Session 20 – Assignment 1

**Question:**

Build the linear regression model using scikit learn in boston data to predict 'Price'

based on other dependent variable.

**Code:**

#Importing required Python packages

import numpy as np

import pandas as pd

import scipy.stats as stats

import matplotlib.pylab as plt

from scipy import sparse

import sklearn

import seaborn

from pandas.plotting import scatter\_matrix

from sklearn.decomposition import PCA

plt.rcParams['figure.figsize'] = (15, 5)

# Set some Pandas options

pd.set\_option('display.notebook\_repr\_html', False)

pd.set\_option('display.max\_columns', 40)

pd.set\_option('display.max\_rows', 25)

pd.options.display.max\_colwidth = 50

from sklearn.datasets import load\_boston

boston = load\_boston()

display(boston)

bos = pd.DataFrame(boston.data)

display(bos)

print(boston.keys())

print(boston.data.shape)

print(boston.target.shape)

print(boston.feature\_names)

print(boston.DESCR)

bos = pd.DataFrame(boston.data)

bos.head()

bos.columns = boston.feature\_names

bos.head()

boston.target[:5]

bos['PRICE'] = boston.target

display(bos['PRICE'])

df = pd.concat([pd.DataFrame(boston.data, columns=boston.feature\_names), pd.DataFrame(boston.target, columns=['MEDV'])], axis=1)

df.describe()

#Let's do a simple histogram on available house prices to see what we'll be working on

plt.hist(boston.target,bins=20)

plt.suptitle('Boston Housing Prices in $1000s', fontsize=15)

plt.xlabel('Prices in $1000s')

plt.ylabel('Count')

plt.show()

from sklearn.linear\_model import LinearRegression

X = bos.drop('PRICE', axis = 1)

lm = LinearRegression()

lm

plt.scatter(bos.RM, bos.PRICE)

plt.xlabel('Average number of Rooms RM')

plt.ylabel('Housing Price')

plt.title('Relationship between RM and PRice')

plt.show()

lm.fit(X, bos.PRICE)

lm.predict(X)[0:5]

plt.scatter(bos.PRICE, lm.predict(X))

plt.xlabel('Prices: $Y\_i$')

plt.ylabel('Predicted Prices: $\hat{Y}\_i$')

plt.title('Prices vs Predicted Prices: $Y-i$ vs $\hat{Y}\_i$')

plt.show()

mseFull = np.mean((bos.PRICE - lm.predict(X)) \*\* 2)

print(mseFull)

lm = LinearRegression()

lm.fit(X[['PTRATIO']], bos.PRICE)

msePTRATIO = np.mean((bos.PRICE - lm.predict(X[['PTRATIO']])) \*\* 2)

print(msePTRATIO)

X\_train = X[:-50]

X\_test = X[-50:]

Y\_train = bos.PRICE[:-50]

Y\_test = bos.PRICE[-50:]

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = sklearn.model\_selection.train\_test\_split(X, bos.PRICE, test\_size=0.33, random\_state = 5)

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

lm = LinearRegression()

lm.fit(X\_train, Y\_train)

pred\_train = lm.predict(X\_train)

pred\_test = lm.predict(X\_test)

print ("Fit a model X\_train, and calculate MSE with Y\_train:", np.mean((Y\_train - lm.predict(X\_train)) \*\* 2))

print ("Fit a model X\_train, and calculate MSE with X\_test, Y\_test:", np.mean((Y\_test - lm.predict(X\_test)) \*\* 2))

plt.scatter(lm.predict(X\_train), lm.predict(X\_train) - Y\_train, c="b", s=40, alpha=0.5)

plt.scatter(lm.predict(X\_test), lm.predict(X\_test) - Y\_test, c="g", s=40)

plt.hlines(y=0, xmin=0, xmax=50)

plt.title('Residual Plot using training (blue) and test (green) data')

plt.ylabel('Residuals')

with seaborn.axes\_style('white'):

smaller\_frame = df[['CRIM', 'RM', 'DIS', 'RAD', 'TAX','MEDV']]

scatter\_matrix(smaller\_frame, alpha=0.8, figsize=(12, 12), diagonal="kde")

plt.show()

df[['MEDV','RM','CRIM']].corr()

data\_reduced=PCA(n\_components=1).fit\_transform(boston.data)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data\_reduced, boston.target)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

linr=LinearRegression().fit(X\_train, y\_train)

y\_pred = linr.predict(X\_test)

print ("R-squared for train: %.2f" %linr.score(X\_train, y\_train))

print ("R-squared for test: %.2f" %linr.score(X\_test, y\_test))

print ("Coefficients (Parameters theta\_1..theta\_n")

print (linr.coef\_)

print ("Y intercept (theta\_0): %.2f" %linr.intercept\_)

with seaborn.axes\_style('white'):

plt.scatter(data\_reduced, boston.target, c='r')

plt.plot(X\_test, y\_pred,'--k', c='b')

plt.show()

Output:

























