In [4]:

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline


pd.set_option('display.max_columns', 500)
boston_data = load_boston()

# print(df)

df = pd.DataFrame(boston_data.data,columns=boston_data.feature_names)
df['target'] = pd.Series(boston_data.target)
print(df.describe())
```

	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

	AGE	DIS	RAD	TAX	PTRATIO	B
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032
std	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000
25%	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000
75%	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000

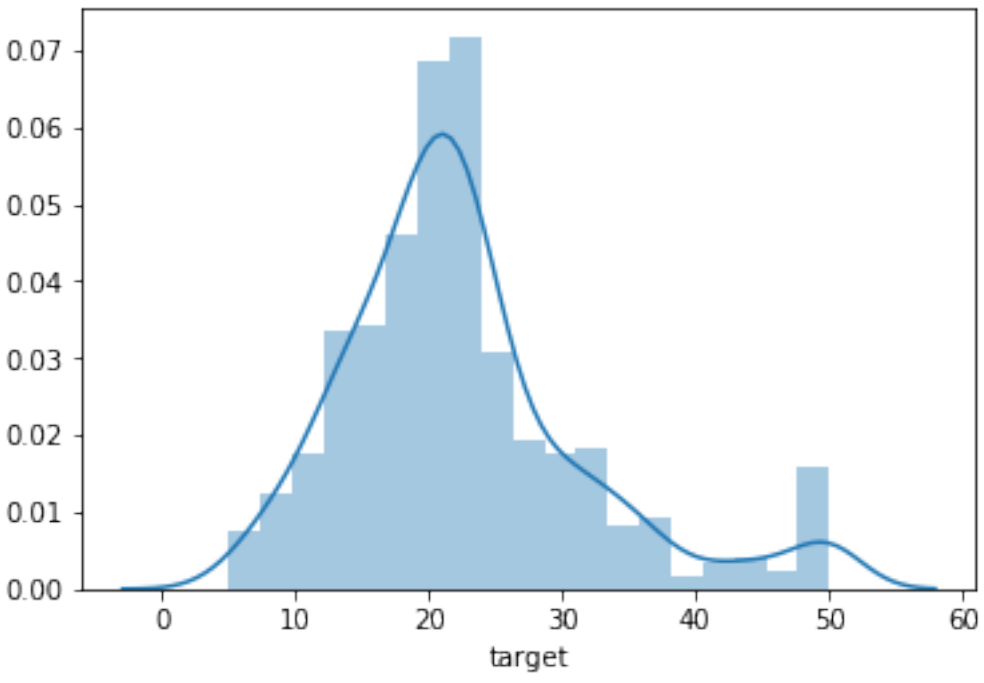
	LSTAT	target
count	506.000000	506.000000
mean	12.653063	22.532806
std	7.141062	9.197104
min	1.730000	5.000000
25%	6.950000	17.025000
50%	11.360000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

In [5]:

```
sns.distplot(df['target'], bins=19)
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1040da208>
```



In [6]:

```
my_col_picks = ['CRIM', 'PTRATIO', 'RAD']
# I am going to evaluate a linear model for a set of combinations while keepin
g some columns fixed
# I chose rooms, neighbourhoud and employment as a primary indicator for price
of house.

print("Keeping the RM, LSTAT and DIS columns fixed, evaluating the following c
olumns for a Linear Model! \n")
for choice in my_col_picks:

    X = pd.DataFrame(np.c_[df['RM'],df['LSTAT'],df['DIS'], df[f"{choice}"]], c
olumns = ['RM','LSTAT','DIS',f"{choice}"])
    y = df['target']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15
)
    linear_model = LinearRegression()
    linear_model.fit(X_train, y_train)

    y_train_pred = linear_model.predict(X_train)
    RootMeanSqError = (np.sqrt(mean_squared_error(y_train, y_train_pred)))
    RSq = linear_model.score(X_train, y_train)

    print(f"\nTraining performance with {choice} the training results are \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 = linear_model.predict(X_test)

    RootMeanSqError = (np.sqrt(mean_squared_error(y_test, y_test_pred)))
    RSq = linear_model.score(X_test, y_test)


    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')
```

Keeping the RM, LSTAT and DIS columns fixed, evaluating the follow
ing columns for a Linear Model!

Training performance with CRIM the training results are

Training Results

R2 score is: 0.6306890279139064
RootMeanSqError computes: 5.3839579496617835

Testing Results

R2 score is: 0.746406543420154
RootMeanSqError computes: 5.44418962081807

Training performance with PTRATIO the training results are

Training Results

R2 score is: 0.6785176162989659
RootMeanSqError computes: 5.0647309040889885

Testing Results

R2 score is: 0.7269521645710901
RootMeanSqError computes: 5.460188956483667

Training performance with RAD the training results are

Training Results

R2 score is: 0.6645982700297816
RootMeanSqError computes: 5.404946416634118

Testing Results

R2 score is: 0.5311634223233317
RootMeanSqError computes: 5.395494886571268

In [7]:

```
#model scheme
X = pd.DataFrame(np.c_[df['DIS'], df['RM'], df['LSTAT'], df['PTRATIO']], colum
ns = ['DIS','RM', 'LSTAT', 'PTRATIO'])
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15, ra
ndom_state = 77)
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

y_train_pred = linear_model.predict(X_train)
RootMeanSqError = (np.sqrt(mean_squared_error(y_train, y_train_pred)))
RSq = linear_model.score(X_train,y_train)

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 = linear_model.predict(X_test)

RootMeanSqError = (np.sqrt(mean_squared_error(y_test, y_test_pred)))
RSq = linear_model.score(X_test, y_test)


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')
```

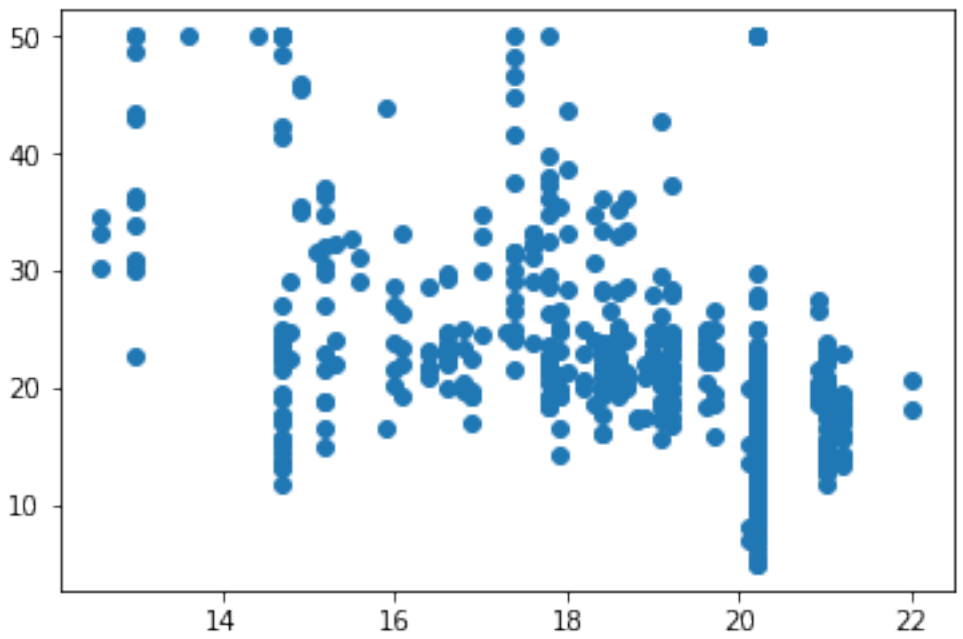
```
 Selected featureset based on location, neighbours, employement an
d education

  Training Results

    R2 score is:  0.6920300104192643
    RootMeanSqError computes:  5.2452044252813375

  Testing Results

    R2 score is:  0.6449132375849227
    RootMeanSqError computes:  4.424287799686455
```

In [9]:

```
x = df['PTRATIO']
y = df['target']
plt.scatter(x,y)
#plt.plot(x, np.poly1d(np.polyfit(x, y, 1))(x),'r--')
plt.plot()
```

Out[9]:

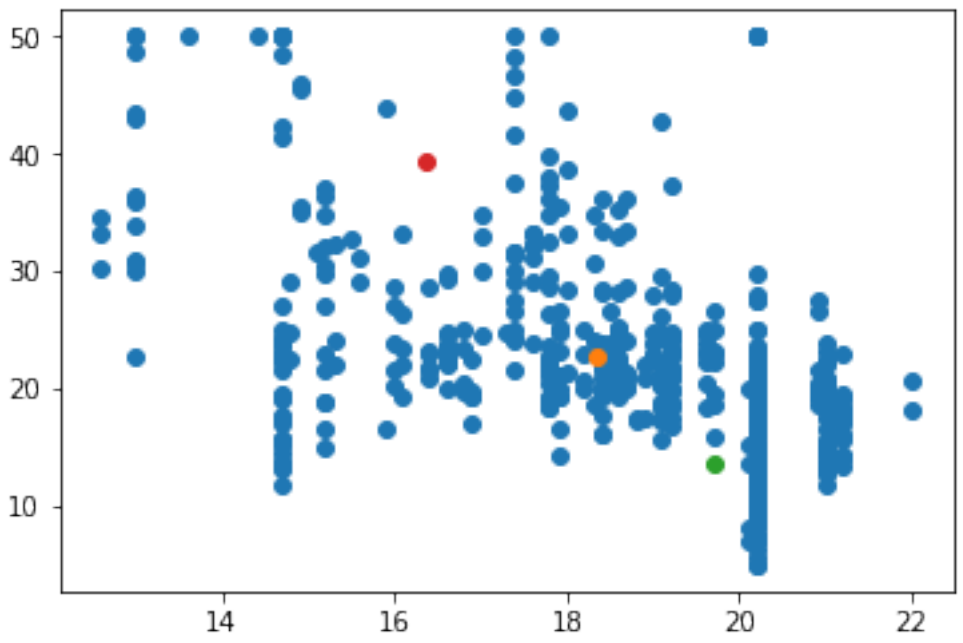
[]



In [11]:

```
Kx = df[['PTRATIO', 'target']].values
kmeans = KMeans(n_clusters=3).fit(Kx)
centroids = kmeans.cluster_centers_
print(centroids)
x = df['PTRATIO']
y = df['target']
plt.scatter(x,y)
for i in range(len(centroids)):
    plt.scatter(centroids[i][0],centroids[i][1],)
```

```
[[18.35036496 22.74233577]
 [19.70324675 13.59155844]
 [16.36153846 39.45      ]]
```

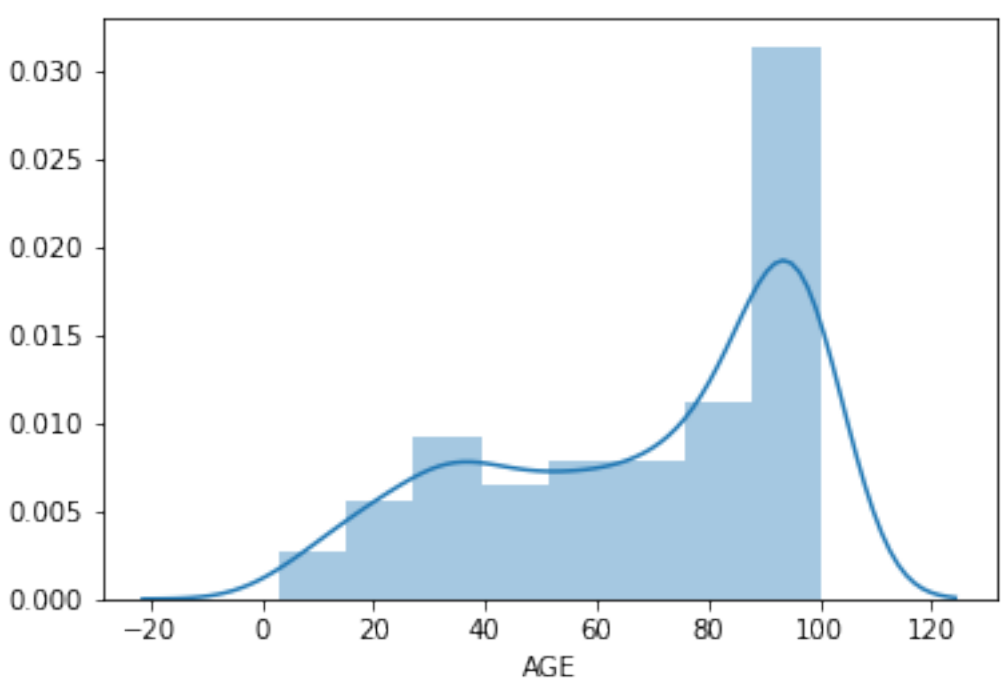


In [13]:

```
sns.distplot(df["AGE"])
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x122f93ef0>
```



In [ ]:

In [ ]:

In [ ]: