In [14]:

```
# Baseline Approach
import numpy
from numpy import arange
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error

# Load dataset
filename = 'D:\\Dataset\housing.csv'
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX
dataset = read_csv(filename, delim_whitespace=True, names=names)

# shape
#print(dataset.shape)
# types
#print(dataset.dtypes)
#head
#print(dataset.head(20))
#print(dataset.describe())

#correlation
#set_option('display.precision',2)
#print(dataset.corr(method='pearson'))

# Split-out validation dataset
array = dataset.values
X = array[:,0:13]
Y = array[:,13]
validation_size = 0.20
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_siz

# Test options and evaluation metric
num_folds = 10
scoring = 'neg_mean_squared_error'

# Spot-Check Algorithms
```

```python
models = []
models.append(('LR', LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('SVR', SVR()))

# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=num_folds)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=sc
    results.append(cv_results)
    names.append(name)
    print(name, cv_results.mean())
```

```
LR -23.908033788422156
LASSO -28.0448870703122
EN -28.175123417894305
KNN -42.17117773170732
CART -25.10081219512195
SVR -70.84599513338853
```

In [18]:

```
# Standardization Approach
import numpy
from numpy import arange
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error

# Load dataset
filename = 'D:\\Dataset\housing.csv'
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX
dataset = read_csv(filename, delim_whitespace=True, names=names)

# shape
#print(dataset.shape)
# types
#print(dataset.dtypes)
#head
#print(dataset.head(20))
#print(dataset.describe())

#correlation
#set_option('display.precision',2)
#print(dataset.corr(method='pearson'))

# Split-out validation dataset
array = dataset.values
X = array[:,0:13]
Y = array[:,13]
validation_size = 0.20
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_siz

# Test options and evaluation metric
num_folds = 10
scoring = 'neg_mean_squared_error'

# Standardize the dataset
pipelines = []
```

```
pipelines.append(('ScaledLR', Pipeline([('Scaler', StandardScaler()),('LR',Lin
pipelines.append(('ScaledLASSO', Pipeline([('Scaler', StandardScaler()),('LASS
pipelines.append(('ScaledEN', Pipeline([('Scaler', StandardScaler()),('EN', El
pipelines.append(('ScaledKNN', Pipeline([('Scaler', StandardScaler()),('KNN',
pipelines.append(('ScaledCART', Pipeline([('Scaler', StandardScaler()),('CART'
pipelines.append(('ScaledSVR', Pipeline([('Scaler', StandardScaler()),('SVR',

# evaluate each model in turn
results = []
names = []
for name, model in pipelines:
    kfold = KFold(n_splits=num_folds)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=sc
    results.append(cv_results)
    names.append(name)
    print(name, cv_results.mean())

# KNN Algorithm tuning
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

k_values = numpy.array([1,3,5,7,9,11,13,15,17,19,21])
param_grid = dict(n_neighbors=k_values)
model = KNeighborsRegressor()
kfold = KFold(n_splits=num_folds)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, c
grid_result = grid.fit(rescaledX, Y_train)
print(grid_result.best_score_, grid_result.best_params_)
```

```
ScaledLR -21.805142053793766
ScaledLASSO -26.638284280189385
ScaledEN -28.508444245973287
ScaledKNN -17.33440097560976
ScaledCART -22.343270121951218
ScaledSVR -26.830711144817805
-16.88772296747968 {'n_neighbors': 3}
```

In [22]:

```
# Ensemble Approach
import numpy
from numpy import arange
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error

# Load dataset
filename = 'D:\\Dataset\housing.csv'
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX
dataset = read_csv(filename, delim_whitespace=True, names=names)

# shape
#print(dataset.shape)
# types
#print(dataset.dtypes)
#head
#print(dataset.head(20))
#print(dataset.describe())

#correlation
#set_option('display.precision',2)
#print(dataset.corr(method='pearson'))

# Split-out validation dataset
array = dataset.values
X = array[:,0:13]
Y = array[:,13]
validation_size = 0.20
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_siz

# Test options and evaluation metric
num_folds = 10
scoring = 'neg_mean_squared_error'

# ensembles
ensembles = []
```

```
ensembles.append(('ScaledAB', Pipeline([('Scaler', StandardScaler()),('AB',  A
ensembles.append(('ScaledGBM', Pipeline([('Scaler', StandardScaler()),('GBM', 
ensembles.append(('ScaledRF', Pipeline([('Scaler', StandardScaler()),('RF', Ra
ensembles.append(('ScaledET', Pipeline([('Scaler', StandardScaler()),('ET', Ex

results = []
names = []
for name, model in ensembles:
    kfold = KFold(n_splits=num_folds)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=sc
    results.append(cv_results)
    names.append(name)
    print(name, cv_results.mean())

# Tune scaled GBM
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
param_grid = dict(n_estimators=numpy.array([50,100,150,200,250,300,350,400]))
model = GradientBoostingRegressor()
kfold = KFold(n_splits=num_folds)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, c
grid_result = grid.fit(rescaledX, Y_train)
print(grid_result.best_score_, grid_result.best_params_)
```

```
ScaledAB -15.321239084956028
ScaledGBM -9.323988375157091
ScaledRF -12.140909998963416
ScaledET -10.4813758827439
-9.281621488756766 {'n_estimators': 100}
```

In [26]:

```
# Finalized model
import numpy
from numpy import arange
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error

# Load dataset
filename = 'D:\\Dataset\housing.csv'
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX
dataset = read_csv(filename, delim_whitespace=True, names=names)

# shape
#print(dataset.shape)
# types
#print(dataset.dtypes)
#head
#print(dataset.head(20))
#print(dataset.describe())

#correlation
#set_option('display.precision',2)
#print(dataset.corr(method='pearson'))

# Split-out validation dataset
array = dataset.values
X = array[:,0:13]
Y = array[:,13]
validation_size = 0.20
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_siz

# Test options and evaluation metric
num_folds = 10
scoring = 'neg_mean_squared_error'


# prepare the model
```

```python
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = GradientBoostingRegressor(n_estimators=100)
model.fit(rescaledX, Y_train)

# transform the validation dataset
rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))
```

```
6.695555724225275
```

In [ ]: