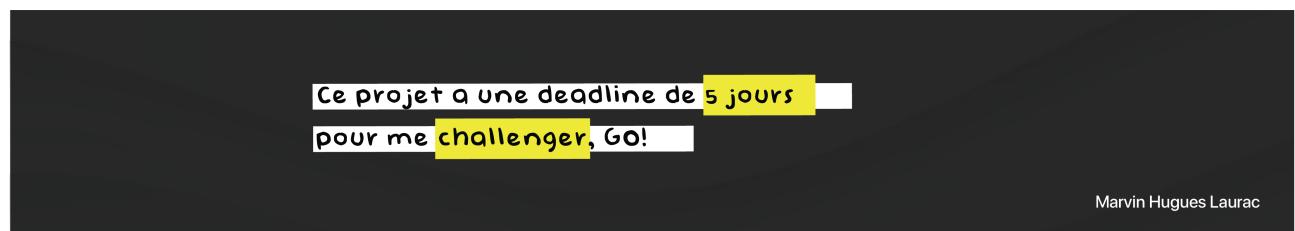
À noter: Ces informations ont été indiquées par l'auteur. De mon côté, j'ai travaillé sur ce DataFrame en prenant en compte ces indications, en les reformulant et en faisant mes recherches de mon côté.

**DATAFRAME INFO**

| # | Column       | Non-Null Count | Dtype    |         |   |   |                        |
|---|--------------|----------------|----------|---------|---|---|------------------------|
| 0 | City         | 35768          | non-null | object  | 5 | Province                                | 35768 non-null object  |
| 1 | Price        | 35768          | non-null | float64 | 6 | Population                              | 35768 non-null int64   |
| 2 | Address      | 35768          | non-null | object  | 7 | Latitude                                | 35768 non-null float64 |
| 3 | Number_Beds  | 35768          | non-null | int64   | 8 | Longitude                               | 35768 non-null float64 |
| 4 | Number_Baths | 35768          | non-null | int64   | 9 | Median_Family_Income                    | 35768 non-null float64 |
| 5 | Province     | 35768          | non-null | object  |   | dtypes: float64(4), int64(3), object(3) |                        |
|   |              |                |          |         |   | memory usage: 2.7+ MB                   |                        |

**LICENCE**

kaggle (Database Contents License (DbCL) v1.0)



```

Entrée [9]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

output_notebook()

BokehJS 3.2.1 successfully loaded.

Entrée [10]: #je commence par importer le DataFrame
df = pd.read_csv('/Users/octoberone/Desktop/DataFrame GitHub/DataFrame/Prévision des prix de immobilier canadien.csv')

Entrée [11]: #affichage des 5 premières lignes
df.head()

Out[11]:
   City      Price        Address  Number_Beds  Number_Baths  Province  Population  Latitude  Longitude  Median_Family_Income
0  Toronto  779900.0  #318-20 SOUTHPORT ST       3            2    Ontario     5647656   43.7417   -79.3733        97000.0
1  Toronto  799999.0  #818-60 SOUTHPORT ST       3            1    Ontario     5647656   43.7417   -79.3733        97000.0
2  Toronto  799900.0  #714-859 THE QUEENSWAY      2            2    Ontario     5647656   43.7417   -79.3733        97000.0
3  Toronto  1200000.0      275 MORTIMER AVE      4            2    Ontario     5647656   43.7417   -79.3733        97000.0
4  Toronto  668800.0  #420-388 RICHMOND ST       1            1    Ontario     5647656   43.7417   -79.3733        97000.0

Entrée [12]: #voici le nombre de lignes et de colonnes sans le détail
df.shape

Out[12]: (35768, 10)

```

**PRÉVISION DES PRIX DE L'IMMOBILIER CANADIEN**

Entrée [13]: #voici les détails des colonnes avec beaucoup plus d'informations (variables, types, le nombre de valeurs nulles)

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35768 entries, 0 to 35767
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   City              35768 non-null   object  
 1   Price             35768 non-null   float64 
 2   Address           35768 non-null   object  
 3   Number_Beds       35768 non-null   int64  
 4   Number_Baths      35768 non-null   int64  
 5   Province          35768 non-null   object  
 6   Population        35768 non-null   int64  
 7   Latitude          35768 non-null   float64 
 8   Longitude         35768 non-null   float64 
 9   Median_Family_Income 35768 non-null   float64 
dtypes: float64(4), int64(3), object(3)
memory usage: 2.7+ MB
```

Entrée [14]: #j'ai utilisé cette commande pour être sûr qu'il n'y a pas de valeurs manquantes

```
df.isnull().sum()
#par conséquent, j'en déduis que ce DataFrame est plutôt agréable à utiliser dans un premier temps
```

```
Out[14]: City          0
Price          0
Address        0
Number_Beds    0
Number_Baths   0
Province        0
Population     0
Latitude        0
Longitude       0
Median_Family_Income 0
dtype: int64
```

Entrée [15]: #Afin d'avoir plus de détails statistiques, j'affiche les colonnes numériques

```
#count%:nombre de valeurs nulles
#mean%:la moyenne des valeurs
#std%:l'écart type (dispersion valeurs)
#min%:le min de la colonne
#25%:1er quartile à moins de 25
#50%:2er quartile à moins de 25
#75%:3er quartile à moins de 25
#max:le maxi de la colonne

print(df.describe())

      Price  Number_Beds  Number_Baths  Population  Latitude \
count  3.576800e+04  35768.000000  35768.000000  3.576800e+04  35768.000000
mean   9.432963e+05   3.283661   2.532403   6.360151e+05   47.446556
std    1.020110e+06   1.730654   1.371910   1.120016e+06   3.333855
min    2.150000e+04   0.000000   0.000000   6.338200e+04   42.283300
25%   4.599000e+05   2.000000   2.000000   1.091670e+05   43.866700
50%   6.990000e+05   3.000000   2.000000   2.424600e+05   49.025000
75%   1.095000e+06   4.000000   3.000000   5.228880e+05   49.888100
max   3.700000e+07  109.000000  59.000000  5.647656e+06  53.916900

      Longitude  Median_Family_Income
count  35768.000000  35768.000000
mean   -98.421636  89643.103416
std    22.280935  12132.353510
min    -123.936400  62400.000000
25%   -122.316700  82000.000000
50%   -104.606700  89000.000000
75%   -79.866700  97000.000000
max    63.100500  133000.000000
```

**PRÉVISION DES PRIX DE L'IMMOBILIER CANADIEN**

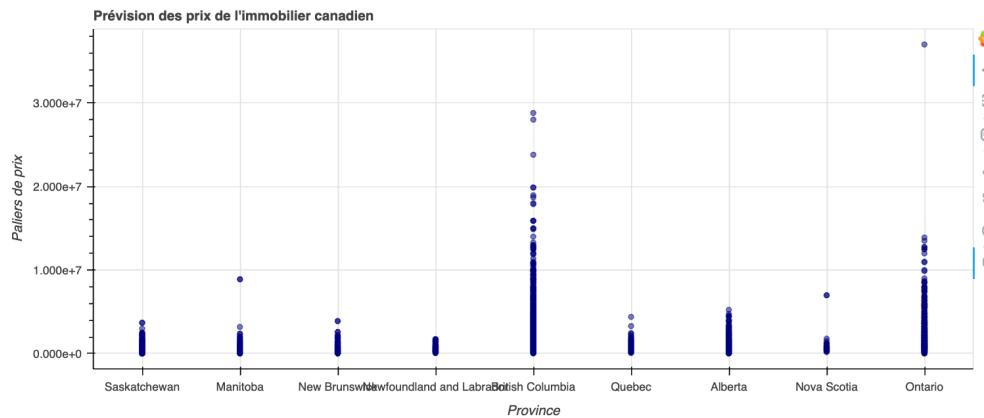
```
Entrée [16]: #je commence par ce premier affichage
#je regroupe les provinces dans la variable province
#puis grâce à une visualisation interactive, j'affiche le prix des villes regroupées dans les provinces
#la visualisation n'est la plus agréable par rapport aux données variées et volumineuses du DataFrame
#mais je trouve intéressant de réaliser cet affichage interactif

plt.figure(figsize=(10,15))
unique_provinces = list(set(df["Province"]))
source = ColumnDataSource(data=df)

hover = HoverTool(
    tooltips=[
        ("City", "@City"),
        ("Price", "@Price")
    ])

p = figure(width=950, height = 400, title ="Prévision des prix de l'immobilier canadien",
           x_axis_label = "Province",
           y_axis_label = "Prix de l'immobilier",
           x_range = unique_provinces)
p.circle(x="Province", y="Price", source = source, size = 5, color = "navy", alpha = 0.5)
p.add_tools(hover)

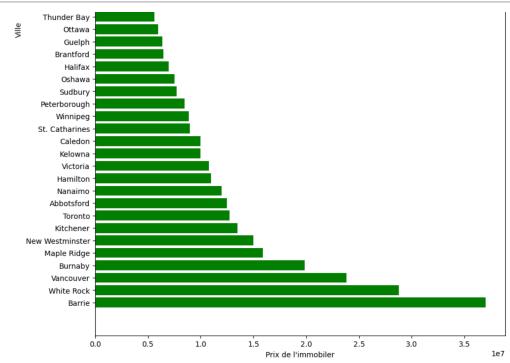
show(p)
```



&lt;Figure size 1000x1500 with 0 Axes&gt;

```
Entrée [17]: #dans cet histogramme plutôt classique
#je peux observer une tendance plus significative sur le prix de l'immobilier en fonction des villes au Canada
top_city = df.sort_values(by="Price", ascending=False)
plt.figure(figsize=(10,15))

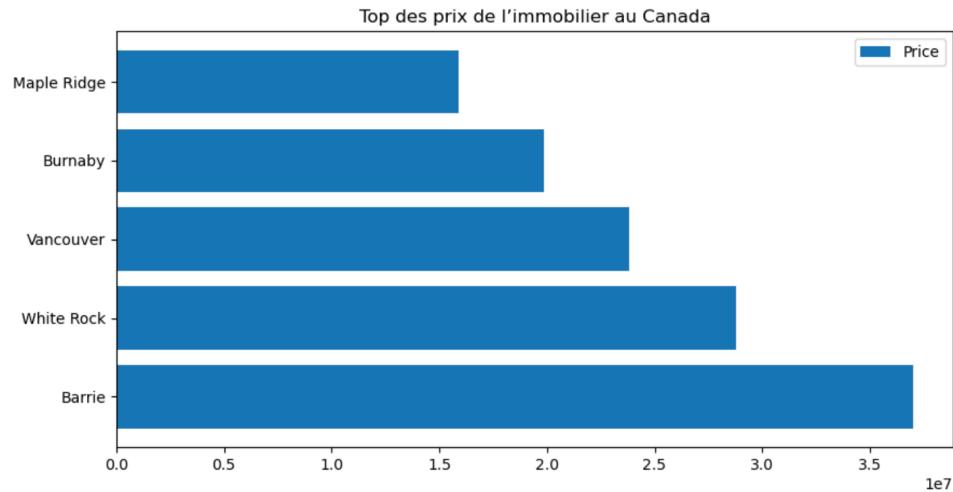
plt.barh(top_city["City"], top_city["Price"], label="Prix (price)", color= "g")
plt.title("Affichage des prix croissants de l'immobilier au Canada")
plt.xlabel("Ville")
plt.ylabel("Prix de l'immobilier")
plt.legend(title="Légende")
plt.show()
```



**PRÉVISION DES PRIX DE L'IMMOBILIER CANADIEN**

```
Entrée [18]: #ans cet histogramme des Top 5 des prix de l'immobilier canadien
#je peux voir que la province de Barrie est en tête avec le prix le plus élevé et Maple Ridge est à la dernière place
top_city = df.sort_values(by="Price", ascending=False).head(13)

plt.figure(figsize=(10,5))
plt.barh(top_city["City"], top_city["Price"], label="Price")
plt.legend()
plt.title("Top des prix de l'immobilier au Canada")
plt.show()
```



```
Entrée [19]: #cette partie a juste été utilisée pour observer les variables que je pourrais supprimer en les affichant
#j'ai rencontré plusieurs problèmes car premièrement j'avais réalisé l'encodage avant la suppression
#ce qui m'a donné des erreurs puis je me suis dit qu'il vaut mieux supprimer les variables inutiles en premier
print(df.columns)
```

```
Index(['City', 'Price', 'Address', 'Number_Beds', 'Number_Baths', 'Province',
       'Population', 'Latitude', 'Longitude', 'Median_Family_Income'],
      dtype='object')
```

```
Entrée [20]: df.drop(["Address", "Latitude", "Longitude"], axis=1, inplace=True)
df.head()
```

Out[20]:

|   | City    | Price     | Number_Beds | Number_Baths | Province | Population | Median_Family_Income |
|---|---------|-----------|-------------|--------------|----------|------------|----------------------|
| 0 | Toronto | 779900.0  | 3           | 2            | Ontario  | 5647656    | 97000.0              |
| 1 | Toronto | 799999.0  | 3           | 1            | Ontario  | 5647656    | 97000.0              |
| 2 | Toronto | 799900.0  | 2           | 2            | Ontario  | 5647656    | 97000.0              |
| 3 | Toronto | 1200000.0 | 4           | 2            | Ontario  | 5647656    | 97000.0              |
| 4 | Toronto | 668800.0  | 1           | 1            | Ontario  | 5647656    | 97000.0              |

```
Entrée [21]: #afin d'avoir le meilleur résultat possible, j'ai réalisé un encodage pour garantir une homogénéité des données en n
label = LabelEncoder()
df[["City"]] = label.fit_transform(df[["City"]])
df[["Province"]] = label.fit_transform(df[["Province"]])
df.head()
```

Out[21]:

|   | City | Price     | Number_Beds | Number_Baths | Province | Population | Median_Family_Income |
|---|------|-----------|-------------|--------------|----------|------------|----------------------|
| 0 | 38   | 779900.0  | 3           | 2            | 6        | 5647656    | 97000.0              |
| 1 | 38   | 799999.0  | 3           | 1            | 6        | 5647656    | 97000.0              |
| 2 | 38   | 799900.0  | 2           | 2            | 6        | 5647656    | 97000.0              |
| 3 | 38   | 1200000.0 | 4           | 2            | 6        | 5647656    | 97000.0              |
| 4 | 38   | 668800.0  | 1           | 1            | 6        | 5647656    | 97000.0              |

**PRÉVISION DES PRIX DE L'IMMOBILIER CANADIEN**

```

Entrée [22]: #e vais séparer les colonnes dans la variable X sauf "Price" dans y
#pour isoler les caractéristiques de l'entraînement
X = df.drop("Price", axis=1)
y = df["Price"]

Entrée [23]: #maintenant, je vais fractionner le jeu de données
#pour tester la performance de la prédition
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Entrée [25]: #Ensuite, une standardisation
#pour améliorer les résultats
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

Entrée [26]: #et enfin j'effectue une régression linéaire pour réaliser la prédition
model = LinearRegression()
model.fit(X_train, y_train)

Out[26]: ▾ LinearRegression
LinearRegression()

```

```

Entrée [27]: #afin de confirmer la prédition je vais calculer l'erreur quadratique moyenne et le coefficient
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(mse)
print(r2)
#observe une performance MSE beaucoup trop élevée (j'aurais préféré avoir un chiffre plus proche de 0)
#j'observe une performance R² beaucoup trop basse de 24 % (j'aurais préféré avoir un chiffre dans les 90 %)

780535792611.0696
0.24400791164081526

```

```

Entrée [28]: #même si je pense que cette prédition est assez compliquée à identifier, j'ai souhaité réaliser un tableau
#e peux apercevoir une tendance plutôt linéaire dans l'ensemble
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)

plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], "k--")

plt.xlabel("Prix actuel")
plt.ylabel("Prix prédit")
plt.title("Prix actuel vs Prix prédict")
plt.show()

```

