In [21]:
```
# Packages
from pathlib import Path

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import NearestNeighbors, KNeighborsRegressor
import matplotlib.pylab as plt

import dmba

%matplotlib inline
```

Data Description:

- `crim` — per capita crime rate by town
- `zn` — proportion of residential land zoned for lots over 25,000 sq.ft
- `indus` — proportion of non-retail business acres per town
- `chas` — Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `nox` — nitric oxides concentration (parts per 10 million)
- `rm` — average number of rooms per dwelling
- `age` — proportion of owner-occupied units built prior to 1940
- `dis` — weighted distances to five Boston employment centers
- `rad` — index of accessibility to radial highways
- `tax` — full-value property-tax rate per USD 10,000
- `ptratio` — pupil-teacher ratio by town
- `lstat` — percentage of lower status of the population
- `medv` — median value of owner-occupied homes in USD 1000's
- `cat.medv` — whether housing price is expensive (= 1 for above 30; 0 otherwise)

In [22]:
```
# Loading in and verifying dataset
housing = pd.read_csv('Datasets/BostonHousing.csv')
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   CRIM       506 non-null    float64
 1   ZN         506 non-null    float64
 2   INDUS      506 non-null    float64
 3   CHAS       506 non-null    int64  
 4   NOX        506 non-null    float64
 5   RM         506 non-null    float64
 6   AGE        506 non-null    float64
 7   DIS        506 non-null    float64
 8   RAD        506 non-null    int64  
 9   TAX        506 non-null    int64  
 10  PTRATIO    506 non-null    float64
 11  LSTAT      506 non-null    float64
 12  MEDV       506 non-null    float64
 13  CAT. MEDV  506 non-null    int64  
dtypes: float64(10), int64(4)
memory usage: 55.5 KB
```

In [ ]:
```python
# Drop ['CAT. MEDV']
housing = housing.drop(columns = 'CAT. MEDV')
```

In [36]:
```python
# Preview
housing.head()
```

Out[36]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LS
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5

In [25]:
```python
# Nulls?
housing.isnull().sum()
```

Out[25]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
LSTAT      0
MEDV       0
dtype: int64
```

In [26]:
```python
# Split features and target
X = housing.drop(columns='MEDV', axis=1)
y = housing['MEDV']
```

In [27]:
```python
# Split into training/test sets, 40% test size
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, rar

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [28]:
```python
# Train knn model using k values between 1 and 10
knn = KNeighborsRegressor

for k in range (1, 11):
    # Initialize kNN model
    knn = KNeighborsRegressor(n_neighbors=k)
    # Fit with training set
    knn.fit(X_train, y_train)
    # Make predictions
    y_pred = knn.predict(X_test)
    # Print RMSE
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print(f'n_neighbors: {k}, RMSE: {rmse:.04f}')
```

```
n_neighbors: 1, RMSE: 5.4032
n_neighbors: 2, RMSE: 4.7786
n_neighbors: 3, RMSE: 4.6718
n_neighbors: 4, RMSE: 4.7892
n_neighbors: 5, RMSE: 5.0148
n_neighbors: 6, RMSE: 4.9318
n_neighbors: 7, RMSE: 4.9254
n_neighbors: 8, RMSE: 5.0255
n_neighbors: 9, RMSE: 5.0588
n_neighbors: 10, RMSE: 5.2331
```

The optimal value of k represents the optimal number of clusters to divide the data into when making predictions. In this case, the optimal value of k is 3 which resulted in the lowest root mean squared error.

In [35]:

```
# Predict MEDV for new tract

# New observation
new_tract = pd.DataFrame([{'CRIM':0.2, 'ZN':0, 'INDUS':7, 'CHAS':1, 'NOX':0.
'RAD':4, 'TAX':200, 'PTRATIO':21, 'LSTAT':20}])

# Standardize
new_tract = scaler.transform(new_tract)

# Predict with trained model with k=3
knn = KNeighborsRegressor(n_neighbors=3)
knn.fit(X_train, y_train)
MEDV_pred = knn.predict(new_tract)

print(f'Predicted MEDV for new observation: {MEDV_pred[0]:.02f}')
```

```
Predicted MEDV for new observation: 21.60
```