In [2]:

```
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn import preprocessing
from yellowbrick.cluster import KElbowVisualizer
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import colors as mcolors
colors = dict(mcolors.BASE_COLORS, **mcolors.CSS4_COLORS)



pd.set_option('display.max_columns', 500)
boston_data = load_boston()
df = pd.DataFrame(boston_data.data,columns=boston_data.feature_names)
df['target'] = pd.Series(boston_data.target)
```

In [3]:

```
def poly_fitting(column):
    # I would like this to return not just any poly_eq but one that asymptotically matches both training and test data
    X = df[[column]]
    y = df['target']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15)
    np.ployfit(X_train, y_train)
```

In [4]:

```
X = df['RM']
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15)
data = np.column_stack((X_train, y_train))
print(data[1:10])
```

```
[[ 6.167 19.9  ]
 [ 6.405 22.   ]
 [ 6.273 24.1  ]
 [ 6.619 23.9  ]
 [ 6.02  23.2  ]
 [ 7.489 50.   ]
 [ 5.404 19.3  ]
 [ 6.511 25.   ]
 [ 6.29  23.5  ]]
```

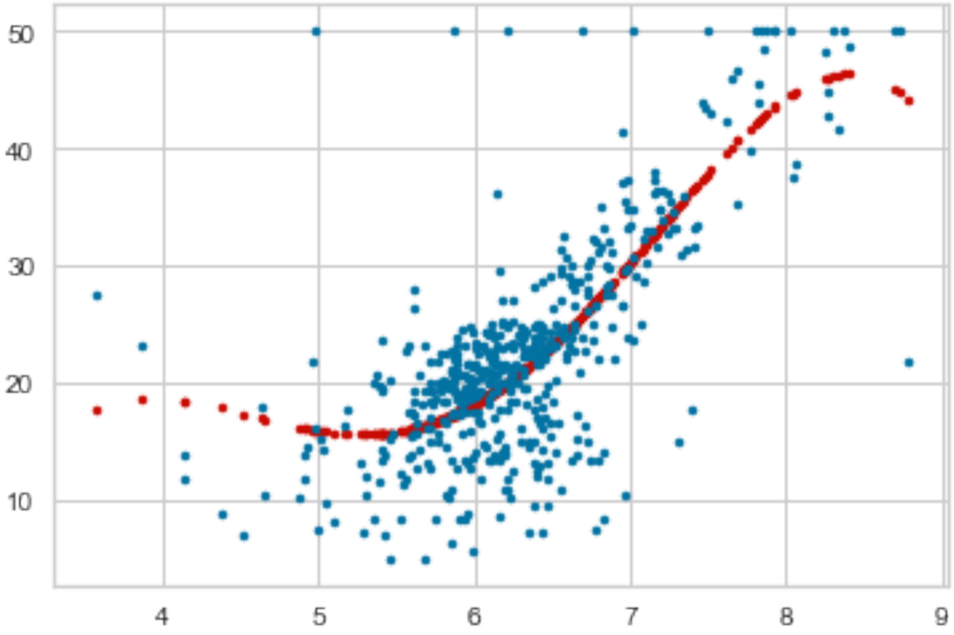
In [5]:

```
X = df['RM']
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15)
ploy = np.polyfit(X_train, y_train, 4)
p = np.poly1d(ploy)

y_train_pred = p(X_train)
RootMeanSqError = (np.sqrt(mean_squared_error(y_train, y_train_pred)))
RSq = r2_score(y_train,y_train_pred)

print(f"\n Selected featureset based on location, neighbours, employement and education  \n")
print("\033[0;32;23m  Training Results")
print("\033[0;30;0m")
print("    R2 score is: ", RSq)
print("    RootMeanSqError computes: ", RootMeanSqError)

y_test_pred = p(X_test)

RootMeanSqError = (np.sqrt(mean_squared_error(y_test, y_test_pred)))
RSq = r2_score(y_test, y_test_pred)


print("\033[0;31;23m\n  Testing Results")
print("\033[0;30;0m")
print("    R2 score is: ", RSq)
print("    RootMeanSqError computes: ", RootMeanSqError)

print('\n \n')

y = p(df['RM'])
x = df['RM']
plt.plot(x,y,'r.')
plt.plot(x, df['target'],'b.')
```

```
 Selected featureset based on location, neighbours, employement an
d education

  Training Results

    R2 score is:  0.5672494097595028
    RootMeanSqError computes:  6.088328700811282

  Testing Results

    R2 score is:  0.6052155556219498
    RootMeanSqError computes:  5.516711868612687
```

Out[5]:

```
[<matplotlib.lines.Line2D at 0x1290e7390>]
```



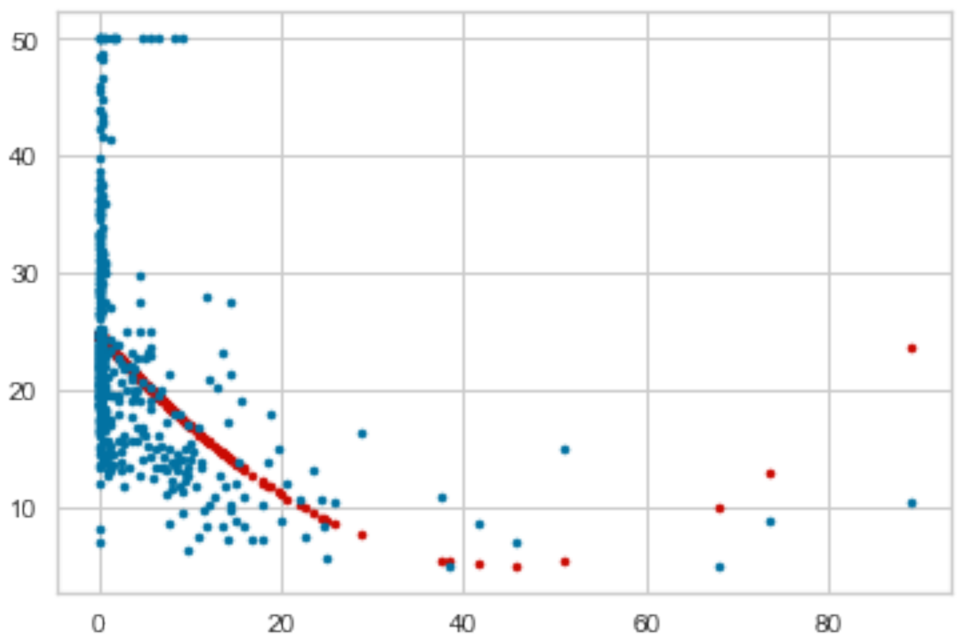
In [6]:

```
def poly_fitting(column):
    X = df[column]
#     X = preprocessing.scale(X)
    y = df['target']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15
)
    ploy = np.polyfit(X_train, y_train, 2)
    p = np.poly1d(ploy)

    y_train_pred = p(X_train)
    RootMeanSqError = (np.sqrt(mean_squared_error(y_train, y_train_pred)))
    RSq = r2_score(y_train,y_train_pred, multioutput='variance_weighted')

    print(f"\n Feature {column} yields  \n")
    print("\033[0;32;23m  Training Results")
    print("\033[0;30;0m")
    print("    R2 score is: ", RSq)
    print("    RootMeanSqError computes: ", RootMeanSqError)

    y_test_pred = p(X_test)

    RootMeanSqError = (np.sqrt(mean_squared_error(y_test, y_test_pred)))
    RSq = r2_score(y_test, y_test_pred)


    print("\033[0;31;23m\n  Testing Results")
    print("\033[0;30;0m")
    print("    R2 score is: ", RSq)
    print("    RootMeanSqError computes: ", RootMeanSqError)

    print('\n')

    y = p(df[column])
    x = df[column]
    plt.plot(x,y,'r.')
    plt.plot(x, df['target'],'b.')
```

In [7]:

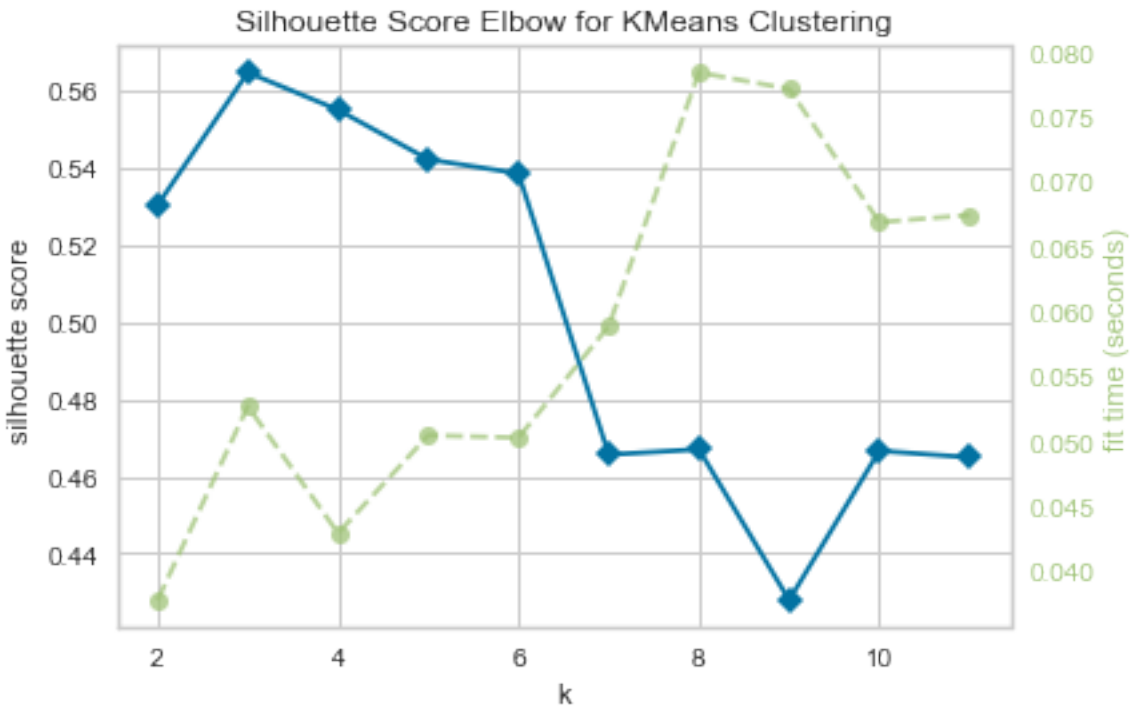
```
# poly_fitting('LSTAT')
poly_fitting('CRIM')
# poly_fitting('PTRATIO')
# print(np.round(df['PTRATIO']))
```

```
Feature CRIM yields

 Training Results

   R2 score is:  0.2032433885811561
   RootMeanSqError computes:  8.156614947713631

 Testing Results

   R2 score is:  0.2157507412761951
   RootMeanSqError computes:  8.311341756582413
```



In [23]:

```
# K Means

# O = df[['RM']]
# O = preprocessing.scale(O)
# K = df[['target']]
# K = preprocessing.scale(K)
# L = np.column_stack((O, K))
L = df[['CRIM', 'target']]
# L = preprocessing.scale(L)

mets = ['silhouette', 'calinski_harabaz', 'distortion']
km = KMeans()
visualize = KElbowVisualizer(km, k = (2,12),metric=mets[0])
visualize.fit(L)
visualize.poof()
# print(L[:,0])
```



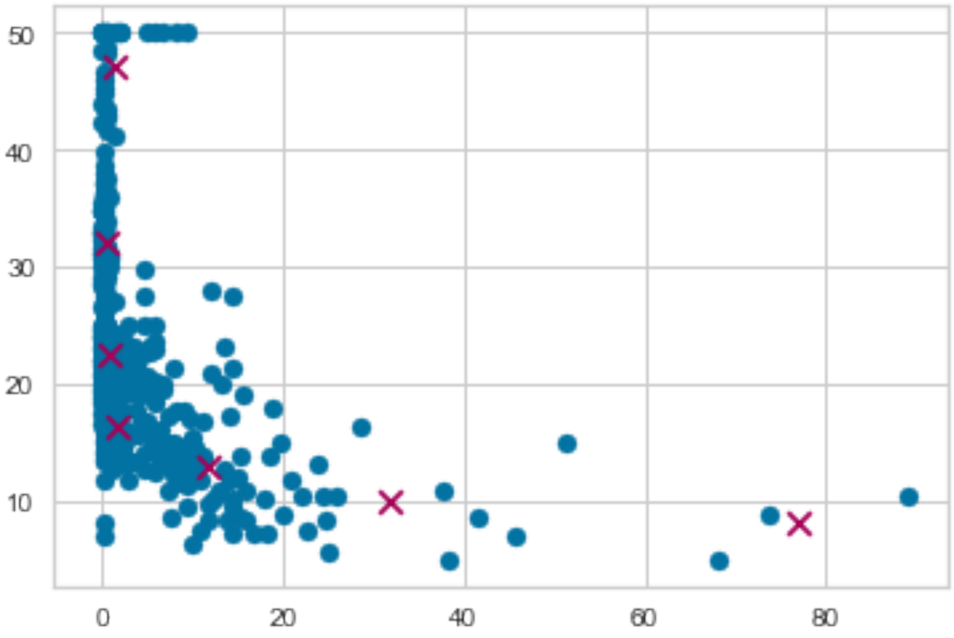
In [24]:

```
# poly_fitting('NOX')
km = KMeans(n_clusters = 7)
km.fit(L)
centroids = km.cluster_centers_
print(centroids)

colors = ['azure', 'darkolivegreen', 'darkblue', 'm', 'y', 'green', 'r', 'papa
yawhip' ]

labels = km.labels_

# plt.scatter(L[:,0], L[:,1])
plt.scatter(L['CRIM'], L["target"])

# print(df['RAD'].value_counts())
# radius = sorted(list(df['RAD'].value_counts()))
# print(radius)
# r1 = [7, 1, 2, 8, 6, 3, 4, 5, 24]
# plt.scatter(radius, r1 )

plt.scatter(centroids[:, 0], centroids[:, 1], marker = 'x',c='m', s=75, linewi
dths = 5, zorder=10)
```

```
[[ 0.75896413 22.41634615]
 [31.6584      9.88461538]
 [ 0.42920377 32.01428571]
 [76.81036667  8.06666667]
 [ 1.63512636 16.41214953]
 [11.70154015 12.94393939]
 [ 1.50872562 47.2125    ]]
```

Out[24]:

```
<matplotlib.collections.PathCollection at 0x12b56df60>
```



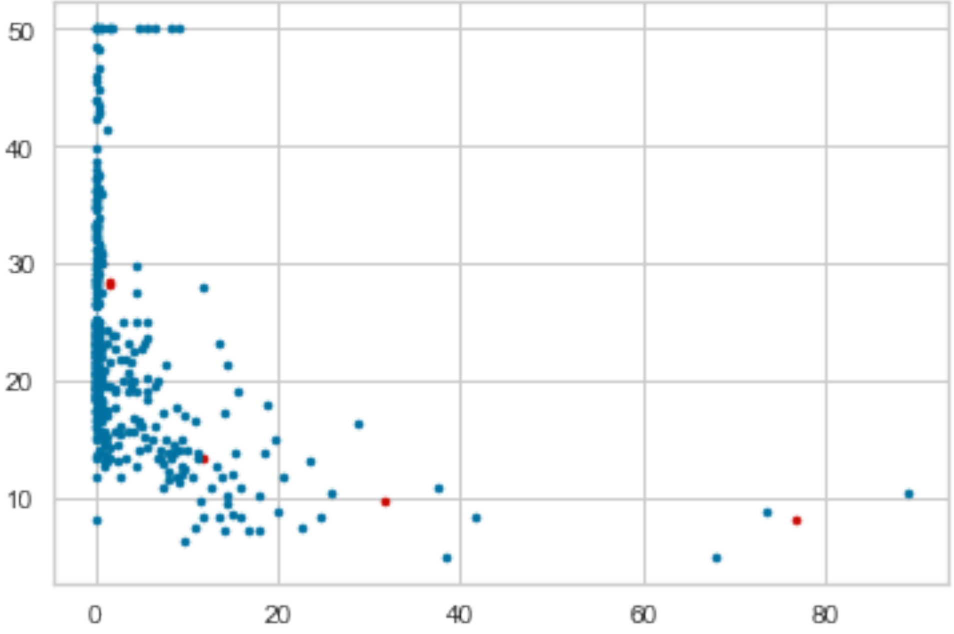
In [28]:

```
X = df['CRIM']
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15)

cx = centroids[:,0]
cy = centroids[:,1]
ploy = np.polyfit(cx, cy, 3)
p = np.poly1d(ploy)


y_train_pred = p(X_train)
RootMeanSqError = (np.sqrt(mean_squared_error(y_train, y_train_pred)))
RSq = r2_score(y_train,y_train_pred)

print(f"\n Selected featureset based on location, neighbours, employement and education  \n")
print("\033[0;32;23m  Training Results")
print("\033[0;30;0m")
print("    R2 score is: ", RSq)
print("    RootMeanSqError computes: ", RootMeanSqError)

y_test_pred = p(X_test)

RootMeanSqError = (np.sqrt(mean_squared_error(y_test, y_test_pred)))
RSq = r2_score(y_test, y_test_pred)


print("\033[0;31;23m\n  Testing Results")
print("\033[0;30;0m")
print("    R2 score is: ", RSq)
print("    RootMeanSqError computes: ", RootMeanSqError)

print('\n \n')


y = p(cx)
x = cx
plt.plot(x,y,'r.')

plt.plot(X_train, y_train,'b.')
```

```
 Selected featureset based on location, neighbours, employement an
d education

  Training Results

    R2 score is:  -0.12587227541496815
    RootMeanSqError computes:  9.932717953921998

  Testing Results

    R2 score is:  -0.3983653239399405
    RootMeanSqError computes:  9.412603407240697
```

Out[28]:

```
[<matplotlib.lines.Line2D at 0x12ba83ac8>]
```



In [29]:

```
# help(preprocessing.scale)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: