RandPython

Tinniam V Ganesh

October 3, 2017

trainTestSplit <- function(df,trainPercent,seed1){  
 ## Sample size percent  
 samp\_size <- floor(trainPercent/100 \* nrow(df))  
   
 ## set the seed to make your partition reproductible  
 ## set the seed to make your partition reproductible  
 set.seed(seed1)  
 idx <- sample(seq\_len(nrow(df)), size = samp\_size)  
 idx  
  
}  
  
Rsquared <- function(lmfit,newdf,y){  
 yhat <- predict(lmfit,newdata=newdf)  
 RSS <- sum((y - yhat)^2)  
 TSS <- sum((y - mean(y))^2)  
 rsquared <-1 - (RSS/TSS)  
 rsquared  
}  
  
  
df=read.csv("Boston.csv",stringsAsFactors = FALSE) # Data from MASS - SL  
train\_idx <- trainTestSplit(df,trainPercent=75,seed=5)  
train <- df[train\_idx, ]  
test <- df[-train\_idx, ]  
  
  
fit=lm(medv~lstat,data=df)  
fit

##   
## Call:  
## lm(formula = medv ~ lstat, data = df)  
##   
## Coefficients:  
## (Intercept) lstat   
## 34.55 -0.95

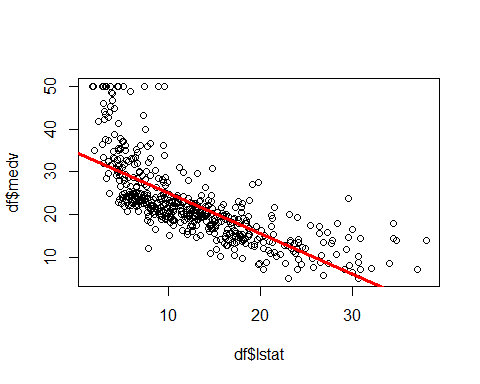
summary(fit)

##   
## Call:  
## lm(formula = medv ~ lstat, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -15.168 -3.990 -1.318 2.034 24.500   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 34.55384 0.56263 61.41 <2e-16 \*\*\*  
## lstat -0.95005 0.03873 -24.53 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 6.216 on 504 degrees of freedom  
## Multiple R-squared: 0.5441, Adjusted R-squared: 0.5432   
## F-statistic: 601.6 on 1 and 504 DF, p-value: < 2.2e-16

confint(fit)

## 2.5 % 97.5 %  
## (Intercept) 33.448457 35.6592247  
## lstat -1.026148 -0.8739505

plot(df$lstat,df$medv)  
abline(fit)  
abline(fit,lwd=3)  
abline(fit,lwd=3,col="red")



import numpy as np  
import pandas as pd  
import os  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
os.chdir("C:\\software\\machine-learning\\RandPython")  
df = pd.read\_csv("Boston.csv",encoding = "ISO-8859-1")  
X=df['lstat']  
y=df['medv']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,random\_state = 0)  
X\_train=X\_train.values.reshape(-1,1)  
X\_test=X\_test.values.reshape(-1,1)  
from sklearn.linear\_model import LinearRegression  
  
linreg = LinearRegression().fit(X\_train, y\_train)  
print('linear model coeff (w): {}'  
 .format(linreg.coef\_))  
print('linear model intercept (b): {:.3f}'  
 .format(linreg.intercept\_))  
print('R-squared score (training): {:.3f}'  
 .format(linreg.score(X\_train, y\_train)))  
print('R-squared score (test): {:.3f}'  
 .format(linreg.score(X\_test, y\_test)))  
   
  
fig=plt.scatter(X\_train,y\_train)  
plt.plot()  
# Create a range of points. Compute yhat=coeff1\*x + intercept and plot  
x=np.linspace(0,40,20)  
plt.plot(x, linreg.coef\_ \* x + linreg.intercept\_, color='red')  
#plt.show()  
fig.figure.savefig('foo.png', bbox\_inches='tight')  
print "finished"

## linear model coeff (w): [-0.97063097]  
## linear model intercept (b): 34.787  
## R-squared score (training): 0.571  
## R-squared score (test): 0.458  
## finished

Output image: 