```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec 22 23:21:20 2018

@author: SRIKANT
"""
#import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


#import data and split it into feature and label
houseData = pd.read_csv('housing.csv')

print(houseData.iloc[0:6,:].values)
X = houseData.iloc[:,:-1].values
y = houseData.iloc[:,9].values

#handel missing values
from sklearn.preprocessing import Imputer
missingValues = Imputer(missing_values="NaN",strategy="mean",axis=0)

X[:,0:8] = missingValues.fit_transform(X[:,0:8])

#dealing with categorical data
from sklearn.preprocessing import LabelEncoder
X_labelencoder = LabelEncoder()

X[:,8] = X_labelencoder.fit_transform(X[:,8])

X_labelencoder.classes_

from sklearn.preprocessing import OneHotEncoder
X_ohe = OneHotEncoder( categorical_features=[8])
X = X_ohe.fit_transform(X).toarray()

#split feature and label into training and testing data
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)

#Feature scaling
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.fit_transform(X_test)

#Linear Regression
from sklearn.linear_model import LinearRegression
linearRegressor = LinearRegression()
linearRegressor.fit(X_train,y_train)

#predict linear regression
LRpredict = linearRegressor.predict(X_test)

#LR trainging score
linearRegressor.score(X_train,y_train)

#LR testing score
linearRegressor.score(X_test,y_test)
```

```
#LR Root Mean Square Error
from sklearn.metrics import mean_squared_error
from math import sqrt
LRrms = sqrt(mean_squared_error(y_test,LRpredict))

#Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
DTregressor = DecisionTreeRegressor(random_state=4)
DTregressor.fit(X_train,y_train)


DTpredict = DTregressor.predict(X_test)
#DT trainging score
DTregressor.score(X_train,y_train)
#DT testing score
DTregressor.score(X_test,y_test)

#DT Root Mean Square Error
from sklearn.metrics import mean_squared_error
from math import sqrt
DTrms = sqrt(mean_squared_error(y_test,DTpredict))

#Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
RFregressor = RandomForestRegressor(n_estimators=20,random_state=2)
RFregressor.fit(X_train,y_train)

RFpredict = RFregressor.predict(X_test)

#RF trainging score
RFregressor.score(X_train,y_train)
#RF testing score
RFregressor.score(X_test,y_test)

#RF Root Mean Square Error
from sklearn.metrics import mean_squared_error
from math import sqrt
RFrms = sqrt(mean_squared_error(y_test,RFpredict))


#Extracting just the median_income column from the independent variable
X = np.delete(X,[0,1,2,3,4,5,6,7,8,9,10,11],axis=1)

#split feature and label into training and testing data
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=1/4,random_state=0)

X_train.shape

#Standardize training and test datasets.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

#LInear Regresson
from sklearn.linear_model import LinearRegression
linearRegressor1 = LinearRegression()
linearRegressor1.fit(X_train,y_train)

linearRegressor1.predict(X_test)
```

```
#LR trainging score
linearRegressor1.score(X_train,y_train)
#LR testing score
linearRegressor1.score(X_test,y_test)

#Visualizing regression result  of training data
plt.scatter(X_train,y_train,color='green')
plt.plot(X_train,linearRegressor1.predict(X_train),color='red')
plt.title('Median House Price Prediction')
plt.xlabel('Median income')
plt.ylabel('House price')
plt.show()

#Visualizing regression result  of testing data
plt.scatter(X_test,y_test,color='blue')
plt.plot(X_train,linearRegressor1.predict(X_train),color='red')
plt.title('Median House Price Prediction')
plt.xlabel('Median income')
plt.ylabel('House price')
plt.show()
```