## Assessment Cover Page

### Student Full Name: Ilia Grishkin

### Student Number: 2024139

### Module Title: Machine Learning

### Assessment Title: CA1 Project

### Assessment Due Date: 21st April 2024

### Date of Submission: 2nd April 2024

# Content

* [**Introduction**](#section1)
  + [**Literate review**](#section2)
  + [**Research question**](#section3)
  + [**Data sourse and methods**](#section4)
* [**EDA**](#section6)
  + [**Libraries and summary**](#section7)
  + [**Boxplots**](#section8)
  + [**Histograms**](#section9)
  + [**Heatmap**](#section10)
* [**Data preparation**](#section11)
  + [**Drop missing values**](#section12)
  + [**Outliers**](#section13)
  + [**Encoding**](#section14)
  + [**PCA**](#section15)
  + [**Defining a target**](#section141)
  + [**Data scalling**](#section142)
  + [**Data splitting**](#section143)
* [**Hyperparameter Tuning**](#section151)
* [**Multiple linear Regression**](#section16)
* [**Decision Tree (Classifier)**](#section18)
* [**Decision Tree (Regressor)**](#section19)
* [**Support Vector Machine**](#section20)
* [**Random forest (Categoric target)**](#section21)
* [**Random forest (Numerical target)**](#section22)
* [**K-nearest neighbors**](#section23)
* [**References**](#section99)

# Housing Price Prediction Using Machine Learning Models

# Introduction

The main problem which may be associated with Ireland’s economy is currently experiencing a severe housing crisis. Housing prices are a big part of a country's economy and affect many other sectors. This issue has deep roots, as was mentioned by Nowlan (2016) and Scuffil (2022), this crisis is going back to the 1900s.

Lyons, R. (2018) point out that the property market is a regular topic of national attention in Ireland. For the majority of house owners, their house is the most valuable asset.

Ireland's economy has been suffering from one of the most severe housing busts of the global financial crisis (Norris and Byrne M. (2017), however in recent years been represented as having recovered economically (Nowicki et al., 2019). Since 2013, Irish house prices have increased by 50%, while rent rates have grown by over 60% according to Byrne M. (2020).

The pricing of houses is affected by many different factors, such as the location of the house, the features of the house and condition Phan, T. (2019).

Many factors may affect the prices of housing. These factors, as well as other parameters such as the materials used for the building, number of bedrooms, living area, location, upcoming projects and proximity were noted by Bourassa, Cantoni and Hoesli (2011).

The prediction of housing sale price may be considered an essential economic metric. The value of a house that grows with time requires the estimated value to be calculated as this value is required for sale, purchase or even mortgage. (Shinde and Gawande, 2018).

## Literate review

Before getting started, we should investigate recent research, methods and results. It will help us to understand which methods we can use and which results we should expect.

1. Aswin (2017) applied 6 different machine learning models to predict house prices in a data set with 2000 records and 10 features. The author used 6 machine learning algorithms, such as Random Forest, Neural Networks, Gradient Boosted, Bagging, Support Vector Machine and Multiple Regression. The best accuracy was performed Random Forest with R-squared value of 90%.
2. Hujia Yu and Jiafu Wu (2016), also were working on a price prediction model. They created regression and classification models which are able to estimate the price of the house given the features. It was concluded that for classification models the best model is the Support Vector Classifier with linear kernel. The model showed an accuracy of 0.6740 and after PCA was performed on the dataset it increased to 0.6913. For the regression problem, the best model is Support Vector Regression with a Gaussian kernel, with an RMSE of 0.5271.
3. Ng A. (2015) explored the use of machine learning methods for London house price prediction. The approach is used to create local models by comparing various regression methods. The Gaussian method was found to be most efficient because of its probabilistic approach to learning and model selection.

## Research question

Each dataset behaves differently. A machine learning model may work with high accuracy in one dataset, but perform poorly in another despite both of them being applied to similar data. Social and economic data is very dynamic in contrast to physical or chemical where approved theory can be reviewed only in unique circumstances. This implies that data related to the economy or social sector should be reviewed preferably as soon as new data has come. This study is going to focus on applying different machine learning algorithms to Irish housing prices datasets, in order to understand which model gives the best accuracy.

## Data sourse and methods

### Data Source

As a data source, we will use an available for public use dataset "daft.ie house price data" published on Kaggle (<https://www.kaggle.com/datasets/eavannan/daftie-house-price-data>). The dataset contains 3869 records about Irish property which were published in 2021 and 2022. Additionally data set includes accounts 22 features which reflect all essential property parameters for creating a machine learning model.

### EDA

Exploratory Data Analysis (EDA). It will help us to understand the structure of the dataset including the size, shape, properties and types of variables. Also, identify patterns and relationships between variables. Additionally, EDA allows us to select appropriate techniques and models for our further analysis.

### Data preparation

After we get the dataset explored using EDA, we can clearly understand which data preparation techniques should be applied.

1. Data Cleaning
2. Data Transformation
3. Handling outliers
4. Feature engineering
5. Feature Selection

### Machine Learning Models

We are going to apply multiple machine learning models to the dataset in order to understand which one gives the highest accuracy. Also, we will try to discover the advantages and disadvantages of an apply of each model to data the datasets related to the property market in Ireland.

1. Multiple linear Regression
2. Linear Regression
3. Decision Trees
4. Support Vector Machine
5. K-Nearest Neighbors
6. Random Forest
7. XGBoost Words in Intoduction section: 520

## EDA

### Libraries and summary

import pandas as pd  
import warnings  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
import xgboost as xgb  
from xgboost import XGBRegressor  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn import preprocessing  
from sklearn.decomposition import PCA  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.tree import DecisionTreeClassifier, plot\_tree   
from sklearn import metrics   
from sklearn import tree  
from sklearn.metrics import confusion\_matrix, classification\_report  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.svm import SVC  
from sklearn.preprocessing import MinMaxScaler   
from sklearn.preprocessing import LabelEncoder  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import GridSearchCV   
from sklearn.neighbors import KNeighborsClassifier  
warnings.filterwarnings("ignore")

df = pd.read\_csv("daft\_ie\_v1.csv")

df.head(1)

id title featuredLevel \  
0 3626025 11 Chestnut Crescent, Bridgemount, Carrigaline... featured   
  
 publishDate price numBedrooms numBathrooms propertyType \  
0 2022-01-28 290000 3 3 End of Terrace   
  
 propertySize category ... seller\_name seller\_branch \  
0 96.0 Buy ... Roy Dennehy Dennehy Auctioneers   
  
 sellerType m\_totalImages m\_hasVideo m\_hasVirtualTour m\_hasBrochure \  
0 BRANDED\_AGENT 16.0 False False False   
  
 ber\_rating longitude latitude   
0 C2 -8.3825 51.82294   
  
[1 rows x 22 columns]

df.shape

(3967, 22)

df.describe(include="all")

id title featuredLevel \  
count 3.258000e+03 3258 3258   
unique NaN 3209 3   
top NaN Glebe Manor Estate, Whitegate, Co. Cork standard   
freq NaN 6 2908   
mean 3.639687e+06 NaN NaN   
std 2.107207e+05 NaN NaN   
min 3.154700e+04 NaN NaN   
25% 3.655543e+06 NaN NaN   
50% 3.674251e+06 NaN NaN   
75% 3.686086e+06 NaN NaN   
max 3.695360e+06 NaN NaN   
  
 publishDate price numBedrooms numBathrooms propertyType \  
count 3258 3258.000000 3258.000000 3258.000000 3258   
unique 53 NaN NaN NaN 9   
top 2022-01-28 NaN NaN NaN Semi-D   
freq 997 NaN NaN NaN 825   
mean NaN 295125.719460 2.983732 1.943524 NaN   
std NaN 142047.768721 1.011165 0.916470 NaN   
min NaN 20000.000000 1.000000 1.000000 NaN   
25% NaN 195000.000000 2.000000 1.000000 NaN   
50% NaN 270000.000000 3.000000 2.000000 NaN   
75% NaN 365000.000000 4.000000 3.000000 NaN   
max NaN 775000.000000 10.000000 8.000000 NaN   
  
 propertySize category ... seller\_name seller\_branch \  
count 3258.000000 3258 ... 3258 3258   
unique NaN 2 ... 1193 795   
top NaN Buy ... Auctioneera Dublin Office\* BidX1   
freq NaN 3206 ... 32 74   
mean 104.196133 NaN ... NaN NaN   
std 39.411193 NaN ... NaN NaN   
min 1.000000 NaN ... NaN NaN   
25% 75.000000 NaN ... NaN NaN   
50% 98.000000 NaN ... NaN NaN   
75% 125.000000 NaN ... NaN NaN   
max 229.000000 NaN ... NaN NaN   
  
 sellerType m\_totalImages m\_hasVideo m\_hasVirtualTour \  
count 3258 3258.000000 3258 3258   
unique 3 NaN 2 2   
top BRANDED\_AGENT NaN False False   
freq 2409 NaN 2658 2945   
mean NaN 18.551565 NaN NaN   
std NaN 10.282826 NaN NaN   
min NaN 0.000000 NaN NaN   
25% NaN 12.000000 NaN NaN   
50% NaN 17.000000 NaN NaN   
75% NaN 23.000000 NaN NaN   
max NaN 93.000000 NaN NaN   
  
 m\_hasBrochure ber\_rating longitude latitude   
count 3258 3258 3258.000000 3258.000000   
unique 2 17 NaN NaN   
top False C2 NaN NaN   
freq 3151 400 NaN NaN   
mean NaN NaN -7.358411 53.133805   
std NaN NaN 1.986695 0.709158   
min NaN NaN -100.445882 39.783730   
25% NaN NaN -8.393205 52.672939   
50% NaN NaN -6.940025 53.304158   
75% NaN NaN -6.291580 53.424796   
max NaN NaN -6.028016 55.299693   
  
[11 rows x 22 columns]

df.dtypes

id int64  
title object  
featuredLevel object  
publishDate object  
price int64  
numBedrooms int64  
numBathrooms int64  
propertyType object  
propertySize float64  
category object  
AMV\_price int64  
sellerId float64  
seller\_name object  
seller\_branch object  
sellerType object  
m\_totalImages float64  
m\_hasVideo bool  
m\_hasVirtualTour bool  
m\_hasBrochure bool  
ber\_rating object  
longitude float64  
latitude float64  
dtype: object

df.isnull().sum()

id 0  
title 0  
featuredLevel 0  
publishDate 0  
price 0  
numBedrooms 0  
numBathrooms 0  
propertyType 0  
propertySize 355  
category 0  
AMV\_price 0  
sellerId 0  
seller\_name 0  
seller\_branch 0  
sellerType 0  
m\_totalImages 0  
m\_hasVideo 0  
m\_hasVirtualTour 0  
m\_hasBrochure 0  
ber\_rating 0  
longitude 0  
latitude 0  
dtype: int64

categorical\_indices = [1, 2, 3, 5, 6, 7, 9, 10, 12, 14, 15, 16, 17]  
rooms\_index = [5, 6]  
selected\_columns2 = df.iloc[:, rooms\_index]

def get\_unique\_values(df, categorical\_indices):  
 unique\_values\_dict = {}  
 for col\_index in categorical\_indices:  
 column\_name = df.columns[col\_index]  
 unique\_values = df.iloc[:, col\_index].unique()  
 unique\_values\_dict[column\_name] = unique\_values  
 return unique\_values\_dict  
  
unique\_values = get\_unique\_values(df, categorical\_indices)  
  
for column\_name, values in unique\_values.items():  
 print(f"Column '{column\_name}' unique values")  
 print(values)  
 print()

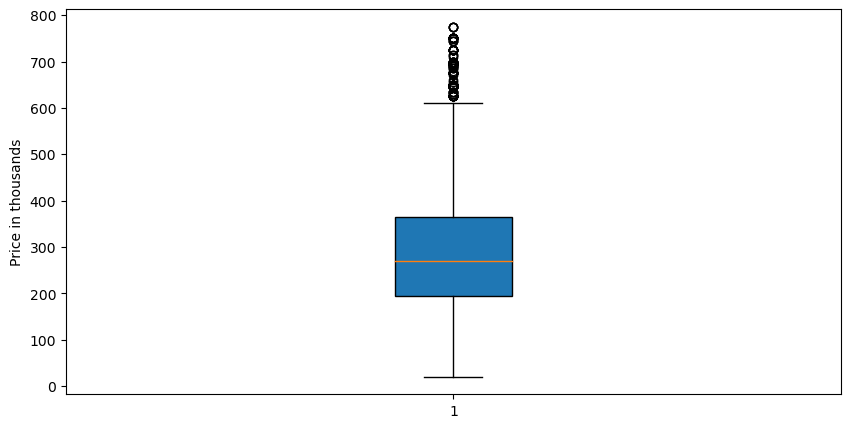
Column 'title' unique values  
['11 Chestnut Crescent, Bridgemount, Carrigaline, Co. Cork'  
 '58 The Glen, Kilnacourt Woods, Portarlington, Co. Laois'  
 '16 Dodderbrook Park, Ballycullen, Dublin 24' ...  
 '69 McAuley Drive, Artane, Artane, Dublin 5'  
 'School Land, Ballinalee, Co. Longford'  
 '14 Coolmagort Ave, Beaufort, Killarney, Co. Kerry']  
  
Column 'featuredLevel' unique values  
['featured' 'premium' 'standard']  
  
Column 'publishDate' unique values  
['2022-01-28' '2022-01-27' '2022-01-30' '2022-01-26' '2022-01-12'  
 '2022-01-14' '2022-01-11' '2022-01-25' '2022-01-10' '2022-01-07'  
 '2022-01-06' '2022-01-24' '2022-01-29' '2022-01-21' '2022-01-20'  
 '2022-01-19' '2022-01-18' '2022-01-17' '2021-12-17' '2022-01-05'  
 '2022-01-04' '2022-01-03' '2022-01-15' '2021-12-31' '2021-12-01'  
 '2022-01-23' '2022-01-09' '2022-01-01' '2021-12-15' '2021-12-30'  
 '2021-12-23' '2021-12-20' '2022-01-22' '2022-01-16' '2022-01-13'  
 '2022-01-08' '2021-12-29' '2021-12-28' '2021-12-27' '2021-12-24'  
 '2021-12-22' '2021-12-21' '2021-12-18' '2021-12-16' '2021-12-14'  
 '2021-12-13' '2021-12-11' '2021-12-10' '2021-12-09' '2021-12-08'  
 '2021-12-07' '2021-12-06' '2021-12-04']  
  
Column 'numBedrooms' unique values  
[ 3 4 6 2 5 7 1 9 8 13 10 16 14 12 23]  
  
Column 'numBathrooms' unique values  
[ 3 2 1 6 4 5 7 13 8 12 10 9 11 23]  
  
Column 'propertyType' unique values  
['End of Terrace' 'Semi-D' 'Terrace' 'Detached' 'Apartment' 'Bungalow'  
 'Townhouse' 'Duplex' 'Site' 'Studio' 'House']  
  
Column 'category' unique values  
['Buy' 'New Homes']  
  
Column 'AMV\_price' unique values  
[0 1]  
  
Column 'seller\_name' unique values  
['Roy Dennehy' 'Marie Kiernan' 'Moovingo' ... 'Rooney Auctioneers'  
 "Paul O'Shea" 'Jackie Horan']  
  
Column 'sellerType' unique values  
['BRANDED\_AGENT' 'UNBRANDED\_AGENT' 'PRIVATE\_USER']  
  
Column 'm\_totalImages' unique values  
[ 16. 33. 38. 22. 5. 20. 37. 21. 24. 15. 28. 14. 43. 25.  
 29. 42. 26. 8. 18. 27. 17. 13. 32. 12. 9. 23. 48. 31.  
 30. 35. 19. 40. 7. 39. 11. 34. 2. 10. 36. 4. 6. 1.  
 3. 51. 0. 50. 60. 44. 56. 63. 52. 47. 55. 46. 53. 41.  
 68. 71. 45. 66. 57. 69. 54. 104. 67. 59. 49. 93. 81. 75.  
 88. 80. 65. 87.]  
  
Column 'm\_hasVideo' unique values  
[False True]  
  
Column 'm\_hasVirtualTour' unique values  
[False True]

### Boxplots

First of all, let's take make a boxplot for our targer varieble "price". However, the outliers may affect clear understanding, so I would recommend to update this and other charts after running preparation section. As we can see in updated boxplot:

* the mean is between 200k and 300k approximately in the middle, that indicates a value of 250k.
* the second and third quartiles which include the majority of data are placed between 200k and 375k more or less. It gives us a range of prices for most of the properties.
* also we can see first and fourth quatiles which show us the whole range of prices except outliers left on the top of boxblot.

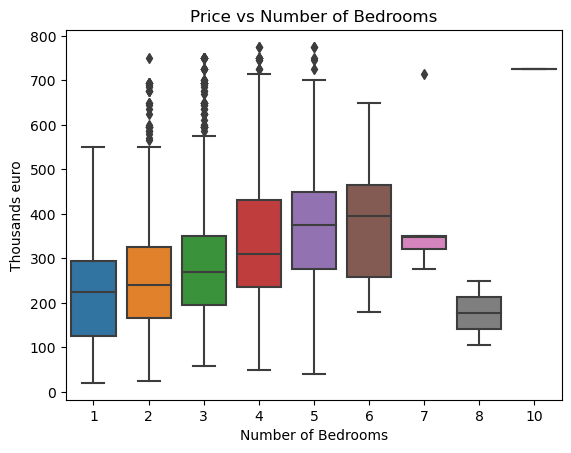
prices\_boxplot = df['price'] / 1000  
  
plt.figure(figsize=(10, 5))  
boxplot = plt.boxplot(prices\_boxplot, vert=True, patch\_artist=True)  
plt.ylabel('Price in thousands')  
plt.show()



We can see that two boxplots below show the distribution for properties with different number of badrooms and bathrooms. It is seen how price is changing according to the number of rooms.

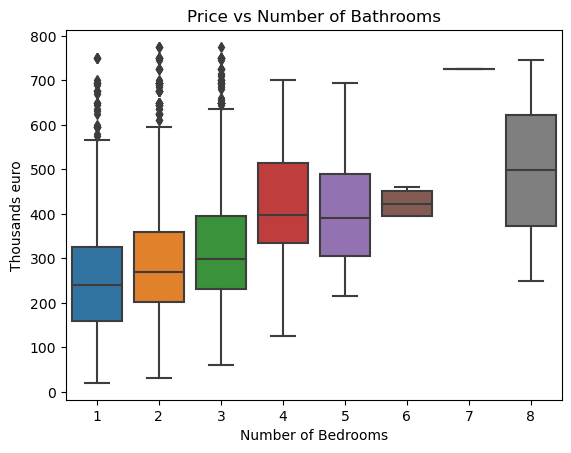
sns.boxplot(df.numBedrooms, prices\_boxplot)   
plt.title('Price vs Number of Bedrooms')  
plt.ylabel('Thousands euro')  
plt.xlabel('Number of Bedrooms')

Text(0.5, 0, 'Number of Bedrooms')



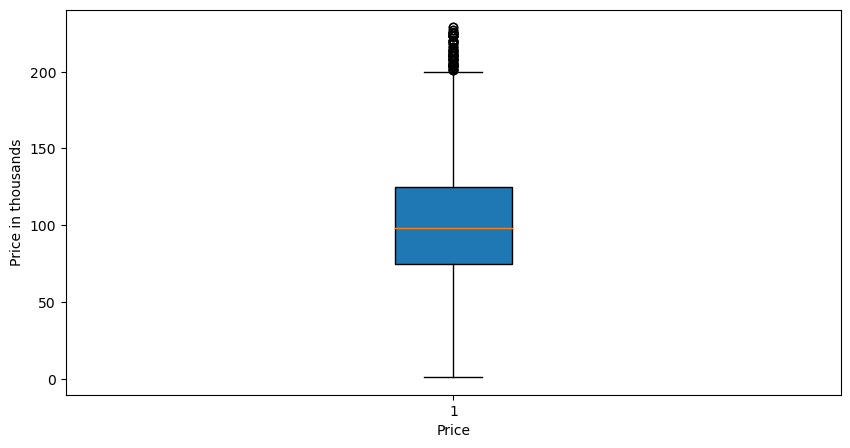
sns.boxplot(df.numBathrooms, prices\_boxplot)   
plt.title('Price vs Number of Bathrooms')  
plt.ylabel('Thousands euro')  
plt.xlabel('Number of Bedrooms')

Text(0.5, 0, 'Number of Bedrooms')



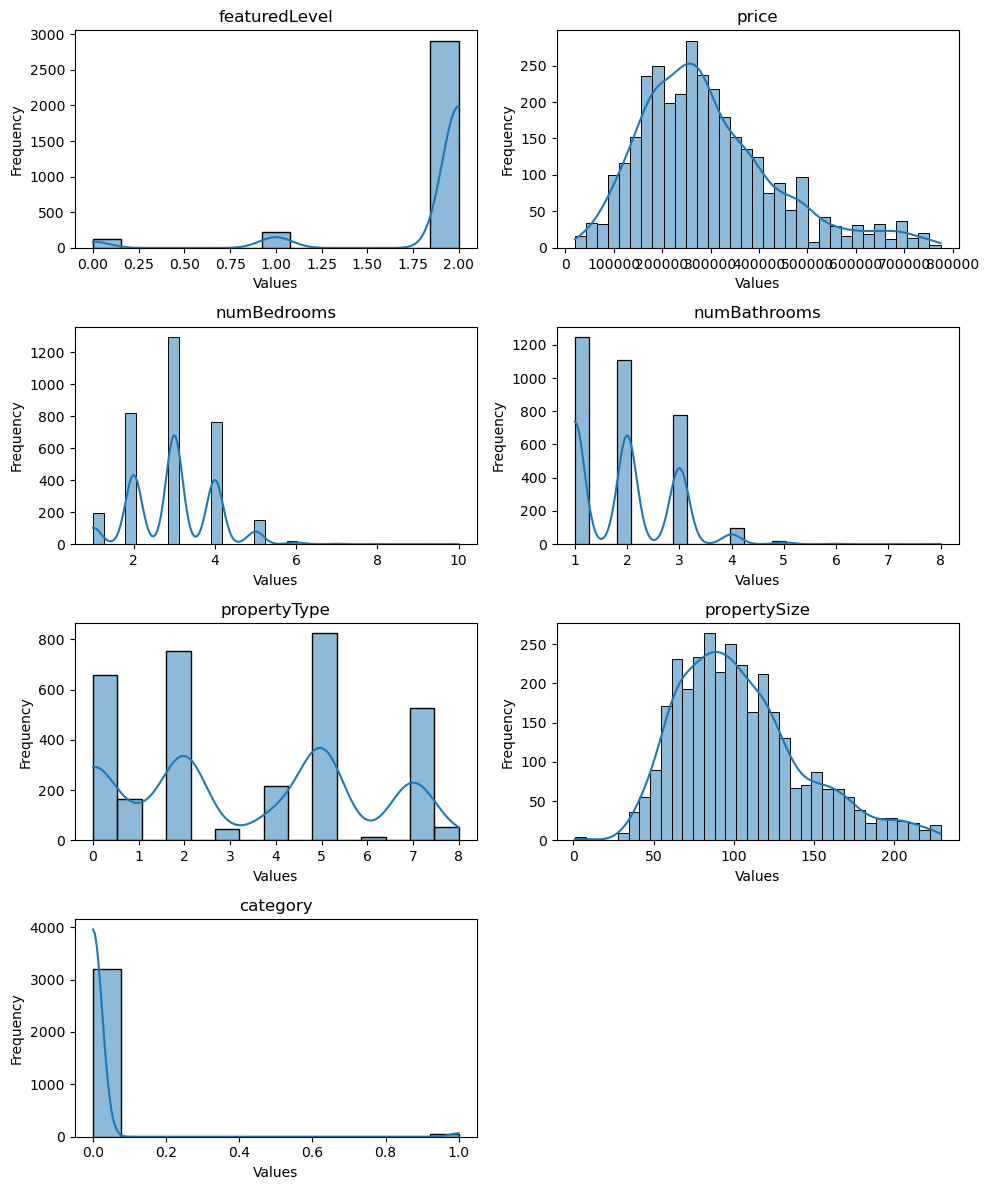
As we can see the avarage property size is 100. Additionally, the size for the most properties in range between 75 and 125.

prices\_boxplot = df['propertySize']  
  
plt.figure(figsize=(10, 5))  
boxplot = plt.boxplot(prices\_boxplot, vert=True, patch\_artist=True)  
plt.ylabel('Property size')  
plt.show()



### Histograms

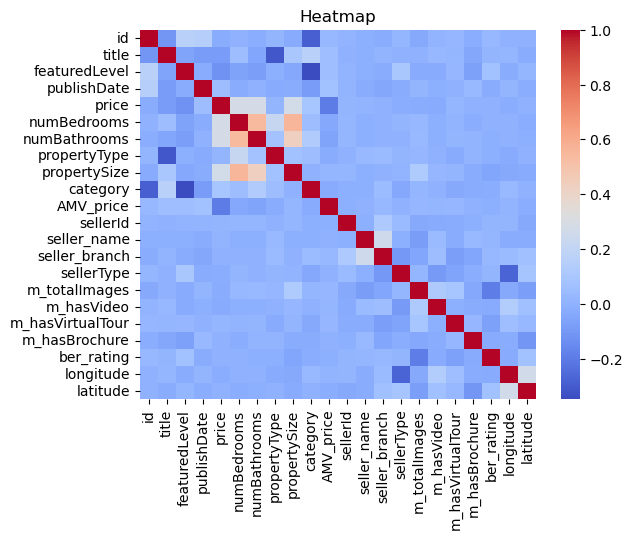
numeric\_index = [2, 4, 5, 6, 7, 8, 9]  
selected\_columns = df.iloc[:, numeric\_index]  
  
num\_cols = 2   
  
num\_features = len(numeric\_index)  
num\_rows = (num\_features - 1) // num\_cols + 1  
  
fig, axes = plt.subplots(num\_rows, num\_cols, figsize=(10, 3 \* num\_rows))  
  
axes = axes.flatten()  
  
for i, col\_index in enumerate(numeric\_index):  
 ax = axes[i]  
 col\_name = df.columns[col\_index]  
 sns.histplot(df.iloc[:, col\_index], ax=ax, kde=True)  
 ax.set\_title(col\_name)  
 ax.set\_xlabel('Values')  
 ax.set\_ylabel('Frequency')  
  
for i in range(num\_features, num\_rows \* num\_cols):  
 fig.delaxes(axes[i])  
   
plt.tight\_layout()  
plt.show()



### Heatmap

A heatmap shows correlation between all values in dataset. We can see that there are 4 feachers which have biggest impact on our target variable "price" such as number of bedrooms, number of bathrooms and property size.

sns.heatmap(df\_encoded.corr(), cmap='coolwarm')  
plt.title('Heatmap')  
plt.show()



## Data preparation

#### Drop missing values

df = df.dropna()

df.isnull().sum()

id 0  
title 0  
featuredLevel 0  
publishDate 0  
price 0  
numBedrooms 0  
numBathrooms 0  
propertyType 0  
propertySize 0  
category 0  
AMV\_price 0  
sellerId 0  
seller\_name 0  
seller\_branch 0  
sellerType 0  
m\_totalImages 0  
m\_hasVideo 0  
m\_hasVirtualTour 0  
m\_hasBrochure 0  
ber\_rating 0  
longitude 0  
latitude 0  
dtype: int64

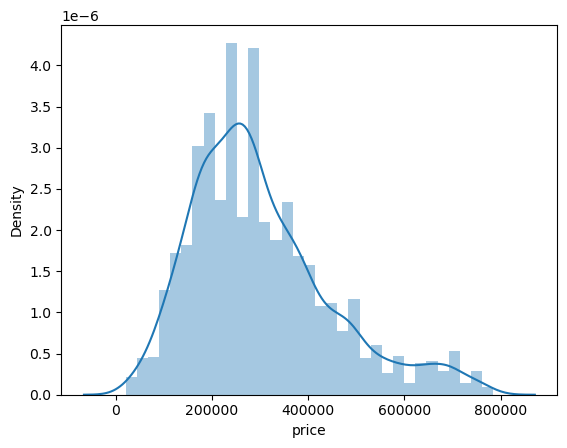
#### Outliers

##### Define percentage of dataset we want to keep which doesn't include outliers, in this case we keep 95% and cut just 5% whith outliers.

outlier = df.price.quantile(0.95)  
df = df.loc[df.price < outlier, :]

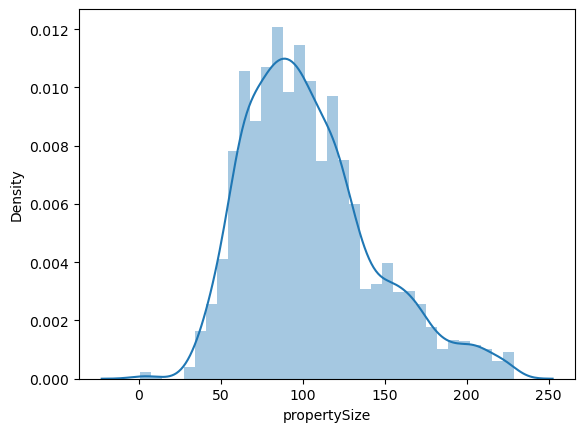
sns.distplot(df.price)

<AxesSubplot:xlabel='price', ylabel='Density'>



sns.distplot(df.propertySize)

<AxesSubplot:xlabel='propertySize', ylabel='Density'>



outlier = df.propertySize.quantile(0.95)  
df = df.loc[df.propertySize < outlier, :]

#### Encoding

label\_encoder = LabelEncoder()  
df\_encoded = df  
  
df\_encoded['title'] = label\_encoder.fit\_transform(df['title'])  
df\_encoded['featuredLevel'] = label\_encoder.fit\_transform(df['featuredLevel'])  
df\_encoded['propertyType'] = label\_encoder.fit\_transform(df['propertyType'])  
df\_encoded['publishDate'] = label\_encoder.fit\_transform(df['publishDate'])  
df\_encoded['category'] = label\_encoder.fit\_transform(df['category'])  
df\_encoded['seller\_name'] = label\_encoder.fit\_transform(df['seller\_name'])  
df\_encoded['seller\_branch'] = label\_encoder.fit\_transform(df['seller\_branch'])  
df\_encoded['sellerType'] = label\_encoder.fit\_transform(df['sellerType'])  
df\_encoded['m\_hasVideo'] = label\_encoder.fit\_transform(df['m\_hasVideo'])  
df\_encoded['m\_hasVirtualTour'] = label\_encoder.fit\_transform(df['m\_hasVirtualTour'])  
df\_encoded['m\_hasBrochure'] = label\_encoder.fit\_transform(df['m\_hasBrochure'])  
df\_encoded['ber\_rating'] = label\_encoder.fit\_transform(df['ber\_rating'])  
df\_encoded['longitude'] = label\_encoder.fit\_transform(df['longitude'])  
df\_encoded['latitude'] = label\_encoder.fit\_transform(df['latitude'])

## PCA

#### Scale dataset

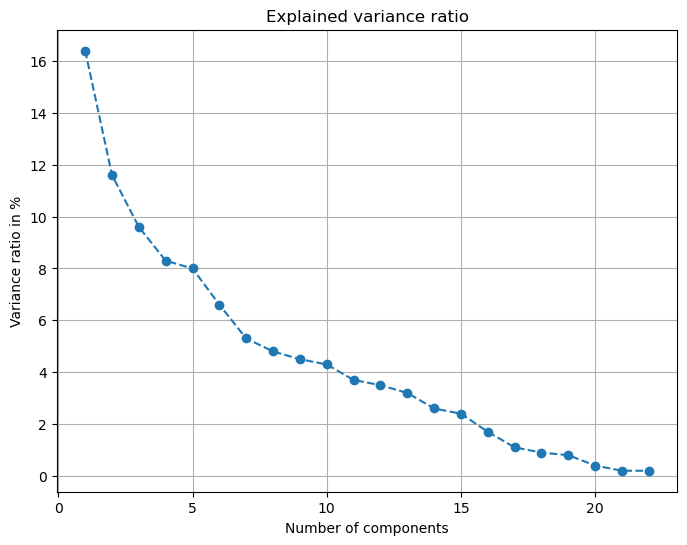
df\_scaled = scaler.fit\_transform(df\_encoded)

#### Create pca object

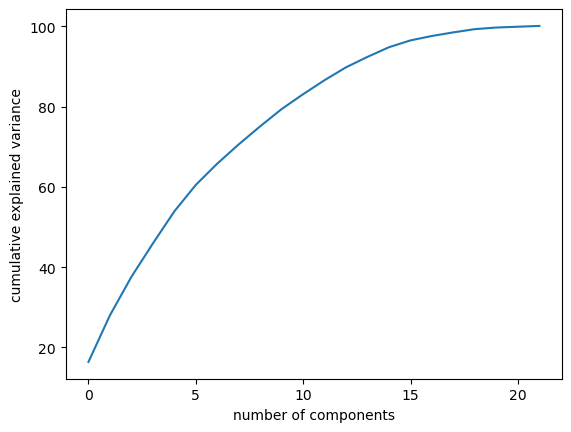
pca = PCA()  
pca.fit(df\_scaled)  
pca\_data = pca.transform(df\_scaled)

#### Plot varietion ratio for each component

variations = np.round(pca.explained\_variance\_ratio\_\* 100, decimals=1)  
labels = [str(x) for x in range (1, len(variations)+1)]  
  
plt.figure(figsize=(8, 6))  
plt.plot(range(1, len(variations) + 1),   
 variations, marker='o', linestyle='--')  
plt.xlabel('Number of components')  
plt.ylabel('Variance ratio in %')  
plt.title('Explained variance ratio')  
plt.grid(True)  
plt.show()



plt.plot(np.cumsum(variations))   
plt.xlabel('number of components')   
plt.ylabel('cumulative explained variance');



#### Calculate number of features which represent a sample with selected accuracy

variance\_ratio\_percentage = 95  
variance\_ratio = np.cumsum(variations)  
num\_components = np.argmax(variance\_ratio >= variance\_ratio\_percentage) + 1  
print("Percentage of variance:", variance\_ratio\_percentage)  
print("Number of components:", num\_components)

Percentage of variance: 95  
Number of components: 16

#### List features and their ratio from calculated number

abs\_loadings = np.abs(pca.components\_)  
importance\_scores = np.sum(abs\_loadings, axis=0)  
variance\_ratios = variations  
  
feature\_variance = {}  
for i, (feature\_name, variance\_ratio) in enumerate(zip(df.columns, variance\_ratios)):  
 feature\_variance[i, feature\_name] = variance\_ratio  
  
sorted\_feature\_variance = dict(sorted(feature\_variance.items(), key=lambda item: item[1], reverse=True))  
  
most\_important\_features = list(sorted\_feature\_variance.items())[:num\_components]  
  
for (index, feature), variance\_ratio in most\_important\_features:  
 print(f"Index: {index}, Feature: {feature}, Variance Ratio: {variance\_ratio:.1f}%")

Index: 0, Feature: id, Variance Ratio: 16.4%  
Index: 1, Feature: title, Variance Ratio: 11.6%  
Index: 2, Feature: featuredLevel, Variance Ratio: 9.6%  
Index: 3, Feature: publishDate, Variance Ratio: 8.3%  
Index: 4, Feature: price, Variance Ratio: 8.0%  
Index: 5, Feature: numBedrooms, Variance Ratio: 6.6%  
Index: 6, Feature: numBathrooms, Variance Ratio: 5.3%  
Index: 7, Feature: propertyType, Variance Ratio: 4.8%  
Index: 8, Feature: propertySize, Variance Ratio: 4.5%  
Index: 9, Feature: category, Variance Ratio: 4.3%  
Index: 10, Feature: AMV\_price, Variance Ratio: 3.7%  
Index: 11, Feature: sellerId, Variance Ratio: 3.5%  
Index: 12, Feature: seller\_name, Variance Ratio: 3.2%  
Index: 13, Feature: seller\_branch, Variance Ratio: 2.6%  
Index: 14, Feature: sellerType, Variance Ratio: 2.4%  
Index: 15, Feature: m\_totalImages, Variance Ratio: 1.7%

#### Defining a target

X=df\_encoded.drop(columns=['price'],axis = 1)  
y=df\_encoded['price']

#### Data scalling

scaler = MinMaxScaler()  
X = scaler.fit\_transform(X)  
y = scaler.fit\_transform(y.values.reshape(-1, 1))

#### Data splitting

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

### Hyperparameter Tuning with XDBoost

Each machine learning model has their own parameters by default. Hyperparameter tuning allows us to set multiple values of each parameter thereby creating new modified models. The more parameters the model has the more variation we can create and track the best parameters for our model. As a result, we should get some information about which combination of parameters gives us the best accuracy. So, after we will be able to change the default parameters for our model.

Firstly, let's define XGBoost model.

xgb\_model = XGBRegressor(random\_state=42)

And set the parameters we want to test. As we can see I have set the:

* 5 values for parameter number of estimators
* 3 values for depth
* 5 values for gamma Note: parameters for grid search tuning can be found in the documentation for the model in the library website, such as this: <https://xgboost.readthedocs.io/en/stable/tutorials/param_tuning.html> As a result, we should get 75 new models.

hyp\_params = {  
 "n\_estimators" : [0.1,1,10,100,1000],  
 "max\_depth" : [3,4,5],  
 "gamma" : [1,0.1,0.01,0.001,0.001]  
}

After parametrs for grid search have been set we can create a GridSearch model and include in it XGBoost model with different parameters we set before

GS = GridSearchCV(estimator = xgb\_model,   
 param\_grid = hyp\_params,  
 scoring = ["r2", 'neg\_root\_mean\_squared\_error'],  
 refit = "r2",  
 cv = 5,  
 verbose = 4)

Train the Grid Search model and get report with accuracy for each combination of parameters.

GS.fit(X\_train, y\_train)

Fitting 5 folds for each of 75 candidates, totalling 375 fits  
[CV 1/5] END gamma=1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.191) r2: (test=0.055) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.058) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.057) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.051) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.070) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.140) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.136) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.140) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.136) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.140) total time= 0.1s  
[CV 2/5] END gamma=1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.1s  
[CV 3/5] END gamma=1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.1s  
[CV 4/5] END gamma=1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.136) total time= 0.2s  
[CV 5/5] END gamma=1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.1s  
[CV 1/5] END gamma=1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.191) r2: (test=0.059) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.058) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.057) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.066) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.070) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.143) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.143) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.143) total time= 0.1s  
[CV 2/5] END gamma=1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.2s  
[CV 3/5] END gamma=1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.1s  
[CV 4/5] END gamma=1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.1s  
[CV 5/5] END gamma=1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.2s  
[CV 1/5] END gamma=1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.191) r2: (test=0.059) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.058) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.057) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.066) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.070) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.143) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.143) total time= 0.0s  
[CV 2/5] END gamma=1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.0s  
[CV 3/5] END gamma=1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.0s  
[CV 4/5] END gamma=1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.0s  
[CV 5/5] END gamma=1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.0s  
[CV 1/5] END gamma=1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.143) total time= 0.3s  
[CV 2/5] END gamma=1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.143) total time= 0.3s  
[CV 3/5] END gamma=1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.167) total time= 0.1s  
[CV 4/5] END gamma=1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.1s  
[CV 5/5] END gamma=1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.146) total time= 0.2s  
[CV 1/5] END gamma=0.1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.190) r2: (test=0.069) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.063) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.062) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.047) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.228) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.159) r2: (test=0.199) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.212) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.183) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.173) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.242) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.220) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.194) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.169) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.242) total time= 0.2s  
[CV 2/5] END gamma=0.1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.2s  
[CV 3/5] END gamma=0.1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.220) total time= 0.1s  
[CV 4/5] END gamma=0.1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.165) r2: (test=0.194) total time= 0.2s  
[CV 5/5] END gamma=0.1, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.169) total time= 0.3s  
[CV 1/5] END gamma=0.1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.189) r2: (test=0.081) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.071) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.072) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.063) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.075) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.243) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.181) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.167) r2: (test=0.173) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.150) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.243) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.185) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.171) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.150) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.243) total time= 0.3s  
[CV 2/5] END gamma=0.1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.2s  
[CV 3/5] END gamma=0.1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.185) total time= 0.2s  
[CV 4/5] END gamma=0.1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.171) total time= 0.3s  
[CV 5/5] END gamma=0.1, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.150) total time= 0.2s  
[CV 1/5] END gamma=0.1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.188) r2: (test=0.084) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.086) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.084) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.072) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.084) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.243) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.195) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.169) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.160) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.243) total time= 0.0s  
[CV 2/5] END gamma=0.1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.0s  
[CV 3/5] END gamma=0.1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.195) total time= 0.0s  
[CV 4/5] END gamma=0.1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.169) total time= 0.0s  
[CV 5/5] END gamma=0.1, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.160) total time= 0.0s  
[CV 1/5] END gamma=0.1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.243) total time= 0.2s  
[CV 2/5] END gamma=0.1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.157) r2: (test=0.218) total time= 0.3s  
[CV 3/5] END gamma=0.1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.195) total time= 0.2s  
[CV 4/5] END gamma=0.1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.169) total time= 0.3s  
[CV 5/5] END gamma=0.1, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.160) total time= 0.3s  
[CV 1/5] END gamma=0.01, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.190) r2: (test=0.069) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.063) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.062) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.047) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.231) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.159) r2: (test=0.198) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.167) r2: (test=0.202) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.184) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.171) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.175) r2: (test=0.208) total time= 0.1s  
[CV 2/5] END gamma=0.01, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.161) r2: (test=0.180) total time= 0.1s  
[CV 3/5] END gamma=0.01, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.156) total time= 0.1s  
[CV 4/5] END gamma=0.01, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.155) total time= 0.1s  
[CV 5/5] END gamma=0.01, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.175) r2: (test=0.122) total time= 0.1s  
[CV 1/5] END gamma=0.01, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.175) r2: (test=0.208) total time= 0.3s  
[CV 2/5] END gamma=0.01, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.161) r2: (test=0.180) total time= 0.2s  
[CV 3/5] END gamma=0.01, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.158) total time= 0.1s  
[CV 4/5] END gamma=0.01, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.155) total time= 0.1s  
[CV 5/5] END gamma=0.01, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.175) r2: (test=0.122) total time= 0.1s  
[CV 1/5] END gamma=0.01, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.189) r2: (test=0.079) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.072) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.064) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.074) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.248) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.158) r2: (test=0.209) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.172) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.187) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.147) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.174) r2: (test=0.215) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.159) r2: (test=0.201) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.174) r2: (test=0.138) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.153) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.096) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.174) r2: (test=0.215) total time= 0.1s  
[CV 2/5] END gamma=0.01, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.159) r2: (test=0.201) total time= 0.1s  
[CV 3/5] END gamma=0.01, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.174) r2: (test=0.138) total time= 0.1s  
[CV 4/5] END gamma=0.01, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.153) total time= 0.1s  
[CV 5/5] END gamma=0.01, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.096) total time= 0.1s  
[CV 1/5] END gamma=0.01, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.188) r2: (test=0.083) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.087) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.082) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.069) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.081) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.174) r2: (test=0.215) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.160) r2: (test=0.193) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.182) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.159) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.151) total time= 0.0s  
[CV 1/5] END gamma=0.01, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.194) total time= 0.0s  
[CV 2/5] END gamma=0.01, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.160) r2: (test=0.191) total time= 0.0s  
[CV 3/5] END gamma=0.01, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.115) total time= 0.0s  
[CV 4/5] END gamma=0.01, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.123) total time= 0.0s  
[CV 5/5] END gamma=0.01, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.113) total time= 0.1s  
[CV 1/5] END gamma=0.01, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.194) total time= 0.1s  
[CV 2/5] END gamma=0.01, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.160) r2: (test=0.191) total time= 0.1s  
[CV 3/5] END gamma=0.01, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.115) total time= 0.2s  
[CV 4/5] END gamma=0.01, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.123) total time= 0.1s  
[CV 5/5] END gamma=0.01, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.113) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.190) r2: (test=0.069) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.063) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.062) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.047) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.231) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.159) r2: (test=0.198) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.201) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.184) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.171) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.229) total time= 0.1s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.164) r2: (test=0.155) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.171) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.109) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.140) total time= 0.2s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.013) total time= 0.2s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.069) total time= 0.2s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.016) total time= 0.2s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.046) total time= 0.2s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.189) r2: (test=0.079) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.072) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.064) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.074) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.248) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.158) r2: (test=0.209) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.172) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.187) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.147) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.184) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.132) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.062) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.089) total time= 0.1s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.050) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.184) r2: (test=0.126) total time= 0.1s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.094) total time= 0.2s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.187) r2: (test=0.007) total time= 0.2s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.028) total time= 0.2s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.186) r2: (test=0.005) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.188) r2: (test=0.083) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.087) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.081) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.069) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.081) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.175) r2: (test=0.213) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.160) r2: (test=0.191) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.158) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.167) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.150) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.176) total time= 0.1s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.167) r2: (test=0.123) total time= 0.1s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.055) total time= 0.1s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.082) total time= 0.1s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.072) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.171) total time= 0.2s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.103) total time= 0.2s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.057) total time= 0.2s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.070) total time= 0.1s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.070) total time= 0.2s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.190) r2: (test=0.069) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.063) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.062) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.047) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.231) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.159) r2: (test=0.198) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.201) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.184) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.171) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.173) r2: (test=0.229) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.164) r2: (test=0.155) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.171) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.151) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.109) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.140) total time= 0.2s  
[CV 2/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.013) total time= 0.2s  
[CV 3/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.069) total time= 0.2s  
[CV 4/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.183) r2: (test=0.016) total time= 0.2s  
[CV 5/5] END gamma=0.001, max\_depth=3, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.046) total time= 0.2s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.189) r2: (test=0.079) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.072) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.073) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.064) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.074) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.248) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.158) r2: (test=0.209) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.171) r2: (test=0.172) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.187) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.147) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.184) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.166) r2: (test=0.132) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.062) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.089) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.050) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.184) r2: (test=0.126) total time= 0.2s  
[CV 2/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.094) total time= 0.2s  
[CV 3/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.187) r2: (test=0.007) total time= 0.2s  
[CV 4/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.181) r2: (test=0.028) total time= 0.2s  
[CV 5/5] END gamma=0.001, max\_depth=4, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.186) r2: (test=0.005) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=0.1; neg\_root\_mean\_squared\_error: (test=nan) r2: (test=nan) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.188) r2: (test=0.083) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.170) r2: (test=0.087) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.081) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.178) r2: (test=0.069) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=1; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.081) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.175) r2: (test=0.213) total time= 0.0s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.160) r2: (test=0.191) total time= 0.0s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.158) total time= 0.0s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.168) r2: (test=0.167) total time= 0.0s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=10; neg\_root\_mean\_squared\_error: (test=-0.172) r2: (test=0.150) total time= 0.0s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.176) total time= 0.1s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.167) r2: (test=0.123) total time= 0.1s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.055) total time= 0.1s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.176) r2: (test=0.082) total time= 0.1s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=100; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.072) total time= 0.1s  
[CV 1/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.179) r2: (test=0.171) total time= 0.1s  
[CV 2/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.169) r2: (test=0.103) total time= 0.2s  
[CV 3/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.182) r2: (test=0.057) total time= 0.1s  
[CV 4/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.177) r2: (test=0.070) total time= 0.2s  
[CV 5/5] END gamma=0.001, max\_depth=5, n\_estimators=1000; neg\_root\_mean\_squared\_error: (test=-0.180) r2: (test=0.070) total time= 0.1s

GridSearchCV(cv=5,  
 estimator=XGBRegressor(base\_score=None, booster=None,  
 callbacks=None, colsample\_bylevel=None,  
 colsample\_bynode=None,  
 colsample\_bytree=None, device=None,  
 early\_stopping\_rounds=None,  
 enable\_categorical=False, eval\_metric=None,  
 feature\_types=None, gamma=None,  
 grow\_policy=None, importance\_type=None,  
 interaction\_constraints=None,  
 learning\_rate=None, m...  
 max\_depth=None, max\_leaves=None,  
 min\_child\_weight=None, missing=nan,  
 monotone\_constraints=None,  
 multi\_strategy=None, n\_estimators=None,  
 n\_jobs=None, num\_parallel\_tree=None,  
 random\_state=42, ...),  
 param\_grid={'gamma': [1, 0.1, 0.01, 0.001, 0.001],  
 'max\_depth': [3, 4, 5],  
 'n\_estimators': [0.1, 1, 10, 100, 1000]},  
 refit='r2', scoring=['r2', 'neg\_root\_mean\_squared\_error'],  
 verbose=4)

Also we can display the best setting and accuracy score for it.

print(GS.best\_estimator\_)

XGBRegressor(base\_score=None, booster=None, callbacks=None,  
 colsample\_bylevel=None, colsample\_bynode=None,  
 colsample\_bytree=None, device=None, early\_stopping\_rounds=None,  
 enable\_categorical=False, eval\_metric=None, feature\_types=None,  
 gamma=0.1, grow\_policy=None, importance\_type=None,  
 interaction\_constraints=None, learning\_rate=None, max\_bin=None,  
 max\_cat\_threshold=None, max\_cat\_to\_onehot=None,  
 max\_delta\_step=None, max\_depth=3, max\_leaves=None,  
 min\_child\_weight=None, missing=nan, monotone\_constraints=None,  
 multi\_strategy=None, n\_estimators=100, n\_jobs=None,  
 num\_parallel\_tree=None, random\_state=42, ...)

print(GS.best\_params\_)

{'gamma': 0.1, 'max\_depth': 3, 'n\_estimators': 100}

print(GS.best\_score\_)

0.20867758599096167

## Multiple liniar regression

In this case we are going apply Liniar Regression model for target variable using GridSearch. As I mentioned before most of the models have their own hyperparameter, especially if the models focus on different problems such as Classification or Regression.

model = LinearRegression()

hyp\_params = {'fit\_intercept': [True, False]}

GS2 = GridSearchCV(estimator = model,   
 param\_grid = hyp\_params,  
 scoring = ["r2", 'neg\_root\_mean\_squared\_error'],  
 refit = "r2",  
 cv = 5,  
 verbose = 4)

GS2.fit(X\_train, y\_train)

Fitting 5 folds for each of 2 candidates, totalling 10 fits  
[CV 1/5] END fit\_intercept=True; neg\_root\_mean\_squared\_error: (test=-132523.136) r2: (test=0.207) total time= 0.1s  
[CV 2/5] END fit\_intercept=True; neg\_root\_mean\_squared\_error: (test=-127775.333) r2: (test=0.146) total time= 0.0s  
[CV 3/5] END fit\_intercept=True; neg\_root\_mean\_squared\_error: (test=-123930.046) r2: (test=0.210) total time= 0.0s  
[CV 4/5] END fit\_intercept=True; neg\_root\_mean\_squared\_error: (test=-128042.760) r2: (test=0.183) total time= 0.0s  
[CV 5/5] END fit\_intercept=True; neg\_root\_mean\_squared\_error: (test=-127389.672) r2: (test=0.133) total time= 0.0s  
[CV 1/5] END fit\_intercept=False; neg\_root\_mean\_squared\_error: (test=-134102.510) r2: (test=0.188) total time= 0.0s  
[CV 2/5] END fit\_intercept=False; neg\_root\_mean\_squared\_error: (test=-128401.313) r2: (test=0.138) total time= 0.0s  
[CV 3/5] END fit\_intercept=False; neg\_root\_mean\_squared\_error: (test=-124113.139) r2: (test=0.208) total time= 0.0s  
[CV 4/5] END fit\_intercept=False; neg\_root\_mean\_squared\_error: (test=-128092.538) r2: (test=0.182) total time= 0.0s  
[CV 5/5] END fit\_intercept=False; neg\_root\_mean\_squared\_error: (test=-126806.414) r2: (test=0.141) total time= 0.0s

GridSearchCV(cv=5, estimator=LinearRegression(),  
 param\_grid={'fit\_intercept': [True, False]}, refit='r2',  
 scoring=['r2', 'neg\_root\_mean\_squared\_error'], verbose=4)

print(GS2.best\_estimator\_)

LinearRegression()

print(GS2.best\_params\_)

{'fit\_intercept': True}

print(GS2.best\_score\_)

0.17586233165784135

#### Cross validation, results test and avarage

Also, very useful tool is cross validation. Generally speaking, it chenges train and test parts thereby displays more accurate R2 score by calculating the mean from all tests.

cv\_scores = cross\_val\_score(model, X, y, cv=10)  
  
print("r2\_score per test:", cv\_scores)  
  
mean\_r2\_score = cv\_scores.mean()  
print("Avarage r2\_score:", mean\_r2\_score)

r2\_score per test: [0.11331503 0.24261559 0.0832419 0.23481264 0.10426145 0.09375017  
 0.18174136 0.1393357 0.16997269 0.16572329]  
Avarage r2\_score: 0.15287698171327208

## Decision Tree (Classifier)

In order to understand how some categorical features correlate with others we are going to test few of them using models for classification. In this case we will apply Decision Tree Classifier model to predict a categorical feacher "numBedrooms" based on selected features.

#### Feature selection

Because we are focusing on a different target in this model we need to define a new target and split data accordingly.

selected\_columns = ['propertySize', 'numBathrooms', 'propertyType', 'price', 'category']  
X = df[selected\_columns]  
y = df.numBedrooms

#### Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 42)

#### Create a model and train it

model\_tree = DecisionTreeClassifier(criterion='gini', max\_depth = 5, random\_state = 42)  
model\_tree = model\_tree.fit(X\_train, y\_train)  
y\_pred = model\_tree.predict(X\_test)

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

Accuracy: 0.6441717791411042

X.columns.tolist()

['propertySize', 'numBathrooms', 'propertyType', 'price', 'category']

df['numBedrooms'].unique().tolist()

[3, 4, 2, 5, 6, 1, 7, 8, 10]

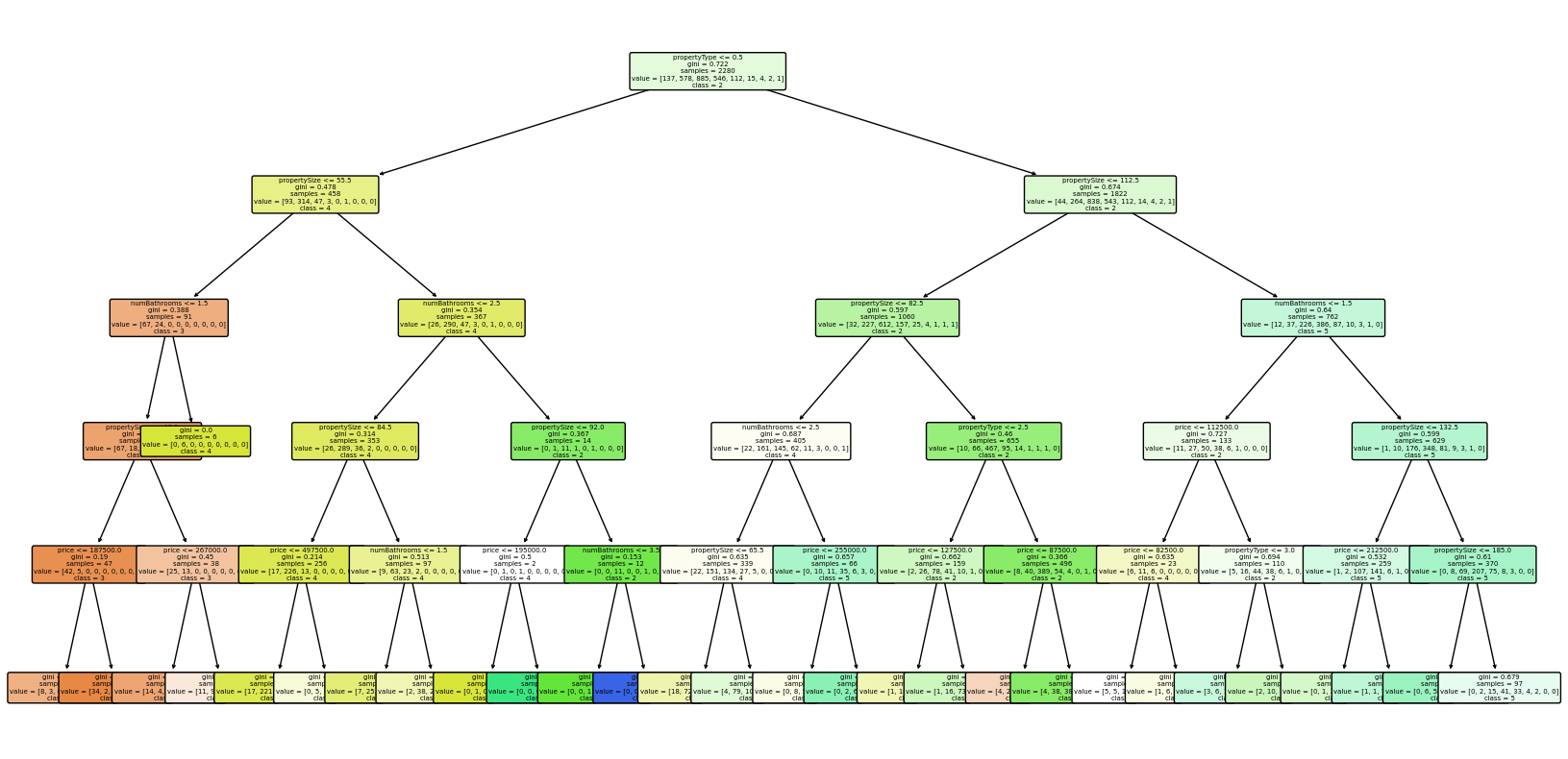
#### Test prediction

model\_tree.predict([[162, 4, 6, 300000, 0]])

array([4])

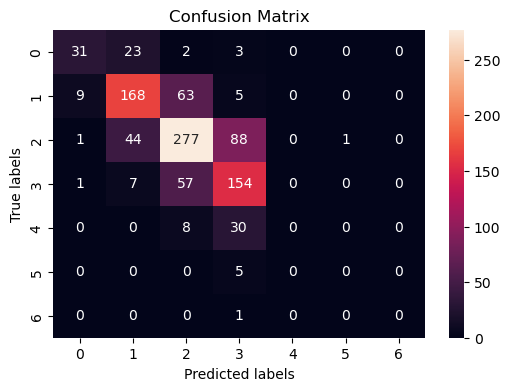
#### Visualisation Tree

plt.figure(figsize=(20, 10))  
plot\_tree(model\_tree, filled=True, feature\_names=X.columns.tolist(), class\_names=[str(label) for label in df['numBedrooms'].unique()], rounded=True, fontsize=5)  
plt.show()



#### Visualisation Confusion matrix

y\_predict = model\_tree.predict(X\_test)  
  
cm = confusion\_matrix(y\_test, y\_predict)  
  
plt.figure(figsize=(6, 4))  
sns.heatmap(cm, annot=True, fmt='d')  
plt.xlabel('Predicted labels')  
plt.ylabel('True labels')  
plt.title('Confusion Matrix')  
plt.show()  
  
report = classification\_report(y\_test, y\_predict)  
print("Classification Report:")  
print(report)



Classification Report:  
 precision recall f1-score support  
  
 1 0.74 0.53 0.61 59  
 2 0.69 0.69 0.69 245  
 3 0.68 0.67 0.68 411  
 4 0.54 0.70 0.61 219  
 5 0.00 0.00 0.00 38  
 6 0.00 0.00 0.00 5  
 7 0.00 0.00 0.00 1  
  
 accuracy 0.64 978  
 macro avg 0.38 0.37 0.37 978  
weighted avg 0.63 0.64 0.63 978

## Decision Tree (Regressor)

To predict price of properties based on selected features we are going to use the same decision tree algorithm but this time with regressor model due to our target variable is continuous.

X=df\_encoded.drop(columns=['price'],axis = 1)  
y=df\_encoded['price']

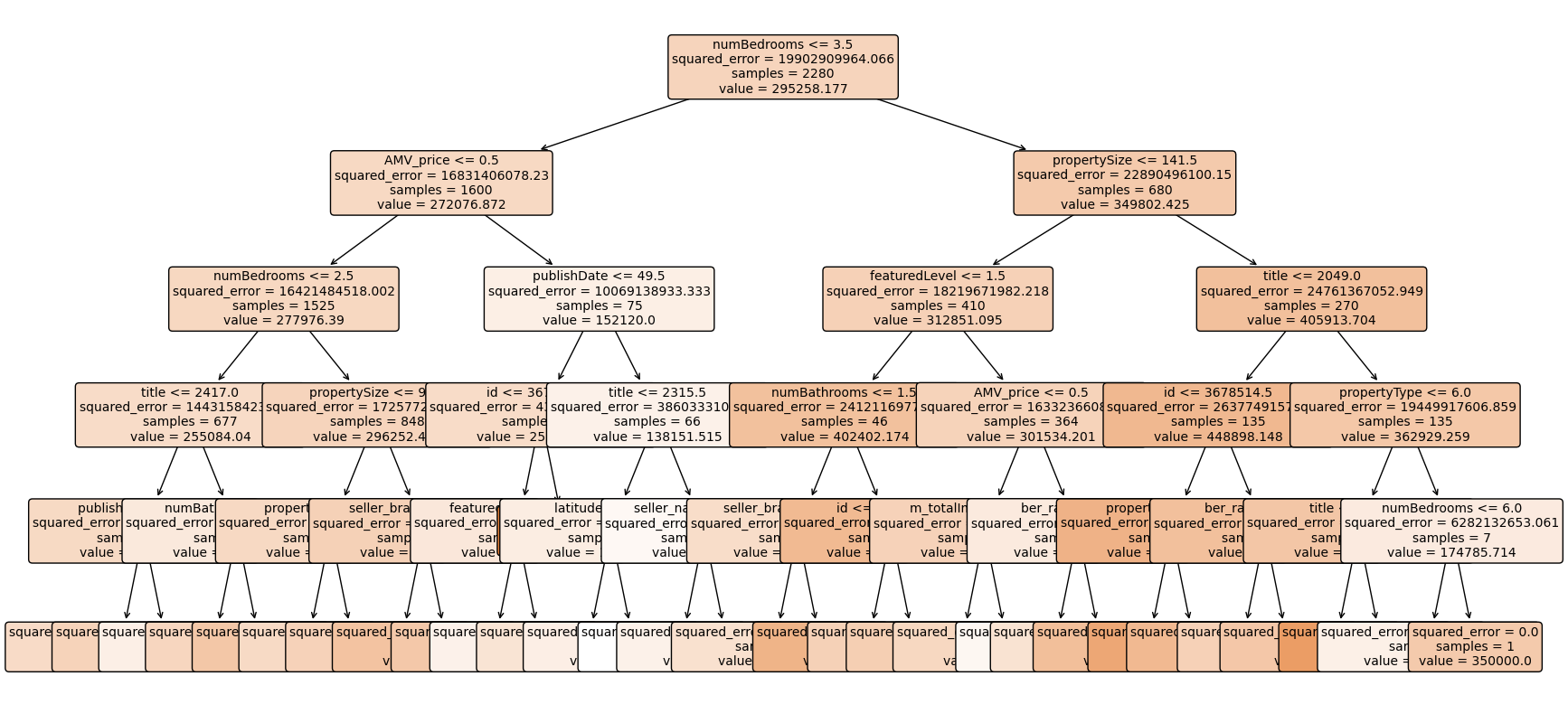
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 42)

model\_tree2 = DecisionTreeRegressor(max\_depth = 5, random\_state = 42)  
  
model\_tree2.fit(X\_train, y\_train)  
  
y\_pred = model\_tree2.predict(X\_test)  
  
print("R-squared:", mean\_r2\_score)

R-squared: 0.15287698171327208

#### Visualisation Tree

plt.figure(figsize=(20, 10))  
plot\_tree(model\_tree2, filled=True, feature\_names=X.columns.tolist(), class\_names=[str(label) for label in df['price'].unique().tolist()], rounded=True, fontsize=10)  
plt.show()



## Support Vector Machine

#### Feature selection

selected\_columns = ['propertySize', 'price', 'propertyType', 'numBathrooms', 'category']  
X = df[selected\_columns]  
y = df.numBedrooms

#### Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 42)

#### Create a model and train it

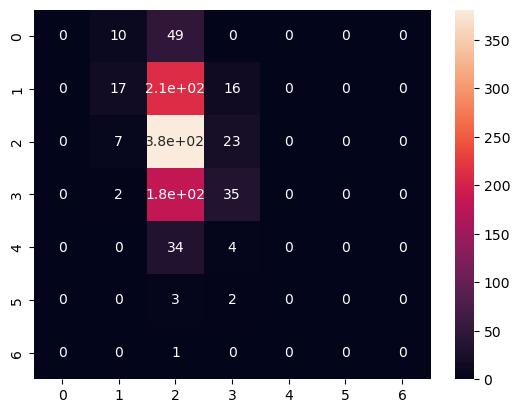
svc\_model = SVC()  
svc\_model.fit(X\_train, y\_train)

SVC()

y\_predict = svc\_model.predict(X\_test)  
cm = confusion\_matrix(y\_test, y\_predict)  
print(classification\_report(y\_test, y\_predict))  
sns.heatmap(cm, annot=True)

precision recall f1-score support  
  
 1 0.00 0.00 0.00 59  
 2 0.47 0.07 0.12 245  
 3 0.44 0.93 0.60 411  
 4 0.44 0.16 0.23 219  
 5 0.00 0.00 0.00 38  
 6 0.00 0.00 0.00 5  
 7 0.00 0.00 0.00 1  
  
 accuracy 0.44 978  
 macro avg 0.19 0.17 0.14 978  
weighted avg 0.40 0.44 0.33 978

<AxesSubplot:>



#### Test prediction

svc\_model.predict([[162, 575000, 6, 3, 0]])

array([4])

As we can see accuracy, micro average and weighted average have risen. Basically, we can conclude that using selected we are predicting only 51% of our sample. In this case, we defined properties with different numbers of bedrooms as the target variable. The most predictable are properties with 2 and 3 bedrooms.

## Random forest (Categoric target)

Random Forest is a powerful machine learning algorithm which works by creating multiple decision trees for training and outputs the average prediction for the each tree for regression tasks. The model creates multiple decision trees, and each tree is trained on a random subset of the training data,where each tree gets a random sample of the data with replacement. Additionally to use a random subset of the data for each tree, Random Forest also introduces randomness in the selection of features to split each node of the decision tree. Instead of considering all features for splitting, it randomly selects a subset of features at each node.

selected\_columns = ['propertySize', 'price', 'propertyType', 'numBathrooms', 'category']  
X = df[selected\_columns]  
y = df.numBedrooms  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 42)

#### Create a model

model\_rf = RandomForestClassifier(n\_estimators = 1000)

#### Train model and prediction

model\_rf.fit(X\_train, y\_train)  
y\_pred = model\_rf.predict(X\_test)

df.head()

id title featuredLevel publishDate price numBedrooms \  
0 3626025 178 0 50 290000 3   
1 3675175 1732 0 50 225000 3   
2 3673450 495 0 49 575000 4   
4 3643947 1592 0 50 120000 3   
5 3598816 1978 0 52 400000 4   
  
 numBathrooms propertyType propertySize category ... seller\_name \  
0 3 4 96.0 0 ... 1058   
1 2 6 93.0 0 ... 754   
2 3 6 162.0 0 ... 844   
4 1 9 68.0 0 ... 993   
5 3 6 113.0 0 ... 881   
  
 seller\_branch sellerType m\_totalImages m\_hasVideo m\_hasVirtualTour \  
0 199 0 16.0 0 0   
1 777 0 33.0 0 0   
2 475 0 38.0 0 1   
4 621 0 5.0 0 0   
5 671 0 20.0 1 0   
  
 m\_hasBrochure ber\_rating longitude latitude   
0 0 7 891 94   
1 0 6 1573 1188   
2 0 2 2379 1374   
4 0 14 1761 2981   
5 0 6 626 168   
  
[5 rows x 22 columns]

#### Evaluating the model

mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print("Mean Squared Error:", mse)  
print("R-squared:", r2)

Mean Squared Error: 0.724105461393597  
R-squared: 0.3894617070712809

#### Test prediction

model\_rf.predict([[162, 575000, 6, 3, 0]])

array([4])

## Random forest (Numerical target)

X=df\_encoded.drop(columns=['price'],axis = 1)  
y=df\_encoded['price']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 42)

#### Create Random Forest regressor

model\_rf2 = RandomForestRegressor(n\_estimators=1000, random\_state=42)

#### Train model and prediction

model\_rf2.fit(X\_train, y\_train)  
y\_pred = model\_rf2.predict(X\_test)

#### Evaluating the model

mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print("Mean Squared Error:", mse)  
print("R-squared:", r2)

Mean Squared Error: 17160048535.021908  
R-squared: 0.1748829972673075

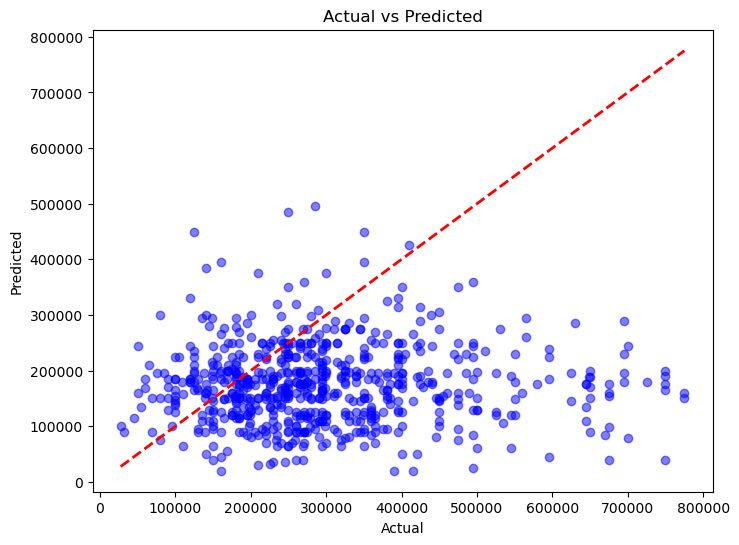
## K-nearest neighbors

When making a prediction for a new data point, KNN calculates the distance between the new data point and all other data points in the training set. The most common distance metric used is Euclidean distance, but other distance metrics can also be used depending on the problem.

X=df\_encoded.drop(columns=['price'],axis = 1)  
y=df\_encoded['price']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
knn\_model = KNeighborsClassifier(n\_neighbors=4)  
knn\_model.fit(X\_train, y\_train)  
y\_pred = knn\_model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print("Mean Squared Error", mse)  
print("R-squared", r2)

Mean Squared Error 40085091917.17792  
R-squared -0.9661081495146358

plt.figure(figsize=(8, 6))  
plt.scatter(y\_test, y\_pred, color='blue', alpha=0.5)  
plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], linestyle='--', color='red', linewidth=2)  
plt.title('Actual vs Predicted')  
plt.xlabel('Actual')  
plt.ylabel('Predicted')  
plt.show()



# References

1. Nowlan, B., (2015). Housing Supply in Ireland: Perennial Problems and Sustainable Solutions. Sunday Business Property Conference Paper 25.09.15
2. Scuffil, C., (2022). Dublin’s Housing Crisis in Troubled Times. Dublin City Council Libraries Blog. Accessed on 03.09.22 at <https://www.dublincity.ie/library/blog/dublins-housing-crisis-troubled-times>
3. Phan, T. D. (2019) ‘Housing price prediction using machine learning algorithms: The case of Melbourne city, Australia’, Proceedings - International Conference on Machine Learning and Data Engineering, iCMLDE 2018. IEEE, pp. 8–13. DOI: 10.1109/iCMLDE.2018.00017.
4. Norris, M. & Byrne, M. (2017) A tale of two busts (and a boom): Irish social housing before and after the global financial crisis, Critical Housing Analysis, 4, pp. 19–28.
5. Nowicki, M., Brickell, K. & Harris, E. (2019) The hotelisation of the housing crisis: Experiences of family homelessness in Dublin hotels, The Geographical Journal, 185, pp. 313–324.
6. Byrne, M. (2020) Generation rent and the financialization of housing: A comparative exploration of the growth of the private rental sector in Ireland, the UK and Spain, Housing Studies, 35, pp. 743–765.
7. Bourassa, S. C., Cantoni, E. and Hoesli, M. (2011) ‘Predicting House Prices with Spatial Dependence:Impacts of Alternative Submarket Definitions’, SSRN Electronic Journal. DOI: 10.2139/ssrn.1090147.
8. Aswin S. R (2017), ‘Real Estate Price Prediction Using Machine Learning’ <https://norma.ncirl.ie/3096/1/aswinsivamravikumar.pdf>
9. Hujia Yu and Jiafu Wu., (2016), ' Real Estate Price Prediction with Regression and Classification' <https://cs229.stanford.edu/proj2016/report/WuYu_HousingPrice_report.pdf>
10. Ng, A. (2015). Machine Learning for a London Housing Price Prediction Mobile Application. Imperial College London. <http://www.doc.ic.ac.uk/~mpd37/theses/2015_beng_aaron-ng.pdf>

### Author: Ilia Grishkin

#### <https://github.com/Ilia-Grishkin/Machine_learning>

### Change log

| Date | Change Description | Changed By | Status |
| --- | --- | --- | --- |
| 19.03.2024 | Created title, structure | Ilia |  |
| 20.03.2024 | Created intoduction | Ilia | Pushed |
| 21.03.2024 | Created literate review, content | Ilia | Pushed |
| 22.03.2024 | Add data sourse | Ilia | Pushed |
| 24.03.2024 | Methods selection | Ilia | Pushed |
| 24.03.2024 | Liniar regressions | Ilia | Pushed |
| 24.03.2024 | SVM | Ilia | Planned |
| 24.03.2024 | Random forest | Ilia | Pushed |
| 31.03.2024 | GridSearchCV | Ilia | Pushed |
| 01.04.2024 | KNN | Ilia | Pushed |