Relatório Atividade 2 – Regressão Linear

Nome: Wallesson Cavalcante da Silva

Curso: Ciência da Computação

Matrícula: 397670

Considerações:

Nesse relatório busco descrever somente o que é novidade de implementação. Poderia ter criado uma função para fazer o plot dos gráficos e deixar o código mais elegante, porém ainda tenho uma certa dificuldade com a linguagem e me preocupei mais com resolver os problemas em si.

Passos do algoritmo:

Primeiramente carreguei os dados e utilizei o numpy para embaralhar os dados. Depois armazeno em uma variável 'porc' o valor de uma conta para delimitar um inteiro que corresponda 80% dos dados, utilizo a função 'round' para retornar um inteiro da conta. Logo após, separo os dados entre '_Treino e Teste

```
data = np.loadtxt("housing.data")
np.random.shuffle(data)

porc = round(data.shape[0]*0.8)

LSTAT_Treino = data[0:porc, -2]
MEDV_Treino = data[0:porc, -1]

LSTAT_Teste = data[porc:, -2]
MEDV_Teste = data[porc:, -1]
```

Logo abaixo segue a implementação das funções pedidas para MSE e R2, que são F_MSE e F_R2 que retornam os valores respectivamente.

```
def F_MSE (self,Y,MEDV_pred):
    Sub_ = Y-MEDV_pred
    Mult= Sub_ * Sub_
    SOMA = sum(Mult)
    MSE_ = SOMA.mean()
    return MSE_

def F_R2(self,Y,teste_Pred):
    return 1-(self.RSS(Y, teste_Pred)/self.TSS(Y, teste_Pred))
```

Por fim, faço o treinamento e a predição dos dados coletados. Chamo as funções que me retornam MSE e R2 e printo na tela. Depois utilizo a função 'scatter' para marcar os pontos que foram uados para teste de LSTAT, no eixo X, e os que foram utilizados como teste de

MSVD no eixo Y. Logo após, utilizo os valores LSTAT de teste e os valores preditos de MEDV para gerar a reta de regressão linear. Assim tenho o primeiro gráfico solicitado.

Para gerar o segundo gráfico. Utilizo os valores preditos de MEDV no eixo X e os valores de teste que usei do MEDV para demarcar os pontos. Para mostrar a reta da regressão utilizo os valores de teste de MEDV nos dois pontos.

```
reg = SimpleLinearRegression()
reg.fit(LSTAT_Treino, MEDV_Treino)
MEDV_pred = reg.predict(LSTAT_Teste)

f = Fun()

MSE_Teste = f.F_MSE(MEDV_Teste,MEDV_pred)
R_2_Teste = f.F_R2(MEDV_Teste,MEDV_pred)

print("MSE Conjunto de Teste: ",MSE_Teste)
print("R2 Conjunto de Teste: ",R_2_Teste)

plt.scatter(LSTAT_Teste,MEDV_Teste)
plt.plot(LSTAT_Teste,MEDV_pred,c='r')
plt.show()

plt.scatter(MEDV_pred,MEDV_Teste)
plt.plot(MEDV_Teste,MEDV_Teste, c='r')
plt.show()
```

Obs. Para LSTAT2 e LSTAT3 usa a mesma ideia, diferença é que utilizei multiplicação vetorial nos dados de teste e de treino de ambos os casos. No LSTAT2 fiz os dados L_2_ = LSTAT_*LSTAT_ e M_2_ = MEDV_ * MEDV_. O mesmo vale para o LSTAT3 que faz as multiplicações serem repetidas mais uma vez.

Realizei essas operações com base no que entendi que as questões pediam.

Resultados:

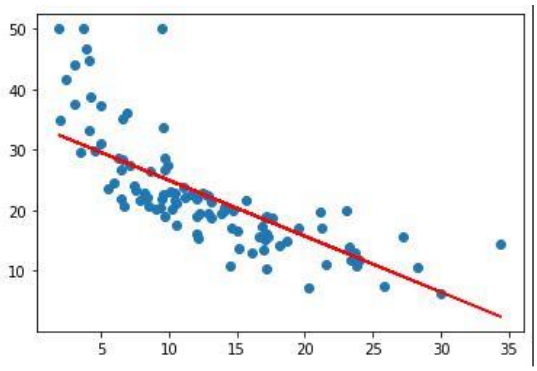
a) **MSE e R2:**

```
MSE Conjunto de Teste:  3772.884467026932
R2 Conjunto de Teste:  0.58159270182191

MSE_2 Conjunto de Teste:  21397767.683569815
R2_2 Conjunto de Teste:  0.2713586925093263

MSE_3 Conjunto de Teste:  64292391015.77599
R2_3 Conjunto de Teste:  0.10338787504479496
```

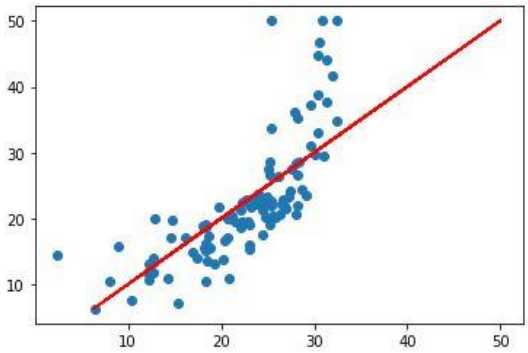
Gráficos:

LSTAT e MEDV normais

b)

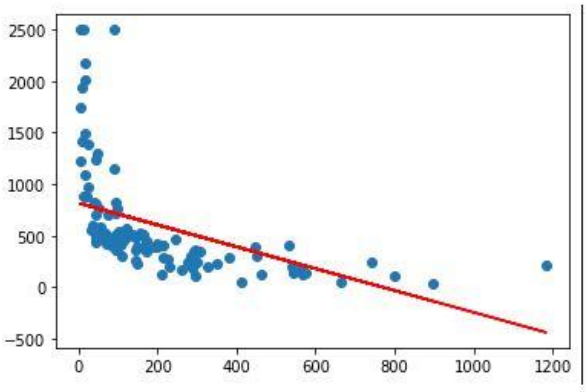


c)

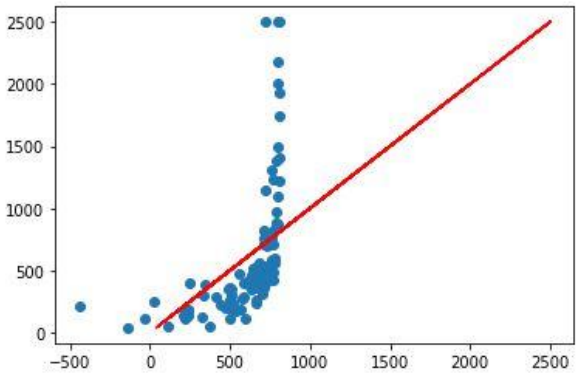


LSTAT2 e MEDV2

b)

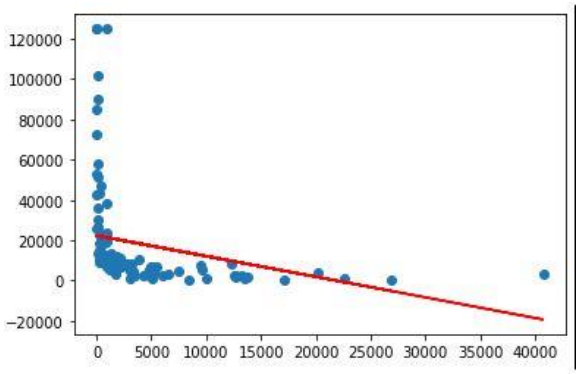


c)



LSTAT3 e MEDV3

b)



c)