In [ ]:

```
#*****************************************************************************************
#
#                        Program : To predict House Price based on Random Forest Model
#
#*****************************************************************************************

# Step0 : Loading of Packages and loading the Data Set

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_score,accuracy_score

# Loading the standard Boston House Data
boston = datasets.load_boston()

# Columns /Features in the Boston Data
print(boston.feature_names)
print('-'*80)

# understanding the meaning of the various features/Atrributes of Boston Data
print(boston.DESCR)
print('-'*80)
# Creating a DataFrame of the Boston Data

HouseData = pd.DataFrame(boston.data, columns=boston.feature_names)
print(" The Boston Data for Predicting the House price \n",'-'*80)
print(HouseData.head())
print('-'*80)

# Housing price
targets = boston.target

```

In [ ]:
```
# Step1 : Analysing the Data

# including the target column in the main dataframe ,to analyse the data together
HouseData['target'] = targets

print(" Analysing The Data , by using info, describe functions , checking for nulls and size")
print(HouseData.info())
print('-'*80)

print(HouseData.shape)
print('-'*80)

print(HouseData.describe())
print('-'*80)

print(HouseData.isnull().sum())
print('-'*80)

print(" Observing the Data , We see  no null values , and 506 entries and 14 features \n")

```

In [ ]:
```
# Step2 : Visualising the Data

# visualise the data

features = boston.feature_names
y = 'target'

print(" Visualising the Data to see the relationship with House Price")
column_cnt = len(features)
i = 0
while i< column_cnt:
    sns.pairplot(HouseData,x_vars=features[i:i+4],y_vars=y,kind="reg")
    i = i+4


```

In [ ]:
```
# Step3 : Preprocessing the Data
# since the target is continuous Data , we will make it discreet  by using the median value
# converting the taget field to binary

# the median value of the housing price
med = np.median(targets)

HouseData['target_num'] = HouseData.target.map(lambda x: 1 if x>med else 0)
print(HouseData.head(2))
```

In [ ]:
```
# Step4 : Creating the Train and  Test sets by  splitting the Data

y = HouseData['target_num']

print(y.shape)
y = y.values.reshape(-1,1)

X = HouseData.drop(['target','target_num'],axis=1)
features = ['RM','AGE','DIS','B','CRIM','ZN','INDUS','CHAS','NOX','RAD','TAX','PTRATIO','LSTAT']

print(HouseData.head())

# training set size is 70% and test size is 30%
X_train,X_test,y_train,y_test = train_test_split(X[features],y,test_size=0.3, random_state=2)

print("\n The size after splitting of the Data into Training and Test Set ",'-'*80)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

In [ ]:

```
# Step5 : Model Selection

#implemneting the random forest manually , by 3 decision trees and then combinning their predictions
#

clf = DecisionTreeClassifier(random_state=1, min_samples_leaf=3)
clf.fit(X_train, y_train)

clf2 = DecisionTreeClassifier(random_state=1, max_depth=4)
clf2.fit(X_train, y_train)

clf3 = DecisionTreeClassifier(random_state=1, max_features=5)
clf3.fit(X_train, y_train)

print(" The accuracy of each Decision Tree for Random Forest build manually is :\n",'-'*80)

predictions1 = clf.predict(X_test)
print(round(roc_auc_score(y_test, predictions1),3))

predictions2 = clf2.predict(X_test)
print(round(roc_auc_score(y_test, predictions2),3))

predictions3 = clf3.predict(X_test)
print(round(roc_auc_score(y_test, predictions3),3))

# now finding the result when both the trees are combined

predictions = clf.predict_proba(X_test)[:,1]
predictions2 = clf2.predict_proba(X_test)[:,1]
predictions3 = clf3.predict_proba(X_test)[:,1]
combined = (predictions + predictions2 + predictions3) / 3
rounded = np.round(combined,3)

print(" The accuracy of Manually Build Random Forest  is :\n",'-'*80)
print(round(roc_auc_score(y_test, rounded),3))


```

In [ ]:

```
# Step6 : Evaluation of the Model with different Random Forest Models

rfc1 = RandomForestClassifier(max_features=13, random_state=1)
rfc1.fit(X_train, y_train)
pred1 = rfc1.predict(X_test)

#y_test = y_test.ravel()
print(y_test.shape)
print(pred1.shape)


print(" The accuracy of  Random Forest with max_features=13 :\n",'-'*80)
print(round(roc_auc_score(y_test, pred1),3))

# play around with the setting for max_features
rfc2 = RandomForestClassifier(max_features=5, random_state=1)
rfc2.fit(X_train, y_train)
pred2 = rfc2.predict(X_test)

print(" The accuracy of  Random Forest with max_features=5 :\n",'-'*80)
print(round(roc_auc_score(y_test, pred2),3))

# play around with the setting for max_features
print(" The accuracy of  Random Forest with max_features=auto :\n",'-'*80)
rfc2 = RandomForestClassifier(max_features="auto", bootstrap=True,random_state=1)
rfc2.fit(X_train, y_train)
pred2 = rfc2.predict(X_test)
print(round(roc_auc_score(y_test, pred2),3))
```