Math 342w Final Project Spring 2022

BY: Asher Katz

Abstract:

In this project, we attempted to model apartment sale price in mainland Queens in the year of 2016-2017. I sought to use industry standard technologies such as RandomForestReggressor, MissForest, Pandas and NumPy to develop my professional skills and bolster my self-esteem. The dataset which we were operating with was an MTurk harvested dataset from the time period with 2230 observations and many incomplete observations and missing data points. Our target variable was sale_price_$. We successfully cleaned up the dataset, saved it as a .csv file, and were able to open it in a new window for modeling the dataset. We fairly were successful at training many different models like DecisionTreeReggressors, OLS's and RandomForestReggressors on the cleaned dataset. These models performed well but they were not nearly sharp enough to use in real life.

Section 1:

This is a report on predictions which we made using python 3 libraries like pandas, numpy, DecisionTreeRegressor and RandomForestRegressor for apartment selling prices in Queens, NY. I used a dataset which was harvested with MTurk in 2016-17 which had many features about apartments with a maximum sale price of $1M which were sold between February 2016 and February 2017. The dataset was limited to the zip codes found on mainland Queens. It excludes the Rockaways, a peninsula near JFK airport that's geographically distinct from the rest of the neighborhoods. Our target variable was *sale_price_$* . I picked this project because I felt that I could do better than zillow.com, a website which generates their own secret-sauce predictions that they whimsically call "zestimates." The challenge of beating their estimates is that apartment prices are dependent on so many factors and determining the exact important of any one factor can be hard. If I could determine the factors that make people pay a lot more for an apartment, then I can make a predictive model which will accurately and precisely predict apartment prices based on the presence or absence of these factors.

Section 2:

When I first opened the data, there were many features contained in the set which were simply products of MTurk. MTurk is a marketplace where users are paid to collect the data needed to form an observation, a single row of our dataset. One can only hope that these minions did good job collecting the data. Each of our observations represents an apartment unit and all its characteristics or features. As mentioned above, the goal of our model is to take these features and predict the unit's *sale_price_$*. All the features which were generated by MTurk had no actual bearing on the unit, so they were the first to get dropped from the dataset. Features likes 'HITId', 'HITTypeId', 'Reward', 'CreationTime', 'AcceptTime', 'SubmitTime', and even 'URL ' where of no use to me. I further whittled the list of features to 26. This list contained information like the number of half bathrooms, the walk score of how close it is to necessary or awesome places, the sale price, the listing price if not yet sold, the square footage, the percentage of tax deductible, the number of rooms, full bathrooms, bedrooms, floors in the building, the kitchen type, model type, dining room type, whether it's a cooperative or condo, any parking charges, maintenance costs, common charges, and total property taxes, whether it has a parking garage, how is it heated, are cat and or dogs allowed, when was it built and, most importantly, where is it located. The next, and arguably biggest challenge we had to overcome was missing data. Many of the features where missing many if not most of the data points. Most significantly, of the 2230 observations in the dataset, only 528 of them had a *sale_price*, meaning that no matter how much cleaning I did, I was only going to get 528 observations to train my model on. Because of this, I had to make sure that these 528 observations were as clean and accurate as possible. This is all part of the challenge faced by data scientists all the time and it is known as "missingness".

So, how did I address this missingness, in the data? Well, the first thing I did was to record or "featurize" the missingness. This is to say that sometimes, the fact that a particular observation is missing is strongly correlated with a particular outcome. More generally, sometimes a low value will be even lower if it's a missing low value. In my case, apartments are marketed based on what they have, not based on what they don't have. As such, an apartment either had a half bathroom or the information was missing. It either had a parking garage, or the information was missing. The ideal way to handle missingness is to impute it or fill in the missing value with some type of predicted value. If one were to impute the missing half bath or garage as "none", since there are no recorded values besides "yes", I chose to impute them that way and not record the missingness. However, with all the various ancillary costs associated with the units like parking charges, maintenance costs, common charges, and property taxes, I decided to do two things. First, I recorded a separate missingness feature for each of those costs. Then I imputed all the missing values as zero and combined them into a single feature called "additional costs". This simplified the dataset without noticeable loss in quality, at least as far as I can tell. Next, I cleaned up all the typos in feature responses so that, for example, instead of having "yes" recorded in 6 different ways, I would have recorded in greater number in just 1 way. Further, I had trouble making predictive use of the full address of the apartment, so I created a feature of just the zip code. This trouble was part of the broader difficulty I had working with categorical features and continuous features in the same dataset. The biggest imputation I did by far was by using the MissForest method from the missingpy library. It attempts to predict the missing data points based on all the data that's not missing. I intentionally left as many features in the dataset as possible, even when they likely had little effect on *sale_price*, as the more data it had, the better MissForest would do. Perhaps because of my inexperience and or MissForest' s finicky-ness, I had to seriously work to encode the categorical features as integers yet prevent them from being imputed as continuous features. Suffice to say that I think I was successful, and I got a robust looking 528 full observations. I even figured out a way to decode them back to their categorical names, though this did not actually turn out to be useful because all tree modeling in python requires them to remain encoded.

Section 3:

So, I got my dataset and now I must train my model on it. First however, I created a train-test split, with k = 4 test set size. Let's briefly look at the features in the data set, which is really X_train with 396 observations.

Name: walk_score, A raw feature mean 83.100379 std 13.090814 min 15.000000 25% 76.000000 50% 85.000000 75% 94.000000 max 99.000000 This is the score of the area around the unit. A high mark means that many helpful, important, or awesome places are within walking distance	Name: sq_footage, A raw feature mean 889.206686 std 364.076320 min 375.000000 25% 716.347500 50% 808.510000 75% 975.500000 max 6215.000000	Name: additional_costs_$, A created feature mean 1288.856061 std 1326.775607 min 0.000000 25% 648.750000 50% 791.000000 75% 1224.000000 max 9980.000000 This is the combination of all the different cost found in the dataset
Name: approx_year_built, A raw feature mean 1962.28 std 20.47 min 1915.00 25% 1950.00 50% 1956.50 75% 1966.50 max 2016.00	Name: pct_tax_deductibl, mean 44.518258 std 5.002121 min 20.000000 25% 40.930000 50% 45.085000 75% 48.280000 max 65.000000	Name: Missing_parking_charges, A featurized featured Cat #occurences 1.0(yes) 393 0.0(no) 135

Name: Missing_maintenance_cost, A featurized feature Cat #occurences 0.0(no) 386 1.0(yes) 142	Name: num_total_rooms, A raw feature min = 1 max = 6	Name: Missing_taxes, A featurized feature Cat #occurences 1.0(yes) 397 0.0(no) 131
Name: num_full_bathrooms, A raw feature Cat #occurences 1 424 2 100 3 4	Name: num_floors_in_building, A raw feature min 0.000000 max 27.000000	Name: num_half_bathrooms A raw feature cat #occurences 0.0 498 1.0 29 2.0 1
Name: num_bedrooms, A raw feature. Cat #occurences 1.0 242 2.0 204 3.0 54 0.0 28	Name: kitchen_type, A raw feature cat #occurences eat in 211 combo 83 none 1	Name: Missing_common_charges, A featurized feature. Cat #occurences 1.0 396 0.0 132
Name: garage_exists, A raw feature. Cat #occurences 0.0(no) 434 1.0(yes) 94	Name: full_address_or_zip_code, A raw feature. Given that “location, location, location’ this could probably have been used better	Name: zip_code, A created feature min 11004 max 27110
Name: fuel_type, A raw feature Cat #occurences oil 190 electric 11 other 10 none 3	Name: dogs_allowed, A raw feature. Cat #occurences 0.0(no) 381 1.0(yes) 147	Name: cats_allowed, A raw feature. Cat #occurences 0.0(no) 285 1.0(yes) 243

Name: dining_room_type, A raw feature Cat #occurences formal 143 other 60 dining area 2	Name: date_of_sale, A raw feature.	Name: community_district_num, A raw feature. min = 3 max = 29
Name: coop_condo, A raw feature. Cat #occurences co-op 399 condo 129 Cooperative or condo		Looks good for 3:15am

I tried a few different models on this dataset before I decided on which one, I was going to ship. Before The first was a DecisionTreeRegressor model. Overall, I was surprised that this tree was not predicting so well. The most important feature was the number of full bathrooms. This could make sense as it's one of those details which is practically given in the name of an apartment listing. The second most important feature was cooperative or condo. Every model I tried felt that this was a very important feature. Next was square footage, followed by zip code, additional costs, community district, number of floors in the building, approximate year of construction and I didn't see any more unique ones as it made numerous further splits in the dataset. When I ran this model a bunch of time, I was getting variance in its preface metrics. I made a program to find the best depth for the tree based on the depth which maximizing R^2 and minimizing RMSE. In sample, R^2 = 0.929561 and RMSE = 48295.713543
I then fitted a tree diagram to the ideal hyperparameter tree and downloaded.

Somewhat let down by the Regression tree, the next model that I experimented with was a multivariate OLS model. It was surprisingly easy to set up and the statsmodel provided an excellent summary of each point. My in-sample OLS predictions were surprisingly poor with R^2 = 74.5128% and RMSE = $91867.96. I don't know why they were low. Nevertheless, oos, it did about the same as the tree model. As mentioned, the summary of OLS was very useful, it easily gives me the coefficients. The most important features were coop_condo, num_full_bathrooms, num_bedrooms, num_half_bathrooms, Missing_taxes , Missing_common_charges , dining_room_type , garage_exists, dogs_allowed, Missing_maintenance_cost and cats_allowed. These features all sound reasonable and oos, it seems to predict decently well so I would say that OLS is relatively useful.

The final model which I experimented with was a RandomForestReggressor model. Performance-wise, it did the best. I think this is because it has so much flexibility. In the same way that I was able to optimize the single tree, I was able to optimize the Random forest an order of magnitude more. I was able to find the ideal number of trees to average, the ideal depth of each tree and the ideal number of mtry features per-split which would drive the variance and hence our error down. I think that, based on the oos performance R^2 = 0.866864 and RMSE 62296.217624, this model is neither underfit nor overfit, rather it's just right (relatively). I think, due to the amount of optimization and hyper-parameters required, it is not a parametric model. The unfortunate tradeoff of all this optimization is that interpretability has gotten lost in the forest.

Section 4:

Performance Results for your Random Forest Model

DecisionTreeRegressor	In sample R^2: 0.929561 In sample RMSE 48295.713543	OOS R^2: 0.742588 OOS RMSE 86622.109789
OLS	In sample OLS R^2: 0.745389 In sample OLS RMSE: $91820.87	OOS R^2: 0.7827 OOS RMSE: $79587.27
RandomForestReggressor	In sample R^2: 0.974242 In sample RMSE 29272.587125	OOS R^2: 0.871798 OOS RMSE 60542.335003

Section 5:

Overall, I had a lot of fun doing this project. Just opening the data set and messing around with python would have been enough but I really cleaned it up well, having gotten MissForest to work for both categorical and continuous features. Unfortunately, I think this issue is probably where I'm losing performance. The fact that most of the features in the data set are nominal, I'm skeptical that all this regression can be good. Often, after doing many models on a dataset, I'd notice that some encoded categorical features became continues and it was just a stop and start game of whack-a-mole of refactorizing them until the end. I think a big aspect which I will be working on for the future is modeling these nominal variables in trees. In the meantime, I do not believe that my model is ready to ship

In [39]:

```
import pandas as pd
import sys
import sklearn.neighbors._base
sys.modules['sklearn.neighbors.base'] = sklearn.neighbors._base
import numpy as np
import re
import pandas as pd
from missingpy import MissForest

houseData = pd.read_csv(r'C:\Users\VAIO\Documents\houseData.csv')
#houseData
```

In [40]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2230 entries, 0 to 2229
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   num_half_bathrooms              172 non-null    float64
 1   total_taxes                     584 non-null    object 
 2   walk_score                      2230 non-null   int64  
 3   listing_price_to_nearest_1000  1696 non-null   object 
 4   sq_footage                      1020 non-null   float64
 5   sale_price                      528 non-null    object 
 6   pct_tax_deductibl               476 non-null    float64
 7   parking_charges                 559 non-null    object 
 8   num_total_rooms                 2228 non-null   float64
 9   num_full_bathrooms              2230 non-null   int64  
 10  num_floors_in_building          1580 non-null   float64
 11  num_bedrooms                    2115 non-null   float64
 12  model_type                      2151 non-null   object 
 13  maintenance_cost                1607 non-null   object 
 14  kitchen_type                    2214 non-null   object 
 15  garage_exists                   404 non-null    object 
 16  full_address_or_zip_code        2230 non-null   object 
 17  fuel_type                       2118 non-null   object 
 18  dogs_allowed                    2230 non-null   object 
 19  dining_room_type                1782 non-null   object 
 20  date_of_sale                    528 non-null    object 
 21  coop_condo                      2230 non-null   object 
 22  community_district_num          2211 non-null   float64
 23  common_charges                  546 non-null    object 
 24  cats_allowed                    2230 non-null   object 
 25  approx_year_built               2190 non-null   float64
dtypes: float64(8), int64(2), object(16)
memory usage: 453.1+ KB
```

In [41]:

```
# Before combining costs to get a more comprehensive feature, lets record missingness
for x in houseData.index:
    if pd.isnull(houseData.loc[x, 'total_taxes'] ):
        houseData.loc[x, 'Missing_taxes'] = 1
    else:
        houseData.loc[x, 'Missing_taxes'] = 0

    if pd.isnull(houseData.loc[x, 'maintenance_cost'] ):
        houseData.loc[x, 'Missing_maintenance_cost'] = 1
    else:
        houseData.loc[x, 'Missing_maintenance_cost'] = 0

    if pd.isnull(houseData.loc[x, 'common_charges'] ):
        houseData.loc[x, 'Missing_common_charges'] = 1
```

```python
    else:
        houseData.loc[x, 'Missing_common_charges'] = 0

    if pd.isnull(houseData.loc[x, 'parking_charges'] ):
        houseData.loc[x, 'Missing_parking_charges'] = 1
    else:
        houseData.loc[x, 'Missing_parking_charges'] = 0
```

In [42]:

```python
#get rid of dollar signs and commas
#combine maintenence cost, park charge, common charge, tot taxes
houseData['sale_price'] = houseData['sale_price'].replace({'\$': '', ',': ''}, regex=True)
houseData['sale_price_$'] = houseData['sale_price']

houseData[['total_taxes',
           'maintenance_cost',
           'common_charges',
           'parking_charges' ,
           'listing_price_to_nearest_1000'
          ]
         ] = houseData[['total_taxes',
                        'maintenance_cost',
                        'common_charges',
                        'parking_charges' ,
                        'listing_price_to_nearest_1000'
                       ]
                      ].replace({'\$': '', ',': ''}, regex=True)

houseData[['parking_charges',
           'total_taxes',
           'maintenance_cost',
           'common_charges',
           'num_half_bathrooms',
           'garage_exists'
          ]
         ] = houseData[['parking_charges',
                        'total_taxes',
                        'maintenance_cost',
                        'common_charges',
                        'num_half_bathrooms',
                        'garage_exists'
                       ]
                      ].fillna(0)

for x in houseData.index:
    houseData.loc[x,"additional_costs_$"] = (int(houseData.loc[x,"parking_charges"]) +
                                              int(houseData.loc[x,"total_taxes"]) +
                                              int(houseData.loc[x,'maintenance_cost']) +
                                              int(houseData.loc[x,'common_charges'])
                                             )
```

In [43]:

```python
#drop the features with the dollar signs that have now been removed and combined into one
houseData.drop(['total_taxes',
                'sale_price',
                "parking_charges",
                'maintenance_cost',
                'common_charges'
               ], inplace = True, axis = 1)
```

In [44]:

```python
#Clean up the categorical features
for x in houseData.index:
    # kitchen type has a ton of typos
```

```python
    if houseData.loc[x, "kitchen_type"] == "Eat In":
        houseData.loc[x, "kitchen_type"] = "eat in"
    if houseData.loc[x, "kitchen_type"] == "eatin":
        houseData.loc[x, "kitchen_type"] = "eat in"
    if houseData.loc[x, "kitchen_type"] == "Eat in":
        houseData.loc[x, "kitchen_type"] = "eat in"
    if houseData.loc[x, "kitchen_type"] == "Combo":
        houseData.loc[x, "kitchen_type"] = "combo"
    if houseData.loc[x, "kitchen_type"] == "1955":
        houseData.loc[x, "kitchen_type"] = "none"
    if houseData.loc[x, "kitchen_type"] == "efficiency kitchen":
        houseData.loc[x, "kitchen_type"] = "efficiency"
    if houseData.loc[x, "kitchen_type"] == "efficiency kitchene":
        houseData.loc[x, "kitchen_type"] = "efficiency"
    if houseData.loc[x, "kitchen_type"] == "efficiency ktchen":
        houseData.loc[x, "kitchen_type"] = "efficiency"
    if houseData.loc[x, "kitchen_type"] == "efficiemcy":
        houseData.loc[x, "kitchen_type"] = "efficiency"
    # fuel type needs 'Other' - 'other'
    if houseData.loc[x, "fuel_type"] == "Other":
        houseData.loc[x, "fuel_type"] = "other"
    #garage exists needs 'Yes' - 'yes'
    #'Underground' - 'yes'
    # 'UG' - 'yes'
    # '1' - 'yes'
    # 'eys' - 'yes'
    if houseData.loc[x, "garage_exists"] == "Yes":
        houseData.loc[x, "garage_exists"] = "1"
    if houseData.loc[x, "garage_exists"] == 'Underground':
        houseData.loc[x, "garage_exists"] = "1"
    if houseData.loc[x, "garage_exists"] == 'UG':
        houseData.loc[x, "garage_exists"] = "1"
    if houseData.loc[x, "garage_exists"] == 'eys':
        houseData.loc[x, "garage_exists"] = "1"
    if houseData.loc[x, "garage_exists"] == '1':
        houseData.loc[x, "garage_exists"] = "1"
    if houseData.loc[x, "garage_exists"] == 'yes':
        houseData.loc[x, "garage_exists"] = "1"
    #dogs needs 'yes89' -'yes
    if houseData.loc[x, "dogs_allowed"] == "yes89":
        houseData.loc[x, "dogs_allowed"] = "1"
    if houseData.loc[x, "dogs_allowed"] == "yes":
        houseData.loc[x, "dogs_allowed"] = "1"
    if houseData.loc[x, "dogs_allowed"] == "no":
        houseData.loc[x, "dogs_allowed"] = "0"
    #cats needs 'y' - 'yes'
    if houseData.loc[x, "cats_allowed"] == "y":
        houseData.loc[x, "cats_allowed"] = "1"
    if houseData.loc[x, "cats_allowed"] == "yes":
        houseData.loc[x, "cats_allowed"] = "1"
    if houseData.loc[x, "cats_allowed"] == "no":
        houseData.loc[x, "cats_allowed"] = "0"
```

In [45]:

```python
# can't figure out a great way to use full address
houseData['zip_code'] = houseData['full_address_or_zip_code'].str.extract(r'(\d{5}\-?\d{0,
```

In [46]:

```python
import sklearn.neighbors._base
from sklearn.preprocessing import LabelEncoder
import sys
sys.modules['sklearn.neighbors.base'] = sklearn.neighbors._base
from missingpy import MissForest

#how to impute categorical features with missingpy's MissForest https://stackoverflow.com/
```

```python
def label_encoding(df, columns):
    encoders = dict()
    for col_name in columns:
        series = df[col_name]
        label_encoder = LabelEncoder()
        df[col_name] = pd.Series(
            label_encoder.fit_transform(series[series.notnull()]),
            index=series[series.notnull()].index
        )
        encoders[col_name] = label_encoder
    return encoders

# adding to be imputed global category along with features
features = ['num_half_bathrooms',
            'num_total_rooms',
            'num_full_bathrooms',
            'num_floors_in_building',
            'num_bedrooms',
            'kitchen_type',
            'full_address_or_zip_code',
            'fuel_type',
            'dining_room_type',
            'date_of_sale',
            'coop_condo',
            'community_district_num'
           ]
# label encoding features
encoders = label_encoding(houseData, features)
# categorical imputation using random forest
# parameters can be tuned accordingly
imp_cat = MissForest(criterion = "gini")
houseData[features] = imp_cat.fit_transform(houseData[features], cat_vars=[0,1,2,3,4,5,6,7
```

```
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
```

```
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Iteration: 0
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Iteration: 1
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
```

```
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Iteration: 2

C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
```

```
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Iteration: 3
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Iteration: 4
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
```

```
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Iteration: 5
```

In [56]:

```python
pd.set_option('max_columns', None)
pd.set_option("max_rows", None)
houseData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2230 entries, 0 to 2229
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   num_half_bathrooms              2230 non-null   float64
 1   walk_score                      2230 non-null   int64
 2   listing_price_to_nearest_1000   1696 non-null   object
 3   sq_footage                      1020 non-null   float64
 4   pct_tax_deductibl               476 non-null    float64
 5   num_total_rooms                 2230 non-null   float64
 6   num_full_bathrooms              2230 non-null   float64
 7   num_floors_in_building          2230 non-null   float64
 8   num_bedrooms                    2230 non-null   float64
 9   model_type                      2151 non-null   object
 10  kitchen_type                    2230 non-null   float64
 11  garage_exists                   2230 non-null   object
 12  full_address_or_zip_code        2230 non-null   float64
 13  fuel_type                       2230 non-null   float64
 14  dogs_allowed                    2230 non-null   object
 15  dining_room_type                2230 non-null   float64
 16  date_of_sale                    2230 non-null   float64
 17  coop_condo                      2230 non-null   float64
 18  community_district_num          2230 non-null   float64
 19  cats_allowed                    2230 non-null   object
 20  approx_year_built               2190 non-null   float64
 21  Missing_taxes                   2230 non-null   float64
 22  Missing_maintenance_cost        2230 non-null   float64
 23  Missing_common_charges          2230 non-null   float64
 24  Missing_parking_charges         2230 non-null   float64
 25  sale_price_$                    528 non-null    object
 26  additional_costs_$              2230 non-null   float64
 27  zip_code                        2215 non-null   object
dtypes: float64(20), int64(1), object(7)
memory usage: 487.9+ KB
```

In [48]:

```python
# Make an instance and perform the imputation
imp_cont = MissForest()
```

```python
X = houseData.drop(['sale_price_$', 'model_type'], axis = 1)
#X.info()
```

In [49]:
```python
#cat_vars=[0,5,6,7,8,10,11,13,14,15,16]
#[['listing_price_to_nearest_1000', 'sq_footage', 'pct_tax_deductibl', 'zip_code']]
X_imputed = imp_cont.fit_transform(X)
X_imputed
```

```
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
Iteration: 0
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
```

```
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
Iteration: 1
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
```

```
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
Iteration: 2
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion
='squared_error'` which is equivalent.
  warn(
C:\Users\VAIO\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the p
ast behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also th
e default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(
Iteration: 3
```

Out[49]:

```
array([[0.0000e+00, 8.2000e+01, 2.7435e+02, ..., 1.0000e+00, 7.6700e+02,
```

```
        1.1355e+04],
       [0.0000e+00, 8.9000e+01, 2.6641e+02, ..., 1.0000e+00, 6.0400e+02,
        1.1354e+04],
       [0.0000e+00, 9.0000e+01, 4.6933e+02, ..., 1.0000e+00, 5.6670e+03,
        1.1368e+04],
       ...,
       [0.0000e+00, 9.6000e+01, 8.5000e+02, ..., 1.0000e+00, 5.0000e+02,
        1.1385e+04],
       [0.0000e+00, 9.6000e+01, 8.5000e+02, ..., 1.0000e+00, 5.0000e+02,
        1.1385e+04],
       [0.0000e+00, 8.2000e+01, 8.9900e+02, ..., 1.0000e+00, 4.5770e+03,
        1.1360e+04]])
```

In [50]:

```
X_imputed = pd.DataFrame(X_imputed, columns = X.columns)
X_imputed = pd.concat([X_imputed,houseData['sale_price_$']], axis = 1)
# decoding features
#for variable in features:
#    X_imputed[variable] = encoders[variable].inverse_transform(X_imputed[variable].astype
#X_imputed
```

In [51]:

```
X_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2230 entries, 0 to 2229
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   num_half_bathrooms              2230 non-null   float64
 1   walk_score                      2230 non-null   float64
 2   listing_price_to_nearest_1000   2230 non-null   float64
 3   sq_footage                      2230 non-null   float64
 4   pct_tax_deductibl               2230 non-null   float64
 5   num_total_rooms                 2230 non-null   float64
 6   num_full_bathrooms              2230 non-null   float64
 7   num_floors_in_building          2230 non-null   float64
 8   num_bedrooms                    2230 non-null   float64
 9   kitchen_type                    2230 non-null   float64
 10  garage_exists                   2230 non-null   float64
 11  full_address_or_zip_code        2230 non-null   float64
 12  fuel_type                       2230 non-null   float64
 13  dogs_allowed                    2230 non-null   float64
 14  dining_room_type                2230 non-null   float64
 15  date_of_sale                    2230 non-null   float64
 16  coop_condo                      2230 non-null   float64
 17  community_district_num         2230 non-null   float64
 18  cats_allowed                    2230 non-null   float64
 19  approx_year_built               2230 non-null   float64
 20  Missing_taxes                   2230 non-null   float64
 21  Missing_maintenance_cost        2230 non-null   float64
 22  Missing_common_charges          2230 non-null   float64
 23  Missing_parking_charges         2230 non-null   float64
 24  additional_costs_$              2230 non-null   float64
 25  zip_code                        2230 non-null   float64
 26  sale_price_$                    528 non-null    object
dtypes: float64(26), object(1)
memory usage: 470.5+ KB
```

In [52]:

```
# We can't impute sale price so we're done imputing. We must drop any missing sale_price c
# to attempt to train models on this data

houseData5 = X_imputed
houseData5 = houseData5.dropna(axis = 0)
```

In [ ]:

```
houseData5
```

In [55]:

```
houseData5.to_csv(r'C:\Users\VAIO\Documents\houseData5.csv', index = False)
```

In [ ]:

```

```

In [556…

```
# importing dependencies
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_regression
from sklearn import tree
from sklearn.metrics import mean_squared_error
```

In [557…

```
houseData5 = pd.read_csv(r'C:\Users\VAIO\Documents\houseData5.csv')
pd.set_option('max_columns', None)
pd.set_option("max_rows", None)
```

In [563…

```
#prepare the data
X = houseData5.loc[:, ~houseData5.columns.isin(['sale_price_$',
                                                  'listing_price_to_nearest_1000'
                                                 ]
                                                )
                    ]
y = houseData5['sale_price_$']
```

In [579…

```
# train test split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# reset the indices
X_train,y_train,X_test,y_test = X_train.reset_index(drop=True),y_train.reset_index(drop=Tr
```

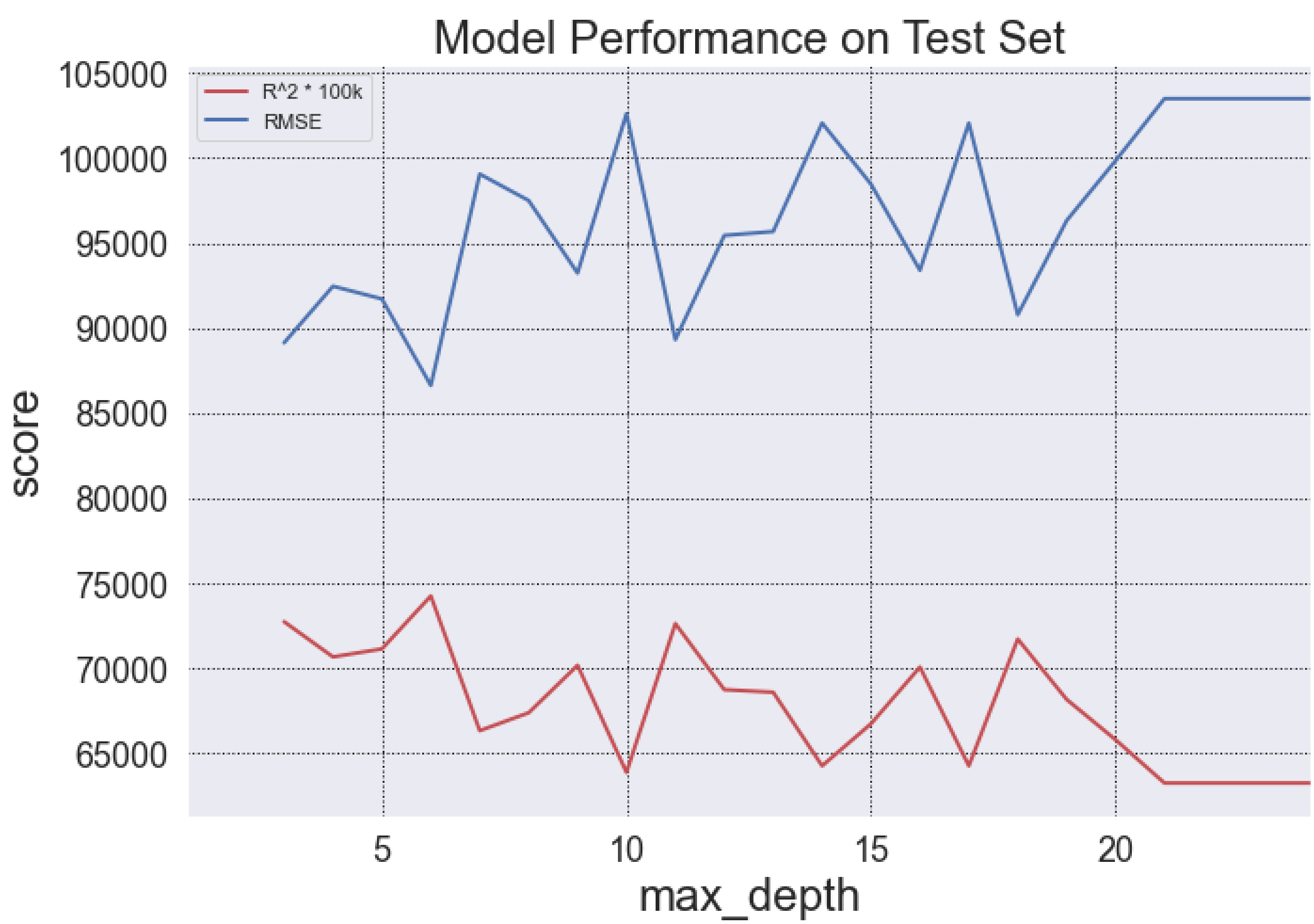
In [ ]:

```
#https://towardsdatascience.com/understanding-train-test-split-scikit-learn-python-ea676d5
from sklearn.tree import DecisionTreeRegressor

# Lets use a regression tree because our sale_price us continuous.
#We want to find the ideal depth of our tree so let's graph depth vs oos score

max_depth_range = list(range(3, 25))
# List to store the average R^2 and RMSE for each value of max_depth:
r2_list = []
RMSE_list = []
for depth in max_depth_range:
    reg = DecisionTreeRegressor(criterion = ('squared_error'),
                                max_depth = depth,
                                random_state = 0
                               )
    reg.fit(X_train, y_train)

    R2k = (reg.score(X_test, y_test))*100000
    RMSE = round(mean_squared_error(y_true=y_test, y_pred = reg.predict(X_test), squared=F
    r2_list.append(R2k)
    RMSE_list.append(RMSE)
```

In [296…

```
fig, ax = plt.subplots(nrows = 1, ncols = 1,
                       figsize = (10,7),
                       facecolor = 'white');
ax.plot(max_depth_range,
        r2_list,
```

```
        lw=2,
        color='r',
        label = "R^2 * 100k")
ax.plot(max_depth_range,
        RMSE_list,
        lw=2,
        color='b',
        label = "RMSE")
ax.legend()
ax.set_xlim([1, max(max_depth_range)])
ax.grid(True,
        axis = 'both',
        zorder = 0,
        linestyle = ':',
        color = 'k')
ax.tick_params(labelsize = 18)
ax.set_xlabel('max_depth', fontsize = 24)
ax.set_ylabel('score', fontsize = 24)
ax.set_title('Model Performance on Test Set', fontsize = 24)
fig.tight_layout()
```



In [297…

```
#It looks like maybe 6 deep is ideal
reg = DecisionTreeRegressor(criterion = ('squared_error'), max_depth = 6, random_state = (
reg.fit(X_train, y_train)
# Let's get those scores

# In sample predictions
yhat_in_sample = reg.predict(X_train)

# oos predictions
yhat_oos = reg.predict(X_test)

# IN SAMPLE
```

```python
print(f"In sample R^2: {round(reg.score(X_train, y_train), 6)}")
print(f"In sample RMSE {round(mean_squared_error(y_true=y_train, y_pred=yhat_in_sample, sq

# OOS
print(f"OOS R^2: {round(reg.score(X_test, y_test), 6)}")
print(f"OOS RMSE {round(mean_squared_error(y_true=y_test, y_pred=yhat_oos, squared=False),
```

```
In sample R^2: 0.929561
In sample RMSE 48295.713543
OOS R^2: 0.742588
OOS RMSE 86622.109789
```

In [485…

```python
text_representation = tree.export_text(reg, feature_names= ['num_half_bathrooms',
                                                              'walk_score',
                                                              'sq_footage',
                                                              'pct_tax_deductibl',
                                                              'num_total_rooms',
                                                              'num_full_bathrooms',
                                                              'num_floors_in_building',
                                                              'num_bedrooms',
                                                              'kitchen_type',
                                                              'garage_exists',
                                                              'full_address_or_zip_code',
                                                              'fuel_type','dogs_allowed',
                                                              'dining_room_type',
                                                              'date_of_sale',
                                                              'coop_condo',
                                                              'community_district_num',
                                                              'cats_allowed',
                                                              'approx_year_built',
                                                              'Missing_taxes',
                                                              'Missing_maintenance_cost',
                                                              'Missing_common_charges',
                                                              'Missing_parking_charges',
                                                              'additional_costs_$',
                                                              'zip_code'
                                                             ]
                                      )
print(text_representation)
```

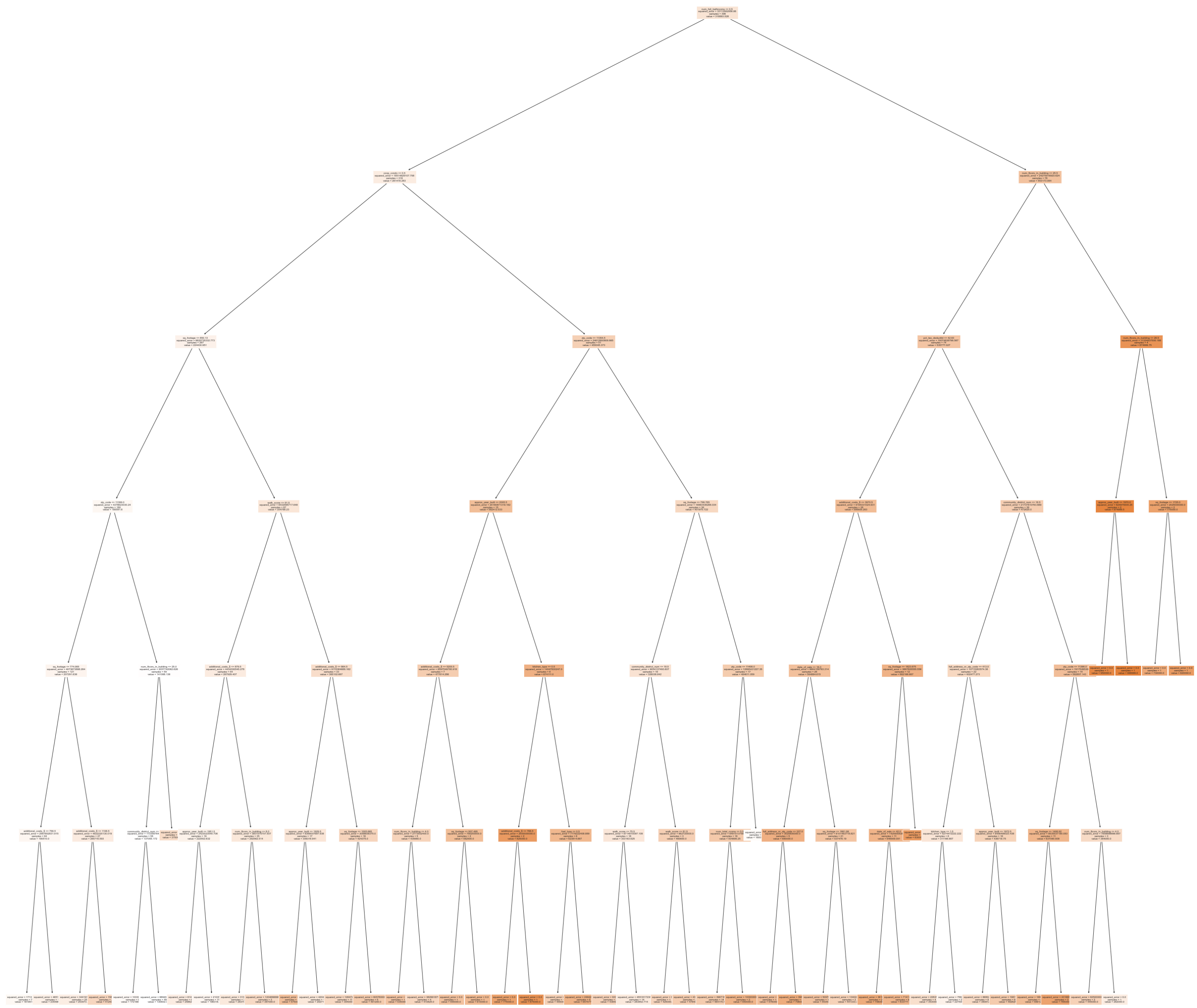
```
|--- num_full_bathrooms <= 0.50
|   |--- coop_condo <= 0.50
|   |   |--- sq_footage <= 850.13
|   |   |   |--- zip_code <= 11399.00
|   |   |   |   |--- sq_footage <= 774.07
|   |   |   |   |   |--- additional_costs_$ <= 756.50
|   |   |   |   |   |   |--- value: [181951.14]
|   |   |   |   |   |--- additional_costs_$ >  756.50
|   |   |   |   |   |   |--- value: [235586.67]
|   |   |   |   |--- sq_footage >  774.07
|   |   |   |   |   |--- additional_costs_$ <= 1106.50
|   |   |   |   |   |   |--- value: [253973.92]
|   |   |   |   |   |--- additional_costs_$ >  1106.50
|   |   |   |   |   |   |--- value: [412500.00]
|   |   |   |--- zip_code >  11399.00
|   |   |   |   |--- num_floors_in_building <= 25.00
|   |   |   |   |   |--- community_district_num <= 17.50
|   |   |   |   |   |   |--- value: [172188.89]
|   |   |   |   |   |--- community_district_num >  17.50
|   |   |   |   |   |   |--- value: [130602.04]
|   |   |   |   |--- num_floors_in_building >  25.00
|   |   |   |   |   |--- value: [375000.00]
|   |   |--- sq_footage >  850.13
|   |   |   |--- walk_score <= 91.50
|   |   |   |   |--- additional_costs_$ <= 879.50
```

```
|   |   |   |   |   |--- approx_year_built <= 1951.50
|   |   |   |   |   |   |--- value: [258625.00]
|   |   |   |   |   |--- approx_year_built >  1951.50
|   |   |   |   |   |   |--- value: [196318.18]
|   |   |   |   |--- additional_costs_$ >  879.50
|   |   |   |   |   |--- num_floors_in_building <= 8.00
|   |   |   |   |   |   |--- value: [283709.00]
|   |   |   |   |   |--- num_floors_in_building >  8.00
|   |   |   |   |   |   |--- value: [387000.00]
|   |   |   |--- walk_score >  91.50
|   |   |   |   |--- additional_costs_$ <= 864.50
|   |   |   |   |   |--- approx_year_built <= 1929.50
|   |   |   |   |   |   |--- value: [575000.00]
|   |   |   |   |   |--- approx_year_built >  1929.50
|   |   |   |   |   |   |--- value: [292711.75]
|   |   |   |   |--- additional_costs_$ >  864.50
|   |   |   |   |   |--- sq_footage <= 1005.67
|   |   |   |   |   |   |--- value: [394000.00]
|   |   |   |   |   |--- sq_footage >  1005.67
|   |   |   |   |   |   |--- value: [491200.00]
|   |--- coop_condo >  0.50
|   |   |--- zip_code <= 11354.50
|   |   |   |--- approx_year_built <= 2005.50
|   |   |   |   |--- additional_costs_$ <= 5200.50
|   |   |   |   |   |--- num_floors_in_building <= 4.00
|   |   |   |   |   |   |--- value: [535000.00]
|   |   |   |   |   |--- num_floors_in_building >  4.00
|   |   |   |   |   |   |--- value: [410825.00]
|   |   |   |   |--- additional_costs_$ >  5200.50
|   |   |   |   |   |--- sq_footage <= 907.49
|   |   |   |   |   |   |--- value: [555000.00]
|   |   |   |   |   |--- sq_footage >  907.49
|   |   |   |   |   |   |--- value: [610000.00]
|   |   |   |--- approx_year_built >  2005.50
|   |   |   |   |--- kitchen_type <= 0.50
|   |   |   |   |   |--- additional_costs_$ <= 766.50
|   |   |   |   |   |   |--- value: [775000.00]
|   |   |   |   |   |--- additional_costs_$ >  766.50
|   |   |   |   |   |   |--- value: [875000.00]
|   |   |   |   |--- kitchen_type >  0.50
|   |   |   |   |   |--- fuel_type <= 0.50
|   |   |   |   |   |   |--- value: [455000.00]
|   |   |   |   |   |--- fuel_type >  0.50
|   |   |   |   |   |   |--- value: [657577.60]
|   |   |--- zip_code >  11354.50
|   |   |   |--- sq_footage <= 799.77
|   |   |   |   |--- community_district_num <= 18.50
|   |   |   |   |   |--- walk_score <= 75.50
|   |   |   |   |   |   |--- value: [438500.00]
|   |   |   |   |   |--- walk_score >  75.50
|   |   |   |   |   |   |--- value: [283852.71]
|   |   |   |   |--- community_district_num >  18.50
|   |   |   |   |   |--- walk_score <= 81.50
|   |   |   |   |   |   |--- value: [425000.00]
|   |   |   |   |   |--- walk_score >  81.50
|   |   |   |   |   |   |--- value: [478400.00]
|   |   |   |--- sq_footage >  799.77
|   |   |   |   |--- zip_code <= 11406.00
|   |   |   |   |   |--- num_total_rooms <= 5.50
|   |   |   |   |   |   |--- value: [484170.57]
|   |   |   |   |   |--- num_total_rooms >  5.50
|   |   |   |   |   |   |--- value: [650000.00]
|   |   |   |   |--- zip_code >  11406.00
|   |   |   |   |   |--- value: [160000.00]
|--- num_full_bathrooms >  0.50
|   |--- num_floors_in_building <= 25.50
```

```
|   |   |--- pct_tax_deductibl <= 42.83
|   |   |   |--- additional_costs_$ <= 3970.50
|   |   |   |   |--- date_of_sale <= 16.00
|   |   |   |   |   |--- full_address_or_zip_code <= 327.00
|   |   |   |   |   |   |--- value: [576000.00]
|   |   |   |   |   |--- full_address_or_zip_code >  327.00
|   |   |   |   |   |   |--- value: [706000.00]
|   |   |   |   |--- date_of_sale >  16.00
|   |   |   |   |   |--- sq_footage <= 1661.86
|   |   |   |   |   |   |--- value: [553500.00]
|   |   |   |   |   |--- sq_footage >  1661.86
|   |   |   |   |   |   |--- value: [441333.33]
|   |   |   |--- additional_costs_$ >  3970.50
|   |   |   |   |--- sq_footage <= 1823.67
|   |   |   |   |   |--- date_of_sale <= 42.00
|   |   |   |   |   |   |--- value: [706000.00]
|   |   |   |   |   |--- date_of_sale >  42.00
|   |   |   |   |   |   |--- value: [633777.78]
|   |   |   |   |--- sq_footage >  1823.67
|   |   |   |   |   |--- value: [830000.00]
|   |   |--- pct_tax_deductibl >  42.83
|   |   |   |--- community_district_num <= 18.50
|   |   |   |   |--- full_address_or_zip_code <= 413.00
|   |   |   |   |   |--- kitchen_type <= 1.50
|   |   |   |   |   |   |--- value: [353000.00]
|   |   |   |   |   |--- kitchen_type >  1.50
|   |   |   |   |   |   |--- value: [227500.00]
|   |   |   |   |--- full_address_or_zip_code >  413.00
|   |   |   |   |   |--- approx_year_built <= 1972.50
|   |   |   |   |   |   |--- value: [380812.50]
|   |   |   |   |   |--- approx_year_built >  1972.50
|   |   |   |   |   |   |--- value: [492625.00]
|   |   |   |--- community_district_num >  18.50
|   |   |   |   |--- zip_code <= 11396.00
|   |   |   |   |   |--- sq_footage <= 1406.92
|   |   |   |   |   |   |--- value: [589125.00]
|   |   |   |   |   |--- sq_footage >  1406.92
|   |   |   |   |   |   |--- value: [765000.00]
|   |   |   |   |--- zip_code >  11396.00
|   |   |   |   |   |--- num_floors_in_building <= 4.00
|   |   |   |   |   |   |--- value: [445000.00]
|   |   |   |   |   |--- num_floors_in_building >  4.00
|   |   |   |   |   |   |--- value: [262000.00]
|   |--- num_floors_in_building >  25.50
|   |   |--- num_floors_in_building <= 26.50
|   |   |   |--- approx_year_built <= 1970.00
|   |   |   |   |--- value: [950000.00]
|   |   |   |--- approx_year_built >  1970.00
|   |   |   |   |--- value: [999999.00]
|   |   |--- num_floors_in_building >  26.50
|   |   |   |--- sq_footage <= 1725.00
|   |   |   |   |--- value: [730000.00]
|   |   |   |--- sq_footage >  1725.00
|   |   |   |   |--- value: [820000.00]
```

In [489…

```
fig = plt.figure(figsize=(50,50))
tree.plot_tree(reg, feature_names = X_train.columns, class_names=y, filled=True, fontsize=
plt.show()
fig.savefig(fname = 'decision_tree.png', dpi= 200)
```



In [ ]:

```
# from sklearn.tree import DecisionTreeClassifier
# classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
# classifier.fit(X_train, y_train)
# #Predicting the test set result
# y_pred= classifier.predict(X_test)
# classifier.score()
```

In [353…

```
#lets try some different OLS regression methods
import statsmodels.api as sm

OLSmodel1 = sm.OLS(y_train, X_train).fit()
OLSmodel1.summary()
```

Out[353…

OLS Regression Results

Dep. Variable:	sale_price_$	**R-squared (uncentered):**	0.937
Model:	OLS	**Adj. R-squared (uncentered):**	0.933
Method:	Least Squares	**F-statistic:**	222.2

Date:	Tue, 24 May 2022	**Prob (F-statistic):**	3.71e-206
Time:	14:01:54	**Log-Likelihood:**	-5087.4
No. Observations:	396	**AIC:**	1.022e+04
Df Residuals:	371	**BIC:**	1.032e+04
Df Model:	25		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
num_half_bathrooms	5.002e+04	1.96e+04	2.553	0.011	1.15e+04	8.85e+04
walk_score	981.4685	405.594	2.420	0.016	183.917	1779.020
sq_footage	26.8503	16.698	1.608	0.109	-5.984	59.684
pct_tax_deductibl	-3791.0074	1008.688	-3.758	0.000	-5774.469	-1807.545
num_total_rooms	-2161.5332	7778.794	-0.278	0.781	-1.75e+04	1.31e+04
num_full_bathrooms	1.133e+05	1.67e+04	6.791	0.000	8.05e+04	1.46e+05
num_floors_in_building	5993.2612	1040.144	5.762	0.000	3947.944	8038.578
num_bedrooms	6.583e+04	1.15e+04	5.722	0.000	4.32e+04	8.85e+04
kitchen_type	-1.921e+04	7005.627	-2.742	0.006	-3.3e+04	-5434.655
garage_exists	1.488e+04	1.4e+04	1.064	0.288	-1.26e+04	4.24e+04
full_address_or_zip_code	-25.3290	14.708	-1.722	0.086	-54.251	3.593
fuel_type	2780.5788	5285.483	0.526	0.599	-7612.683	1.32e+04
dogs_allowed	1.367e+04	1.5e+04	0.909	0.364	-1.59e+04	4.33e+04
dining_room_type	1.49e+04	3758.294	3.965	0.000	7511.226	2.23e+04
date_of_sale	-97.8663	76.771	-1.275	0.203	-248.827	53.094
coop_condo	2.256e+05	4.75e+04	4.747	0.000	1.32e+05	3.19e+05
community_district_num	5726.5213	1987.571	2.881	0.004	1818.203	9634.840
cats_allowed	8186.4007	1.39e+04	0.588	0.557	-1.92e+04	3.55e+04
approx_year_built	-33.4209	51.143	-0.653	0.514	-133.988	67.146
Missing_taxes	4.178e+04	4.21e+04	0.992	0.322	-4.11e+04	1.25e+05
Missing_maintenance_cost	8497.6366	2.84e+04	0.299	0.765	-4.74e+04	6.44e+04
Missing_common_charges	3.758e+04	2.88e+04	1.304	0.193	-1.91e+04	9.42e+04
Missing_parking_charges	6778.0234	1.24e+04	0.546	0.585	-1.76e+04	3.12e+04
additional_costs_$	10.6314	5.335	1.993	0.047	0.140	21.123
zip_code	6.6169	3.435	1.926	0.055	-0.138	13.372

Omnibus:	52.812	**Durbin-Watson:**	2.112
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	166.790
Skew:	0.582	**Prob(JB):**	6.05e-37
Kurtosis:	5.959	**Cond. No.**	1.50e+05

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 1.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [492…

```
from sklearn.metrics import mean_squared_error, r2_score

# Let's get those scores

# In sample predictions
yhat_in_sampleOLS1 = OLSmodel1.predict(X_train)

# oos predictions
yhat_oosOLS1 = OLSmodel1.predict(X_test)

# IN SAMPLE
print(f"In sample OLS R^2: {round(r2_score(y_true = y_train, y_pred= yhat_in_sampleOLS1),
print(f"In sample OLS RMSE: ${round(mean_squared_error(y_true=y_train, y_pred=yhat_in_samp

# OOS
print(f"OOS R^2: {round(r2_score(y_true = y_test, y_pred= yhat_oosOLS1), 6)}")
print(f"OOS RMSE: ${round(mean_squared_error(y_true=y_test, y_pred=yhat_oosOLS1, squared=F


print((OLSmodel1.params).sort_values(ascending=False))
#print((OLSmodel1.params).idxmax)
```

```
In sample OLS R^2: 0.745128
In sample OLS RMSE: $91867.96
OOS R^2: 0.782819
OOS RMSE: $79565.6
coop_condo                  225599.210952
num_full_bathrooms          113313.881925
num_bedrooms                 65833.595700
num_half_bathrooms           50018.385203
Missing_taxes                41784.095494
Missing_common_charges       37575.668254
dining_room_type             14901.456664
garage_exists                14877.300830
dogs_allowed                 13674.503448
Missing_maintenance_cost      8497.636642
cats_allowed                  8186.400710
Missing_parking_charges       6778.023408
num_floors_in_building        5993.261242
community_district_num        5726.521305
fuel_type                     2780.578849
walk_score                     981.468516
sq_footage                      26.850339
additional_costs_$              10.631353
zip_code                         6.616863
full_address_or_zip_code       -25.329048
approx_year_built              -33.420926
date_of_sale                   -97.866288
num_total_rooms              -2161.533248
pct_tax_deductibl            -3791.007389
kitchen_type                -19210.370598
dtype: float64
```

In [348…

```
# lets try another OLS method
```

```python
from sklearn import linear_model

lm = linear_model.LinearRegression()
OLSmodel2 = lm.fit(X_train,y_train)
```

In [349…

```python
from sklearn.metrics import mean_squared_error, r2_score

# Let's get those scores

# In sample predictions
yhat_in_sampleOLS2 = OLSmodel2.predict(X_train)

# oos predictions
yhat_oosOLS2 = OLSmodel2.predict(X_test)

# IN SAMPLE
print(f"In sample OLS R^2: {round(r2_score(y_true = y_train, y_pred= yhat_in_sampleOLS2),
print(f"In sample OLS RMSE: ${round(mean_squared_error(y_true=y_train, y_pred=yhat_in_samp

# OOS
print(f"OOS R^2: {round(r2_score(y_true = y_test, y_pred= yhat_oosOLS2), 6)}")
print(f"OOS RMSE: ${round(mean_squared_error(y_true=y_test, y_pred=yhat_oosOLS2, squared=F
```

```
In sample OLS R^2: 0.745389
In sample OLS RMSE: $91820.87
OOS R^2: 0.7827
OOS RMSE: $79587.27
```

In [350…

```python
lm.coef_
```

Out[350…

```
array([ 4.86051122e+04,  1.00726564e+03,  2.75787158e+01, -3.78572947e+03,
       -1.70758341e+03,  1.11485130e+05,  5.80580645e+03,  6.52273220e+04,
       -1.94234885e+04,  1.48594628e+04, -2.51976823e+01,  3.02019476e+03,
        1.43245381e+04,  1.50175636e+04, -9.54119867e+01,  2.16749192e+05,
        5.96381868e+03,  8.07901925e+03,  2.25824485e+02,  4.52901474e+04,
        1.01793020e+04,  3.81465194e+04,  7.88065883e+03,  1.21330914e+01,
        6.78311305e+00])
```

In [351…

```python
lm.intercept_
```

Out[351…

```
-522063.31937626924
```

In [581…

```python
# All good but we definitely saw some variance. Lets try random forest
#https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

r2_listRF = []
RMSE_listRF = []
position_list = []
max_depth_range = list(range(3, 20))
max_features_range = list(range(5,23))
for depth in max_depth_range:
    for features in max_features_range:
        rf = RandomForestRegressor(criterion='squared_error',
                                   oob_score= True,
                                   n_estimators = 50,
                                   max_features= features,
                                   max_depth= depth,
                                   random_state = 0
                                  )
        rf.fit(X_train, y_train)
```

```python
        r2RF= (rf.score(X_test, y_test))
        RMSERF= round(mean_squared_error(y_true=y_test, y_pred = rf.predict(X_test), squar
        position = str(depth) + ' x ' + str(features)
        r2_listRF.append(r2RF)
        RMSE_listRF.append(RMSERF)
        position_list.append(position)
# print('indices             ',list(range(1,len(max_depth_range))))
# print('max_depth_range:    ', max_depth_range, '\n', 'max_features_range:', max_features
# print('r2: ', max(r2_listRF), r2_listRF.index(max(r2_listRF) ) )
print('max_depth_range x max_features_range')
print('r2: ',
      round(max(r2_listRF),4),
      position_list[ int(r2_listRF.index(max(r2_listRF)))]

     )
print('RMSE:',
      min(RMSE_listRF),
      position_list[ int(RMSE_listRF.index(min(RMSE_listRF)))]

     )
```

```
max_depth_range x max_features_range
r2:  0.8843 7 x 11
RMSE: 57524.32 7 x 11
```

In [ ]:

```python
# 500trees
# max_depth_range x max_features_range
# r2:  0.8603 17d x 10f
# RMSE: 63811.64 17d x 10f

# 100t
# max_depth_range x max_features_range
# r2:  0.8613 17d x 14f
# RMSE: 63574.74 17d x 14f

# 50t
# max_depth_range x max_features_range
# r2:  0.8669 17d x 14f
# RMSE: 62296.22 17d x 14f
```

In [580…

```python
# so we know that depth of 17 and 14 feature try are ideal.


rf = RandomForestRegressor(criterion='squared_error',
                           n_estimators = 50,
                           oob_score=True,
                           max_features= 14,
                           max_depth= 17,
                           random_state = 0
                          )
rf.fit(X_train, y_train)

# In sample predictions
yhat_in_sample = rf.predict(X_train)

# oos predictions
yhat_oos = rf.predict(X_test)

# IN SAMPLE
print(f"In sample R^2: {round(rf.score(X_train, y_train), 6)}")
print(f"In sample RMSE {round(mean_squared_error(y_true=y_train, y_pred=yhat_in_sample, sc

# OOS
```

```python
print(f"OOS R^2: {round(rf.score(X_test, y_test), 6)}")
print(f"OOS RMSE {round(mean_squared_error(y_true=y_test, y_pred=yhat_oos, squared=False),
```

```
In sample R^2: 0.974242
In sample RMSE 29272.587125
OOS R^2: 0.871798
OOS RMSE 60542.335003
```