```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import ensemble
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPRegressor
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```

```
df = pd.read_csv('housing-1.csv')
df
```

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0
...	...	...	...	...
484	6.593	9.67	21.0	470400.0
485	6.120	9.08	21.0	432600.0
486	6.976	5.64	21.0	501900.0
487	6.794	6.48	21.0	462000.0
488	6.030	7.88	21.0	249900.0

489 rows × 4 columns

```
x = df[['RM', 'LSTAT', 'PTRATIO']]
y = df['MEDV']
```

```
print(x.shape)
print(y.shape)
```

```
(489, 3)
(489,)
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = .20, random_state= 0)
```

```
from sklearn.ensemble import BaggingRegressor
from sklearn.model_selection import cross_val_score
```

```
lr = LinearRegression()
gbr = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 2, min_samples_sp
rf = RandomForestRegressor(n_estimators=1000, max_depth=5)
mlpr = MLPRegressor(hidden_layer_sizes=(4,32,64,128,64,32,1), max_iter=4000)
xgbr = XGBRegressor(n_estimators=300, max_depth='2')
'''dlr = Sequential([
    Dense(32, activation='relu', input_shape = (4,)),
    Dense(32, activation='relu'),
    Dense(1, activation='relu'),
])

dlr.compile(optimizer='adam',
              loss='mean_squared_error')'''
clf_array = [lr, gbr, rf, mlpr, xgbr]
for clf in clf_array:
    vanilla_clf = clf
    vanilla_clf.fit(x_train, y_train)
    bagging_clf = BaggingRegressor(clf,
       max_samples=0.4, random_state=1075)
    bagging_clf.fit(x_train, y_train)
    '''bagging_scores = cross_val_score(bagging_clf, x, y, cv=10,
       n_jobs=-1)'''

    print("Score: " + str(vanilla_clf.score(x_test, y_test)) + "[" + str(vanilla_clf) + "]
    print("Score: " + str(bagging_clf.score(x_test, y_test)) + "[Bagging" + str(vanilla_cl
```

```
Score: 0.6574622113312862[LinearRegression(copy_X=True, fit_intercept=True, n_job
Score: 0.6520947284410252[BaggingLinearRegression(copy_X=True, fit_intercept=True
Score: 0.8111485475962935[GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, cri
                          init=None, learning_rate=0.1, loss='ls', max_depth=2,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=400,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)]
Score: 0.8073944232676276[BaggingGradientBoostingRegressor(alpha=0.9, ccp_alpha=0
                          init=None, learning_rate=0.1, loss='ls', max_depth=2,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
```

```
                          min_weight_fraction_leaf=0.0, n_estimators=400,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)]
Score: 0.8201488084307017[RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, cr
                      max_depth=5, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)]
Score: 0.8115675575781444[BaggingRandomForestRegressor(bootstrap=True, ccp_alpha=
                      max_depth=5, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)]
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_percept
  % self.max_iter, ConvergenceWarning)
Score: -7.486529990626431[MLPRegressor(activation='relu', alpha=0.0001, batch_siz
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(4, 32, 64, 128, 64, 32, 1),
             learning_rate='constant', learning_rate_init=0.001, max_fun=15000,
             max_iter=4000, momentum=0.9, n_iter_no_change=10,
             nesterovs_momentum=True, power_t=0.5, random_state=None,
             shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
             verbose=False, warm_start=False)]
Score: -3.1646918692333488[BaggingMLPRegressor(activation='relu', alpha=0.0001, b
```

```
from sklearn.ensemble import VotingRegressor
```

```
eclf = VotingRegressor([('lr',lr), ('gbr',gbr), ('rf',rf), ('mlpr',mlpr), ('xgbr',xgbr)])
```

```
eclf.fit(x_train, y_train)
eclf.score(x_test, y_test)
```

```
[19:40:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron
  % self.max_iter, ConvergenceWarning)
0.46227786389967546
```

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import RepeatedKFold
from sklearn.ensemble import StackingRegressor
```

```
def get_models():
  models = dict()
  models['lr'] = LinearRegression()
  models['gbr'] = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 2, mi
  models['rf'] = RandomForestRegressor(n_estimators=1000, max_depth=5)
  models['mlpr'] = MLPRegressor(hidden_layer_sizes=(4,32,64,128,64,32,1), max_iter=4000)
  models['xgbr'] = XGBRegressor(n_estimators=300, max_depth='2')
  return models
```

```
def get_stacking():
  # define the base models
  level0 = list()
  #level0.append(('lr', LinearRegression()))
  level0.append(('gbr', ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth =
  level0.append(('rf', RandomForestRegressor(n_estimators=1000, max_depth=5)))
  #level0.append(('mlpr', MLPRegressor(hidden_layer_sizes=(4,32,64,128,64,32,1), max_iter=
  level1 =  XGBRegressor(n_estimators=300, max_depth='2')
  # define the stacking ensemble
  model = StackingRegressor(estimators=level0, final_estimator=level1, cv=5)
  return model
```

```
def evaluate_model(model, X, y):
  cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
  scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-
  return scores
```

```
models = get_stacking()
models.fit(x_train, y_train)
models.score(x_test, y_test)
```

```
[20:27:35] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is nov
0.7716293700810974
```