In [1]:
```
import os
```

```
C:\Users\Lenovo
```

In [2]:
```
```

In [3]:
```
# save filepath to variable for easier access
me_file_path = './immodata.csv'
# read the data and store data in DataFrame titled me_data
me_data = pd.read_csv(me_file_path)
# print a summary of the data in me data
```

Out[3]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	1.534242
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	0.691712
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	1.000000
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	1.000000
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	2.000000
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	8.000000

In [4]:
```
import pandas as pd

me_file_path = './immodata.csv'
me_data = pd.read_csv(me_file_path)
```

Out[4]:
```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitud
e',
       'Longtitude', 'Regionname', 'Propertycount'],
      dtype='object')
```

In [5]:
```
# dropna drops missing values (think of na as "not available")
```

In [6]:
```
```

In [7]:
```
```

In [8]:
```
```

In [9]:

Out[9]:

	Rooms	Bathroom	Landsize	Lattitude	Longtitude
count	6196.000000	6196.000000	6196.000000	6196.000000	6196.000000
mean	2.931407	1.576340	471.006940	-37.807904	144.990201
std	0.971079	0.711362	897.449881	0.075850	0.099165
min	1.000000	1.000000	0.000000	-38.164920	144.542370
25%	2.000000	1.000000	152.000000	-37.855438	144.926198
50%	3.000000	1.000000	373.000000	-37.802250	144.995800
75%	4.000000	2.000000	628.000000	-37.758200	145.052700
max	8.000000	8.000000	37000.000000	-37.457090	145.526350

In [10]:

Out[10]:

	Rooms	Bathroom	Landsize	Lattitude	Longtitude
1	2	1.0	156.0	-37.8079	144.9934
2	3	2.0	134.0	-37.8093	144.9944
4	4	1.0	120.0	-37.8072	144.9941
6	3	2.0	245.0	-37.8024	144.9993
7	2	1.0	256.0	-37.8060	144.9954

In [11]:
```
from sklearn.tree import DecisionTreeRegressor

# Define model. Specify a number for random_state to ensure same results ea
me_model = DecisionTreeRegressor(random_state=1)

# Fit model
```

Out[11]:
```
DecisionTreeRegressor

DecisionTreeRegressor(random_state=1)
```

In [12]:
```
print("Making predictions for the following 5 houses:")
print(X.head())
print("The predictions are")
```

```
Making predictions for the following 5 houses:
   Rooms  Bathroom  Landsize  Lattitude  Longtitude
1      2       1.0     156.0   -37.8079    144.9934
2      3       2.0     134.0   -37.8093    144.9944
4      4       1.0     120.0   -37.8072    144.9941
6      3       2.0     245.0   -37.8024    144.9993
7      2       1.0     256.0   -37.8060    144.9954
The predictions are
[1035000. 1465000. 1600000. 1876000. 1636000.]
```

In [13]:
```
# Filter rows with missing price values
filtered_me_data = me_data.dropna(axis=0)
# Choose target and features
y = filtered_me_data.Price
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                        'YearBuilt', 'Lattitude', 'Longtitude']
X = filtered_me_data[fme_features]

from sklearn.tree import DecisionTreeRegressor
# Define model
me_model = DecisionTreeRegressor()
# Fit model
```

Out[13]:
```
▾ DecisionTreeRegressor

DecisionTreeRegressor()
```

In [14]:
```
from sklearn.metrics import mean_absolute_error

predicted_home_prices = me_model.predict(X)
```

Out[14]:
```
434.71594577146544
```

In [15]:
```
from sklearn.model_selection import train_test_split

# split data into training and validation data, for both features and targe
# The split is based on a random number generator. Supplying a numeric valu
# the random_state argument guarantees we get the same split every time we
# run this script.
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 0)
# Define model
me_model = DecisionTreeRegressor()
# Fit model
me_model.fit(train_X, train_y)

# get predicted prices on validation data
val_predictions = me_model.predict(val_X)
```

```
265517.3279535184
```

In [16]:
```
from sklearn.metrics import mean_absolute_error
from sklearn.tree import DecisionTreeRegressor

def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes, random_sta
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
```

In [17]:
```
# Data Loading Code Runs At This Point
import pandas as pd

# Load data
me_file_path = './immodata.csv'
me_data = pd.read_csv(me_file_path)
# Filter rows with missing values
filtered_me_data = me_data.dropna(axis=0)
# Choose target and features
y = filtered_me_data.Price
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                        'YearBuilt', 'Lattitude', 'Longtitude']
X = filtered_me_data[fme_features]

from sklearn.model_selection import train_test_split

# split data into training and validation data, for both features and targe
```

In [18]:
```
# compare MAE with differing values of max_leaf_nodes
for max_leaf_nodes in [5, 50, 150, 250, 500, 1500, 5000]:
    my_mae = get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y)
```

```
Max leaf nodes: 5               Mean Absolute Error:  347380
Max leaf nodes: 50              Mean Absolute Error:  258171
Max leaf nodes: 150             Mean Absolute Error:  253766
Max leaf nodes: 250             Mean Absolute Error:  247206
Max leaf nodes: 500             Mean Absolute Error:  243495
Max leaf nodes: 1500            Mean Absolute Error:  252130
Max leaf nodes: 5000            Mean Absolute Error:  255575
```

In [19]:
```
import pandas as pd

# Load data
me_file_path = './immodata.csv'
me_data = pd.read_csv(me_file_path)
# Filter rows with missing values
me_data = me_data.dropna(axis=0)
# Choose target and features
y = me_data.Price
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                        'YearBuilt', 'Lattitude', 'Longtitude']
X = me_data[fme_features]

from sklearn.model_selection import train_test_split

# split data into training and validation data, for both features and targe
# The split is based on a random number generator. Supplying a numeric valu
# the random_state argument guarantees we get the same split every time we
```

In [20]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
```

```
191669.7536453626
```

In [21]:
```
# Récupération du nombre de feuilles dans chaque arbre
n_leaves_per_tree = [tree.get_n_leaves() for tree in forest_model.estimator

# Afficher les résultats
for i, n_leaves in enumerate(n_leaves_per_tree):
    print(f"Arbre {i+1}: {n_leaves} feuilles")

# Nombre total de feuilles dans la forêt (si pertinent)
```

```
Arbre 1: 2850 feuilles
Arbre 2: 2870 feuilles
Arbre 3: 2868 feuilles
Arbre 4: 2908 feuilles
Arbre 5: 2900 feuilles
Arbre 6: 2889 feuilles
Arbre 7: 2845 feuilles
Arbre 8: 2845 feuilles
Arbre 9: 2844 feuilles
Arbre 10: 2854 feuilles
Arbre 11: 2893 feuilles
Arbre 12: 2908 feuilles
Arbre 13: 2835 feuilles
Arbre 14: 2869 feuilles
Arbre 15: 2857 feuilles
Arbre 16: 2916 feuilles
Arbre 17: 2853 feuilles
Arbre 18: 2888 feuilles
Arbre 19: 2889 feuilles
Arbre 20: 2850 feuilles
Arbre 21: 2889 feuilles
Arbre 22: 2876 feuilles
Arbre 23: 2830 feuilles
Arbre 24: 2876 feuilles
Arbre 25: 2855 feuilles
Arbre 26: 2846 feuilles
Arbre 27: 2896 feuilles
Arbre 28: 2881 feuilles
Arbre 29: 2875 feuilles
Arbre 30: 2867 feuilles
Arbre 31: 2884 feuilles
Arbre 32: 2844 feuilles
Arbre 33: 2882 feuilles
Arbre 34: 2903 feuilles
Arbre 35: 2838 feuilles
Arbre 36: 2868 feuilles
Arbre 37: 2906 feuilles
Arbre 38: 2882 feuilles
Arbre 39: 2869 feuilles
Arbre 40: 2853 feuilles
Arbre 41: 2846 feuilles
Arbre 42: 2859 feuilles
Arbre 43: 2882 feuilles
Arbre 44: 2860 feuilles
Arbre 45: 2868 feuilles
Arbre 46: 2858 feuilles
Arbre 47: 2893 feuilles
Arbre 48: 2867 feuilles
Arbre 49: 2884 feuilles
Arbre 50: 2824 feuilles
Arbre 51: 2905 feuilles
Arbre 52: 2852 feuilles
Arbre 53: 2882 feuilles
Arbre 54: 2865 feuilles
Arbre 55: 2892 feuilles
Arbre 56: 2832 feuilles
Arbre 57: 2865 feuilles
Arbre 58: 2856 feuilles
Arbre 59: 2884 feuilles
Arbre 60: 2866 feuilles
```

```
Arbre 61: 2879 feuilles
Arbre 62: 2883 feuilles
Arbre 63: 2879 feuilles
Arbre 64: 2854 feuilles
Arbre 65: 2888 feuilles
Arbre 66: 2855 feuilles
Arbre 67: 2879 feuilles
Arbre 68: 2896 feuilles
Arbre 69: 2875 feuilles
Arbre 70: 2873 feuilles
Arbre 71: 2840 feuilles
Arbre 72: 2835 feuilles
Arbre 73: 2891 feuilles
Arbre 74: 2864 feuilles
Arbre 75: 2866 feuilles
Arbre 76: 2863 feuilles
Arbre 77: 2869 feuilles
Arbre 78: 2854 feuilles
Arbre 79: 2841 feuilles
Arbre 80: 2872 feuilles
Arbre 81: 2898 feuilles
Arbre 82: 2915 feuilles
Arbre 83: 2853 feuilles
Arbre 84: 2896 feuilles
Arbre 85: 2890 feuilles
Arbre 86: 2903 feuilles
Arbre 87: 2925 feuilles
Arbre 88: 2882 feuilles
Arbre 89: 2853 feuilles
Arbre 90: 2864 feuilles
Arbre 91: 2858 feuilles
Arbre 92: 2877 feuilles
Arbre 93: 2869 feuilles
Arbre 94: 2891 feuilles
Arbre 95: 2933 feuilles
Arbre 96: 2912 feuilles
Arbre 97: 2859 feuilles
Arbre 98: 2863 feuilles
Arbre 99: 2880 feuilles
Arbre 100: 2863 feuilles
Nombre total de feuilles dans tous les arbres : 287231
```

In [22]:
```
import pandas as pd

# Load data
me_file_path = './immodata.csv'
me_data = pd.read_csv(me_file_path)
# Filter rows with missing values
me_data = me_data.dropna(axis=0)
# Choose target and features
y = me_data.Price
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                        'YearBuilt', 'Lattitude', 'Longtitude']
X = me_data[fme_features]

from sklearn.model_selection import train_test_split

# split data into training and validation data, for both features and targe
# The split is based on a random number generator. Supplying a numeric valu
# the random_state argument guarantees we get the same split every time we
```

In [23]:
```
def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = RandomForestRegressor(max_leaf_nodes=max_leaf_nodes, random_sta
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
```

In [24]:
```
# compare MAE with differing values of max_leaf_nodes
for max_leaf_nodes in [2700, 2750, 2800, 2850, 2900, 2950]:
    my_mae = get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y)
```

```
Max leaf nodes: 2700            Mean Absolute Error:  192499
Max leaf nodes: 2750            Mean Absolute Error:  192496
Max leaf nodes: 2800            Mean Absolute Error:  192496
Max leaf nodes: 2850            Mean Absolute Error:  192497
Max leaf nodes: 2900            Mean Absolute Error:  192497
Max leaf nodes: 2950            Mean Absolute Error:  192497
```

In [25]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(max_depth=20, random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
```

```
191945.7841081549
```

In [26]:
```
import pandas as pd

# Load data
me_file_path = './immodata.csv'
me_data = pd.read_csv(me_file_path)
# Filter rows with missing values
#me_data = me_data.dropna(axis=0)
me_data = me_data.fillna(me_data.mean(numeric_only=True))
# Choose target and features
y = me_data.Price
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                        'YearBuilt', 'Lattitude', 'Longtitude']
X = me_data[fme_features]

from sklearn.model_selection import train_test_split

# split data into training and validation data, for both features and targe
# The split is based on a random number generator. Supplying a numeric valu
# the random_state argument guarantees we get the same split every time we
```

In [27]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(max_depth=20, random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
```

```
175294.6787475501
```

In [28]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error

df = pd.DataFrame(me_data)

# Préparation des données
X = df.drop(columns=['Price'])
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                          'YearBuilt', 'Lattitude', 'Longtitude']
X = me_data[fme_features]
y = df['Price']

# Configuration de GridSearchCV
param_grid = {
    'n_estimators': [100, 150, 200, 250, 300],  # Nombre d'arbres
    'max_depth': [None, 20],   # Profondeur maximale des arbres
}

model = RandomForestRegressor(random_state=1)

# GridSearchCV
grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    cv=3,  # 3-fold cross-validation
    scoring='neg_mean_squared_error',  # Minimize MAE
    n_jobs=-1,  # Utiliser tous les cœurs disponibles
    verbose=1
)

# Entraînement
grid_search.fit(X, y)

# Meilleurs paramètres
print("Meilleurs paramètres :")
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Meilleurs paramètres :
{'max_depth': None, 'n_estimators': 250}
```

```
import pandas as pd

# Load data
me_file_path = './immodata.csv'
me_data = pd.read_csv(me_file_path)
# Filter rows with missing values
#me_data = me_data.dropna(axis=0)
me_data = me_data.fillna(me_data.mean(numeric_only=True))
# Choose target and features
y = me_data.Price
fme_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',
                          'YearBuilt', 'Lattitude', 'Longtitude']
X = me_data[fme_features]

from sklearn.model_selection import train_test_split
```

```
# split data into training and validation data, for both features and targe
# The split is based on a random number generator. Supplying a numeric valu
to
# the random_state argument guarantees we get the same split every time we
```

In [30]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(n_estimators=250, random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
```

```
174089.68573549337
```