```
# Boston Housing with Linear Regression
# With this data our objective is create a
model using linear regression to predict the
houses price
#
# The data contains the following columns:
#
# 'crim': per capita crime rate by town.
# 'zn': proportion of residential land zoned
for lots over 25,000 sq.ft.
# 'indus': proportion of non-retail business
acres per town.
# 'chas':Charles River dummy variable (= 1 if
tract bounds river; 0 otherwise).
# 'nox': nitrogen oxides concentration (parts
per 10 million).
# 'rm': average number of rooms per dwelling.
# 'age': proportion of owner-occupied units
built prior to 1940.
# 'dis': weighted mean of distances to five
Boston employment centres.
# 'rad': index of accessibility to radial
highways.
# 'tax': full-value property-tax rate per
$10,000.
# 'ptratio': pupil-teacher ratio by town
# 'black': 1000(Bk - 0.63)^2 where Bk is the
proportion of blacks by town.
# 'lstat': lower status of the population
(percent).
# 'medv': median value of owner-occupied homes
in $$1000s

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import
```

```
train_test_split
from sklearn.linear_model import
LinearRegression,Lasso,Ridge

# pd.set_option('display.max_columns',1000)
# pd.set_option('display.width',1000)
# np.random.seed(2)
# Importing DataSet and take a look at Data

BostonTrain = pd.read_csv("boston_train.csv")
# ID columns does not relevant for our
analysis.
BostonTrain.drop('ID', axis = 1, inplace=True)

X = BostonTrain[['crim', 'zn', 'indus', 'chas',
'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat']]
y = BostonTrain['medv']


X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.4)

model = LinearRegression()
model.fit(X_train,y_train)

predictions = model.predict(X_test)

print(predictions)

plt.scatter(y_test, predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()

from sklearn import metrics
print('simple Linear Regression')
print('MAE:',
metrics.mean_absolute_error(y_test,
predictions))
print('MSE:',
metrics.mean_squared_error(y_test,
predictions))
print('RMSE:',
np.sqrt(metrics.mean_squared_error(y_test,
predictions)))
```