```
import numpy as np
import pandas as pd
import sklearn


from sklearn.datasets import load_boston
df=load_boston()
df.keys()

      dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])


print(df.feature_names)


print(df.target)


print(df.filename)


print(df.data)


boston=pd.DataFrame(df.data,columns=df.feature_names)
boston.head(10)


boston['MEDV']=df.target
boston.head(10)


boston.isnull().head(10)


boston.isnull().sum()


from sklearn.model_selection import train_test_split
X=boston.drop('MEDV',axis=1)
Y=boston['MEDV']

X_train,X_test,Y_train,Y_test = train_test_split(X ,Y, test_size=0.15 ,random_state=5)

print(X_train.shape)
print(X_test.shape)
print(Y_test.shape)
print(Y_train.shape)


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error


lin_model = LinearRegression()
lin_model.fit(X_train,Y_train)


y_train_predict= lin_model.predict(X_train)
```

```
rmse =np.sqrt(mean_squared_error(Y_train,y_train_predict))

print("the model performance of the training set ")
print("RMSE is {}".format(rmse))
print('\n')

y_test_predict= lin_model.predict(X_test)
rmse =np.sqrt(mean_squared_error(Y_test,y_test_predict))

print("the model performance of the testing set ")
print("RMSE is {}".format(rmse))
```