SCOTT LEVY
MIS 776
ASSIGNMENT 5

CLUSTERING & MODEL IMPROVEMENT

1) Housing file opened and viewed

2) River dummy variable created and mean is 0.69:

In [11]:
```
datHousing = pd.concat([datHousing, pd.get_dummies(datHousing['RIVER'], prefix='river', drop_first=True)], axis=1)
datHousing.drop(['RIVER'], inplace=True, axis=1)
datHousing.head()
```

Out[11]:

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

In [13]:
```
np.mean(datHousing['river_Yes'])
```

Out[13]: 0.0691699604743083

3) R2 value is 0.74 and the MSE is 21.03

In [20]:
```
pd.Series(ml_reg.coef_, index=X.columns).sort_values(ascending=False).round(3)
```

Out[20]:
```
RM            3.614
river_Yes     3.252
RAD           0.246
ZN            0.048
AGE           0.007
INDUS        -0.009
TAX          -0.011
CRIM         -0.133
LSTAT        -0.566
PRATIO       -0.936
DIS          -1.425
NOX         -15.199
dtype: float64
```

In [21]:
```
ml_reg.intercept_.round(3)
```

Out[21]: 39.438

In [22]:
```
from sklearn.metrics import mean_squared_error, r2_score
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(y_train, y_pred_ml_reg))
```

Mean squared error: 21.03

In [23]:
```
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(y_train, y_pred_ml_reg))
```

Variance score: 0.74

4) datHousingSub created:

```
In [27]: datHousingSub = datHousing.drop(['MEDV'], axis=1)
```

```
In [28]: datHousingSub.head()
```

Out[28]:

	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

5) Kmeans 2 cluster model:

```
In [29]:
# Using scikit-learn to perform K-Means clustering
from sklearn.cluster import KMeans

# Specify the number of clusters (2) and fit the data datHousingSub
kmeans = KMeans(n_clusters=2, random_state=0).fit(datHousingSub)
```

```
In [30]:
centroids = (kmeans.cluster_centers_)
print(centroids)
```

```
[[3.88774444e-01 1.55826558e+01 8.42089431e+00 5.11847425e-01
  6.38800542e+00 6.06322493e+01 4.44127154e+00 4.45528455e+00
  3.11926829e+02 1.78092141e+01 1.04174526e+01 7.31707317e-02]
 [1.22991617e+01 3.01980663e-14 1.84518248e+01 6.70102190e-01
  6.00621168e+00 8.99678832e+01 2.05447007e+00 2.32700730e+01
  6.67642336e+02 2.01963504e+01 1.86745255e+01 5.83941606e-02]]
```

Based on the divergent characteristics, I would call cluster 1 “High crime industrial” and cluster 2 “Low crime non-industrial”

6) 3 cluster model:

```
In [32]:
# Specify the number of clusters (3) and fit the data datHousingSub
kmeans = KMeans(n_clusters=3, random_state=0).fit(datHousingSub)
```

```
In [33]:
centroids = (kmeans.cluster_centers_)
print(centroids)
```

```
[[2.44205703e-01 1.73764259e+01 6.70262357e+00 4.84713688e-01
  6.47416350e+00 5.61661597e+01 4.83579772e+00 4.32699620e+00
  2.75212928e+02 1.78733840e+01 9.55292776e+00 7.60456274e-02]
 [1.22991617e+01 3.01980663e-14 1.84518248e+01 6.70102190e-01
  6.00621168e+00 8.99678832e+01 2.05447007e+00 2.32700730e+01
  6.67642336e+02 2.01963504e+01 1.86745255e+01 5.83941606e-02]
 [7.47468585e-01 1.11320755e+01 1.26841509e+01 5.79169811e-01
  6.17423585e+00 7.17132075e+01 3.46240000e+00 4.77358491e+00
  4.03018868e+02 1.76500000e+01 1.25624528e+01 6.60377358e-02]]
```

In this case, I believe that the 2 cluster model works better than the 3 cluster model. I have determined this by calculating the silhouette scores for each. The 2 cluster score is higher, at 0.77, as opposed to the 3 cluster score, which is 0.62 (see below).

In [18]:

```python
# Calculate silhouette_score
from sklearn.metrics import silhouette_score

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

```
0.7717962604192585
```

In [19]:

```python
# Specify the number of clusters (3) and fit the data datHousingSub
kmeans = KMeans(n_clusters=3, random_state=0).fit(datHousingSub)
```

In [ ]:

```python
centroids = (kmeans.cluster_centers_)
print(centroids)
```

In [ ]:

```python
labels = (kmeans.labels_)
print(labels)
```

In [22]:

```python
# Calculate silhouette_score
from sklearn.metrics import silhouette_score

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

```
0.6217529916045139
```

7) New dataframe with Cluster column:

In [34]:

```python
# Specify the number of clusters (2) and fit the data datHousingSub
kmeans = KMeans(n_clusters=2, random_state=0).fit(datHousingSub)
```

In [35]:

```python
centroids = (kmeans.cluster_centers_)
print(centroids)
```

```
[[3.88774444e-01 1.55826558e+01 8.42089431e+00 5.11847425e-01
  6.38800542e+00 6.06322493e+01 4.44127154e+00 4.45528455e+00
  3.11926829e+02 1.78092141e+01 1.04174526e+01 7.31707317e-02]
 [1.22991617e+01 3.01980663e-14 1.84518248e+01 6.70102190e-01
  6.00621168e+00 8.99678832e+01 2.05447007e+00 2.32700730e+01
  6.67642336e+02 2.01963504e+01 1.86745255e+01 5.83941606e-02]]
```

In [37]:

```python
labels = (kmeans.labels_)
df_labels = pd.DataFrame(labels)
df_labels.rename(columns={0:'Cluster'}, inplace=True)
df_labels.head()
```

Out[37]:

	Cluster
0	0
1	0
2	0
3	0
4	0

8) datHousingC1 dataframe created:

In [40]:

```python
datHousingC1 = datHousing_Clust.loc[datHousing_Clust['Cluster']==0]
datHousingC2 = datHousing_Clust.loc[datHousing_Clust['Cluster']==1]
```

In [41]:

```python
datHousingC1.head()
```

Out[41]:

click to scroll output; double click to hide

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PRATIO	LSTAT	MEDV	river_Yes	Cluster
0	3.32105	0.0	19.58	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	13.4	1	0
[illegible]	[illegible]	[illegible]	[illegible]	.871	5.012	88.0	1.6102	5	403	14.7	12.12	15.3	1	0
2	1.41385	0.0	19.58	0.871	6.129	96.0	1.7494	5	403	14.7	15.12	17.0	1	0
3	3.53501	0.0	19.58	0.871	6.152	82.6	1.7455	5	403	14.7	15.02	15.6	1	0
4	1.27346	0.0	19.58	0.605	6.250	92.6	1.7984	5	403	14.7	5.50	27.0	1	0

9) datHousingC2 dataframe created:

In [42]:
```
datHousingC2.head()
```

Out[42]:

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PRATIO	LSTAT	MEDV	river_Yes	Cluster
27	8.98296	0.0	18.1	0.770	6.212	97.4	2.1222	24	666	20.2	17.60	17.8	1	1
28	3.84970	0.0	18.1	0.770	6.395	91.0	2.5052	24	666	20.2	13.27	21.7	1	1
29	5.20177	0.0	18.1	0.770	6.127	83.4	2.7227	24	666	20.2	11.48	22.7	1	1
30	4.22239	0.0	18.1	0.770	5.803	89.0	1.9047	24	666	20.2	14.64	16.8	1	1
31	3.47428	0.0	18.1	0.718	8.780	82.9	1.9047	24	666	20.2	5.29	21.9	1	1

10) Regression model from Cluster 1 (datHousingC1):

In [47]:
```
pd.Series(ml_reg.coef_, index=X.columns).sort_values(ascending=False).round(3)
```

Out[47]:
```
RM            9.168
river_Yes     1.045
CRIM          1.033
RAD           0.166
ZN            0.023
Cluster       0.000
INDUS        -0.001
TAX          -0.015
AGE          -0.045
LSTAT        -0.084
PRATIO       -0.530
DIS          -0.836
NOX          -6.519
dtype: float64
```

In [48]:
```
ml_reg.intercept_.round(3)
```

Out[48]:
```
-10.54
```

In [49]:
```
from sklearn.metrics import mean_squared_error, r2_score
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(y_train, y_pred_ml_reg))
```

```
Mean squared error: 9.95
```

In [50]:
```
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(y_train, y_pred_ml_reg))
```

```
Variance score: 0.86
```

The R2 value is 0.86 which is higher than the baseline model, meaning it provides a more accurate prediction. The MSE is 9.95 which is lower than the baseline model, again indicating a better prediction.

11) Regression model from Cluster 2 (datHousingC2):

In [55]:
```
pd.Series(ml_reg.coef_, index=X.columns).sort_values(ascending=False).round(3)
```

Out[55]:
```
RAD          1.073362e+11
TAX          7.370158e+10
river_Yes    1.032100e+01
AGE          1.400000e-02
Cluster      0.000000e+00
CRIM        -1.380000e-01
LSTAT       -8.440000e-01
RM          -9.750000e-01
DIS         -3.321000e+00
NOX         -3.621100e+01
ZN          -7.619852e+07
PRATIO      -6.187353e+09
INDUS       -1.214176e+11
dtype: float64
```

In [56]:
```
ml_reg.intercept_.round(3)
```

Out[56]:
```
-49338680212718.555
```

In [57]:
```
from sklearn.metrics import mean_squared_error, r2_score
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(y_train, y_pred_ml_reg))
```

```
Mean squared error: 20.13
```

In [58]:
```
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(y_train, y_pred_ml_reg))
```

```
Variance score: 0.73
```

The R2 score is 0.73 and the MSE is 20.13, which are both in line with the baseline model.

12) Summary of findings: Yes, I feel that clustering can help improve my ability to predict MEDV, given that it allows me to narrow down more specifically into different types of areas, i.e. Industrial or Non-Industrial and so on. This allows me to apply different specific models for each type of property and arrive at a more accurate prediction than I might get using a model for the general overall population.