In [1]:
```
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

```

Load the file into a dataframe

In [2]:
```
FILE_NAME = 'housing.csv'
datHousing = pd.read_csv(FILE_NAME)

```

Inspect the top rows

In [3]:
```
datHousing.head()

```

Out[3]:

	CRIM	ZN	INDUS	RIVER	NOX	RM	AGE	DIS	RAD	TAX	PRATIO	LSTAT	MEDV
0	3.32105	0.0	19.58	Yes	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	13.4
1	1.12658	0.0	19.58	Yes	0.871	5.012	88.0	1.6102	5	403	14.7	12.12	15.3
2	1.41385	0.0	19.58	Yes	0.871	6.129	96.0	1.7494	5	403	14.7	15.12	17.0
3	3.53501	0.0	19.58	Yes	0.871	6.152	82.6	1.7455	5	403	14.7	15.02	15.6
4	1.27346	0.0	19.58	Yes	0.605	6.250	92.6	1.7984	5	403	14.7	5.50	27.0

In [4]:

```python
# Correctly get the unique values
r = datHousing['RIVER'].unique()
n = datHousing['NOX'].unique()
ra = datHousing['RAD'].unique()
t = datHousing['TAX'].unique()
p = datHousing['PRATIO'].unique()
z = datHousing['ZN'].unique()
c = datHousing['CRIM'].unique()

print(f"Unique values in RIVER: {r}")
print(f"Unique values in NOX: {n}")
print(f"Unique values in RAD: {ra}")
print(f"Unique values in TAX: {t}")
print(f"Unique values in PRATIO: {p}")
print(f"Unique values in ZN: {z}")
print(f"Unique values in CRIM: {c}")
```

```
Unique values in RIVER: ['Yes' 'No']
Unique values in NOX: [0.871  0.605  0.489  0.55   0.507  0.464  0.447  0.4
429 0.401  0.77
 0.718  0.631  0.668  0.538  0.469  0.458  0.524  0.499  0.428  0.448
 0.439  0.41   0.403  0.411  0.453  0.4161 0.398  0.409  0.413  0.437
 0.426  0.449  0.445  0.52   0.547  0.581  0.624  0.51   0.488  0.422
 0.404  0.415  0.504  0.431  0.392  0.394  0.647  0.575  0.4    0.389
 0.385  0.405  0.433  0.472  0.544  0.493  0.46   0.4379 0.515  0.442
 0.518  0.484  0.429  0.435  0.671  0.7    0.693  0.659  0.597  0.679
 0.614  0.584  0.713  0.74   0.655  0.58   0.532  0.583  0.609  0.585
 0.573 ]
Unique values in RAD: [ 5  4  8  3  1 24  2  6  7]
Unique values in TAX: [403 277 276 307 223 254 216 198 666 296 242 222 311
279 252 233 243 469
 226 313 256 284 337 345 305 398 281 247 270 384 432 188 437 193 265 255
 329 402 348 224 300 330 315 244 264 285 241 293 245 289 358 304 287 430
 422 370 352 351 280 335 411 187 334 711 391 273]
Unique values in PRATIO: [14.7 18.6 16.4 17.4 17.6 14.9 13.6 20.2 15.3 17.8
18.7 15.2 21.  19.2
```

Dummy code the categoric variables

In [5]:

```python
datHousing = pd.concat([datHousing, pd.get_dummies(datHousing['RIVER'], pr
datHousing.drop(['RIVER'], inplace=True, axis=1)


```

In [6]:
```
datHousing.head()
```

Out[6]:

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PRATIO	LSTAT	MEDV	RIVER_
0	3.32105	0.0	19.58	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	13.4	
1	1.12658	0.0	19.58	0.871	5.012	88.0	1.6102	5	403	14.7	12.12	15.3	
2	1.41385	0.0	19.58	0.871	6.129	96.0	1.7494	5	403	14.7	15.12	17.0	
3	3.53501	0.0	19.58	0.871	6.152	82.6	1.7455	5	403	14.7	15.02	15.6	
4	1.27346	0.0	19.58	0.605	6.250	92.6	1.7984	5	403	14.7	5.50	27.0	

3. For now, we are not going to break the data frame into training and testing partitions. Create a regression model on the entire data frame predicting MEDV using all of the other predictors. This is your baseline model. Report the R2 value and MSE for this model.

Create new dataset without target variable

In [7]:
```
# Create new dataset without the target variable which is in this case 'ME
datHousingSub = datHousing.drop(['MEDV'], axis=1)
```

In [8]:
```
datHousingSub.head()
```

Out[8]:

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PRATIO	LSTAT	RIVER_Yes
0	3.32105	0.0	19.58	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	1
1	1.12658	0.0	19.58	0.871	5.012	88.0	1.6102	5	403	14.7	12.12	1
2	1.41385	0.0	19.58	0.871	6.129	96.0	1.7494	5	403	14.7	15.12	1
3	3.53501	0.0	19.58	0.871	6.152	82.6	1.7455	5	403	14.7	15.02	1
4	1.27346	0.0	19.58	0.605	6.250	92.6	1.7984	5	403	14.7	5.50	1

Create new dataset with target variable

In [9]:
```python
target_name = list(['MEDV'])
housing_target = datHousing[target_name]

print(datHousingSub.head(10))
print(housing_target.head(10))
print (len(datHousingSub))
print (len(housing_target))
```

```
      CRIM   ZN  INDUS    NOX     RM    AGE     DIS  RAD  TAX  PRATIO  LSTAT  \
0  3.32105  0.0  19.58  0.871  5.403  100.0  1.3216    5  403    14.7  26.82   
1  1.12658  0.0  19.58  0.871  5.012   88.0  1.6102    5  403    14.7  12.12   
2  1.41385  0.0  19.58  0.871  6.129   96.0  1.7494    5  403    14.7  15.12   
3  3.53501  0.0  19.58  0.871  6.152   82.6  1.7455    5  403    14.7  15.02   
4  1.27346  0.0  19.58  0.605  6.250   92.6  1.7984    5  403    14.7   5.50   
5  1.83377  0.0  19.58  0.605  7.802   98.2  2.0407    5  403    14.7   1.92   
6  1.51902  0.0  19.58  0.605  8.375   93.9  2.1620    5  403    14.7   3.32   
7  0.13587  0.0  10.59  0.489  6.064   59.1  4.2392    4  277    18.6  14.66   
8  0.43571  0.0  10.59  0.489  5.344  100.0  3.8750    4  277    18.6  23.09   
9  0.17446  0.0  10.59  0.489  5.960   92.1  3.8771    4  277    18.6  17.27   

   RIVER_Yes  
0          1  
1          1  
2          1  
3          1  
4          1  
5          1  
6          1  
7          1  
8          1  
9          1  
   MEDV
0  13.4
1  15.3
2  17.0
3  15.6
4  27.0
5  50.0
6  50.0
7  24.4
8  20.0
9  21.7
506
506
```

In [10]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(datHousingSub, housing
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(354, 12)
(152, 12)
(354, 1)
(152, 1)
```

In [11]:
```
# import the LinearRegression class
from sklearn.linear_model import LinearRegression

regressor = LinearRegression(fit_intercept = True) # instantiate the Linea
regressor.fit(X_train, y_train) # train the model

print(f'r_sqr value: {regressor.score(X_train, y_train)}')

```

```
r_sqr value: 0.7231057672823237
```

In [12]:
```
y_train_pred = regressor.predict(X_train)
y_test_pred = regressor.predict(X_test)
```

In [13]:
```
#Report the R2 value and MSE for this model.
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from math import sqrt

print("Baseline Regression Model Results:")
print('RMSE train: %.3f, test: %.3f' % (
        sqrt(mean_squared_error(y_train, y_train_pred)),
        sqrt(mean_squared_error(y_test, y_test_pred))))
print('R^2 train: %.3f, test: %.3f' % (
        r2_score(y_train, y_train_pred),
        r2_score(y_test, y_test_pred)))
```

```
Baseline Regression Model Results:
RMSE train: 4.730, test: 4.843
R^2 train: 0.723, test: 0.746
```

In [ ]:
```

```

Fit 2 cluster model

4.Create a data frame called datHousingSub (ALREADY DONE) for clustering by generating a subset with all columns except MEDV. We want to exclude MEDV from the clustering since this is the value that we will try to predict later from the clusters. Do a summary on the subset to verify.

In [14]:
```
# Using scikit-learn to perform K-Means clustering

# Specify the number of clusters (2) and fit the data dat_rider_feats
kmeans = KMeans(n_clusters=2, random_state=42).fit(datHousingSub)

```

```
C:\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [15]:
```
# Set global NumPy print options for formatting
np.set_printoptions(suppress=True)

print(datHousingSub.columns)
centroids = (kmeans.cluster_centers_)
print(centroids)
```

```
Index(['CRIM', 'ZN', 'INDUS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PRATIO', 'LSTAT', 'RIVER_Yes'],
      dtype='object')
[[  0.38877444  15.58265583   8.42089431   0.51184743   6.38800542
   60.63224932   4.44127154   4.45528455 311.92682927  17.80921409
   10.41745257   0.07317073]
 [ 12.29916168   0.          18.45182482   0.67010219   6.00621168
   89.96788321   2.05447007  23.27007299 667.64233577  20.19635036
   18.67452555   0.05839416]]
```

5. Using the KMeans algorithm, create a model with 2 clusters on the datHousingSub data frame. Report the centroid values for each cluster and the sizes of each cluster. Using the characteristics that are especially divergent between the two clusters, what would you name these clusters?

To me this looks like 0 is group low crime, and higher Zonning number (less pop per acre), with Lower Industrial activity, and lower age, I would assess this to represent a Suburban center which tend to be newer areas, with lower crime rates than urban cities. We can see the opposite of this is true for group 1, as it is Older and have higher crime rate, 0 for zoning and industrial activity is rather high in comparison.

0 = Suburban and 1 = Urban

In [16]:
```python
labels = (kmeans.labels_)
print(labels)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [17]:
```python
# Calculate silhouette_score
from sklearn.metrics import silhouette_score

print(silhouette_score(datHousingSub, kmeans.labels_))
```

```
0.7717962604192585
```

6. Repeat step 5 with 3 clusters. Do you think that adding another cluster helps to partition the data? Why or why not?

In [19]:
```python
# Using scikit-learn to perform K-Means clustering

# Specify the number of clusters (2) and fit the data dat_rider_feats
k_means = KMeans(n_clusters=3, random_state=42).fit(datHousingSub)

```

```
C:\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [21]:
```python
print(datHousingSub.columns)
k_centroids = (k_means.cluster_centers_)
print(k_centroids)
```

```
Index(['CRIM', 'ZN', 'INDUS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PRATIO', 'LSTAT', 'RIVER_Yes'],
      dtype='object')
[[ 12.29916168   0.          18.45182482   0.67010219   6.00621168
   89.96788321   2.05447007  23.27007299 667.64233577  20.19635036
   18.67452555   0.05839416]
 [  0.2442057   17.37642586   6.70262357   0.48471369   6.4741635
   56.1661597    4.83579772   4.3269962  275.21292776  17.87338403
    9.55292776   0.07604563]
 [  0.74746858  11.13207547  12.68415094   0.57916981   6.17423585
   71.71320755   3.4624       4.77358491 403.01886792  17.65
   12.56245283   0.06603774]]
```

In [22]:
```
k_labels = (k_means.labels_)
print(k_labels)
```

```
[2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 2 2
 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1
 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 2 2 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 1 1 1 1 1]
```

In [23]:
```
# Calculate silhouette_score
from sklearn.metrics import silhouette_score

print(silhouette_score(datHousingSub, k_means.labels_))
```

```
0.6217529916045139
```

Do you think that adding another cluster helps to partition the data? Why or why not?

No not really, althought I would call the 3 groups in order 0 = Urban, 1 = Rural, and 2 = Suburban based on the variables mentioned before, where the groups seem to be split by zoning and industrial activities, which give indication in pop per acre and how many business are close by, this is also supported by the DIS variable for distance to work. However, based on the silhouette_score I would say the 2 cluster test is suffecient, this is purely based on the fact between what I classified as rurl and urban would be very small, and mostly likely closer to urban and outter urban area as the variable seem to be slightly close and would give better explanation power if they were combined or in a cluster of 2. The lower silhouette score indicates that the clusters are closer to overlapping than the previous score.

In [25]:
```
kmeans
```

Out[25]:
```
KMeans(n_clusters=2, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:
```
centroids
```

Out[26]:
```
array([[  0.38877444,  15.58265583,   8.42089431,   0.51184743,
          6.38800542,  60.63224932,   4.44127154,   4.45528455,
        311.92682927,  17.80921409,  10.41745257,   0.07317073],
       [ 12.29916168,   0.        ,  18.45182482,   0.67010219,
          6.00621168,  89.96788321,   2.05447007,  23.27007299,
        667.64233577,  20.19635036,  18.67452555,   0.05839416]])
```

7. We will use the 2-cluster model going forward. Merge the cluster ids from this 2-cluster model into the datHousing data frame using the column name Cluster to store this cluster id. Look at the first few rows of data to verify.

In [36]:
```python
# Merge the existing 'labels' variable directly into the DataFrame
datHousing['Cluster'] = labels

# Verify the first few rows
print(datHousing.head())
print(datHousing['Cluster'].unique())
```

```
      CRIM   ZN  INDUS    NOX     RM    AGE     DIS  RAD  TAX  PRATIO  LSTAT  \
0  3.32105  0.0  19.58  0.871  5.403  100.0  1.3216    5  403    14.7  26.82
1  1.12658  0.0  19.58  0.871  5.012   88.0  1.6102    5  403    14.7  12.12
2  1.41385  0.0  19.58  0.871  6.129   96.0  1.7494    5  403    14.7  15.12
3  3.53501  0.0  19.58  0.871  6.152   82.6  1.7455    5  403    14.7  15.02
4  1.27346  0.0  19.58  0.605  6.250   92.6  1.7984    5  403    14.7   5.50

   MEDV  RIVER_Yes  Cluster
0  13.4          1        0
1  15.3          1        0
2  17.0          1        0
3  15.6          1        0
4  27.0          1        0
[0 1]
```

8. Create a new data frame called datHousingC1 which contains all of the rows from datHousing in cluster 1. Look at the first few rows of data to verify. Check the Cluster column to make sure that it only stores the value of 1

In [51]:
```
datHousingC1 = datHousing.loc[datHousing['Cluster'] == 0]
# Verify the first few rows
print(datHousingC1.head())
print(datHousingC1['Cluster'].unique())
print (len(datHousingC1))
```

```
      CRIM   ZN  INDUS    NOX     RM    AGE     DIS  RAD  TAX  PRATIO  LSTAT  \
0  3.32105  0.0  19.58  0.871  5.403  100.0  1.3216    5  403    14.7  26.82
1  1.12658  0.0  19.58  0.871  5.012   88.0  1.6102    5  403    14.7  12.12
2  1.41385  0.0  19.58  0.871  6.129   96.0  1.7494    5  403    14.7  15.12
3  3.53501  0.0  19.58  0.871  6.152   82.6  1.7455    5  403    14.7  15.02
4  1.27346  0.0  19.58  0.605  6.250   92.6  1.7984    5  403    14.7   5.50

   MEDV  RIVER_Yes  Cluster
0  13.4          1        0
1  15.3          1        0
2  17.0          1        0
3  15.6          1        0
4  27.0          1        0
[0]
369
```

9. Create a new data frame called datHousingC2 which contains all of the rows from datHousing in cluster 2. Use summary to verify the results. Check the Cluster column to make sure that it only stores the value of 2

In [50]:
```
datHousingC2 = datHousing.loc[datHousing['Cluster'] == 1]
# Verify the first few rows
print(datHousingC2.head())
print(datHousingC2['Cluster'].unique())
print (len(datHousingC2))
```

```
       CRIM   ZN  INDUS    NOX     RM   AGE     DIS  RAD  TAX  PRATIO  LSTAT  \
27  8.98296  0.0   18.1  0.770  6.212  97.4  2.1222   24  666    20.2  17.60
28  3.84970  0.0   18.1  0.770  6.395  91.0  2.5052   24  666    20.2  13.27
29  5.20177  0.0   18.1  0.770  6.127  83.4  2.7227   24  666    20.2  11.48
30  4.22239  0.0   18.1  0.770  5.803  89.0  1.9047   24  666    20.2  14.64
31  3.47428  0.0   18.1  0.718  8.780  82.9  1.9047   24  666    20.2   5.29

    MEDV  RIVER_Yes  Cluster
27  17.8          1        1
28  21.7          1        1
29  22.7          1        1
30  16.8          1        1
31  21.9          1        1
[1]
137
```

10. Create a regression model predicting MEDV and the same predictors as the baseline model in step 3, using the data frame from cluster 1. What are the R2 value and MSE? Is this higher or lower than the baseline model?

In [52]:
```python
target_name = list(['MEDV'])
housing_target1 = datHousingC1[target_name]
datHousingSub1 = datHousingC1.drop(['MEDV', 'Cluster'], axis=1)

print(datHousingSub1.head(10))
print(housing_target1.head(10))
print (len(datHousingSub1))
print (len(housing_target1))
```

```
      CRIM   ZN  INDUS    NOX     RM    AGE     DIS  RAD  TAX  PRATIO  LSTAT  \
0  3.32105  0.0  19.58  0.871  5.403  100.0  1.3216    5  403    14.7  26.82
1  1.12658  0.0  19.58  0.871  5.012   88.0  1.6102    5  403    14.7  12.12
2  1.41385  0.0  19.58  0.871  6.129   96.0  1.7494    5  403    14.7  15.12
3  3.53501  0.0  19.58  0.871  6.152   82.6  1.7455    5  403    14.7  15.02
4  1.27346  0.0  19.58  0.605  6.250   92.6  1.7984    5  403    14.7   5.50
5  1.83377  0.0  19.58  0.605  7.802   98.2  2.0407    5  403    14.7   1.92
6  1.51902  0.0  19.58  0.605  8.375   93.9  2.1620    5  403    14.7   3.32
7  0.13587  0.0  10.59  0.489  6.064   59.1  4.2392    4  277    18.6  14.66
8  0.43571  0.0  10.59  0.489  5.344  100.0  3.8750    4  277    18.6  23.09
9  0.17446  0.0  10.59  0.489  5.960   92.1  3.8771    4  277    18.6  17.27

   RIVER_Yes
0          1
1          1
2          1
3          1
4          1
5          1
6          1
7          1
8          1
9          1
   MEDV
0  13.4
1  15.3
2  17.0
3  15.6
4  27.0
5  50.0
6  50.0
7  24.4
8  20.0
9  21.7
369
369
```

In [58]:
```
X_train, X_test, y_train, y_test = train_test_split(datHousingSub1, housin
print('shape of training data independent vars (x)')
print(X_train.shape)
print('shape of test data independent vars (x)')
print(X_test.shape)
print('shape of train data dependent vars (y)')
print(y_train.shape)
print('shape of test data dependent vars (y)')
print(y_test.shape)
regressor = LinearRegression(fit_intercept = True) # instantiate the Linea
regressor.fit(X_train, y_train) # train the model
print(f'r_sqr value: {regressor.score(X_train, y_train)}')
y_train_pred = regressor.predict(X_train)
y_test_pred = regressor.predict(X_test)
print("Cluster1 Regression Model Results:")
print('RMSE train: %.3f, test: %.3f' % (
        sqrt(mean_squared_error(y_train, y_train_pred)),
        sqrt(mean_squared_error(y_test, y_test_pred))))
print('R^2 train: %.3f, test: %.3f' % (
        r2_score(y_train, y_train_pred),
        r2_score(y_test, y_test_pred)))
```

```
shape of training data independent vars (x)
(258, 12)
shape of test data independent vars (x)
(111, 12)
shape of train data dependent vars (y)
(258, 1)
shape of test data dependent vars (y)
(111, 1)
r_sqr value: 0.8477216916255111
Cluster1 Regression Model Results:
RMSE train: 3.128, test: 3.121
R^2 train: 0.848, test: 0.877
```

Baseline Regression Model Results:

RMSE train: 4.730, test: 4.843

R^2 train: 0.723, test: 0.746

What are the R2 value and MSE? Is this higher or lower than the baseline model? These are higher than base line model as we can see this model performs better at accurately predicting the data relevant to cluster 1 which in this case, would be the data we had a larger amount of our "suburban area = 0" (maybe explaining the higher predictive power)

In [ ]:
```

```

11.Create a regression model predicting MEDV and the same predictors as the baseline model in step 3, using the data frame from cluster 2. What are the R2 value and MSE? Is this higher or lower than the baseline model?

In [60]:
```
target_name = list(['MEDV'])
housing_target2 = datHousingC2[target_name]
datHousingSub2 = datHousingC2.drop(['MEDV', 'Cluster'], axis=1)

print(datHousingSub2.head(10))
print(housing_target2.head(10))
print (len(datHousingSub2))
print (len(housing_target2))
```

```
        CRIM   ZN  INDUS    NOX     RM   AGE     DIS  RAD  TAX  PRATIO  LSTAT  \
27   8.98296  0.0   18.1  0.770  6.212  97.4  2.1222   24  666    20.2  17.60
28   3.84970  0.0   18.1  0.770  6.395  91.0  2.5052   24  666    20.2  13.27
29   5.20177  0.0   18.1  0.770  6.127  83.4  2.7227   24  666    20.2  11.48
30   4.22239  0.0   18.1  0.770  5.803  89.0  1.9047   24  666    20.2  14.64
31   3.47428  0.0   18.1  0.718  8.780  82.9  1.9047   24  666    20.2   5.29
32   5.66998  0.0   18.1  0.631  6.683  96.8  1.3567   24  666    20.2   3.73
33   6.53876  0.0   18.1  0.631  7.016  97.5  1.2024   24  666    20.2   2.96
34   8.26725  0.0   18.1  0.668  5.875  89.6  1.1296   24  666    20.2   8.88
364  4.26131  0.0   18.1  0.770  6.112  81.3  2.5091   24  666    20.2  12.67
365  4.54192  0.0   18.1  0.770  6.398  88.0  2.5182   24  666    20.2   7.79

     RIVER_Yes
27           1
28           1
29           1
30           1
31           1
32           1
33           1
34           1
364          0
365          0
     MEDV
27   17.8
28   21.7
29   22.7
30   16.8
31   21.9
32   50.0
33   50.0
34   50.0
364  22.6
365  25.0
137
137
```

In [61]:
```
X_train, X_test, y_train, y_test = train_test_split(datHousingSub2, housin
print('shape of training data independent vars (x)')
print(X_train.shape)
print('shape of test data independent vars (x)')
print(X_test.shape)
print('shape of train data dependent vars (y)')
print(y_train.shape)
print('shape of test data dependent vars (y)')
print(y_test.shape)
regressor = LinearRegression(fit_intercept = True) # instantiate the Linea
regressor.fit(X_train, y_train) # train the model
print(f'r_sqr value: {regressor.score(X_train, y_train)}')
y_train_pred = regressor.predict(X_train)
y_test_pred = regressor.predict(X_test)
print("Cluster1 Regression Model Results:")
print('RMSE train: %.3f, test: %.3f' % (
        sqrt(mean_squared_error(y_train, y_train_pred)),
        sqrt(mean_squared_error(y_test, y_test_pred))))
print('R^2 train: %.3f, test: %.3f' % (
        r2_score(y_train, y_train_pred),
        r2_score(y_test, y_test_pred)))
```

```
shape of training data independent vars (x)
(95, 12)
shape of test data independent vars (x)
(42, 12)
shape of train data dependent vars (y)
(95, 1)
shape of test data dependent vars (y)
(42, 1)
r_sqr value: 0.6735080000289209
Cluster1 Regression Model Results:
RMSE train: 4.709, test: 5.020
R^2 train: 0.674, test: 0.677
```

Baseline Regression Model Results:

RMSE train: 4.730, test: 4.843

R^2 train: 0.723, test: 0.746

What are the R2 value and MSE? Is this higher or lower than the baseline model? These are lower than base line model as we can see this model performs worse than the baseline at accurately predicting the data relevant to cluster 2 which in this case, would be the data we had a less amount of our "urban area = 1".

12. Summarize your findings. Do you think that clustering might improve your ability to predict the MEDV value? If so, under what contexts or constraints?

Clustering can improve the predictive power of a model as we can see in this example which may be due to the amount of data points vs 0 and 1 cluster giving better predictive power to the underlying factors that influence the target variable (MEDV) within different clusters. In this case:

For suburban areas, the clustering improved the ability to predict MEDV.

For urban areas, where the data maybe be more complex or less of, clustering did not improve the model’s predictive performance.

In [ ]:

```
1
```