Chengjun_Liu_7116099552_hw1

February 12, 2020

1. Did you go over Lectures 1, 2a, 2b, Python tutorials 1 and 2, Weekly exercises 1 and 2, and Lecture 2aEx before writing this homework? If you haven't, please do so first. This homework, like every other homework, only covers a small part of our covered materials. How many hours outside class time did you spend on DSO 530 each week on average so far? (an honest answer will help the instructor adjust the pace if necessary).

Answer:

Yes, I have looked through the lectures, Python tutorials and exercises. I spend 5 hours on average every week.

2. For a regression equation log y = 1 + 50 log x, how does changes in y associated with changes in x? (You don't have to type latex math formulas; as long as your derivation is readable by the grader, it is fine; you might need Python to help get the answer).

Answer:

Assume there is an increase of 1% in independent variable x (that is, $(1+1\%)x$), Let $logy = 1 + 50logx$:

$$\begin{aligned} 1 + 50log((1+1\%)x) =& 1 + 50(log(x) + log(1.01)) \\ =& 1 + 50log(x) + 50 \times log(1.01) \\ =& log(y) + 50 \times log(1.01) \\ =& log(y) + log(1.01^{50}) \\ =& log(y \times 1.01^{50}) \end{aligned}$$

Because $1.01^{50} = e^{log(1.01^{50})} = e^{50 \times log(1.01)} \approx e^{50 \times (1.01-1)} = e^{0.5} \approx 0.5 + 1 = 1.5$:

$$log(y \times 1.01^{50}) \approx log(1.5 \times y)$$

We have $log((1+50\%) \times y) \approx 1 + 50log((1+1\%)x)$, approximately 50% increase on y.

Alternatively:

$$\begin{aligned} e^{logy} =& y \\ =& e^{1+50logx} \\ =& e \times e^{50logx} \\ =& e \times x^{50} \end{aligned}$$

$$\begin{aligned}e^{1+50log((1+1\%)x)} =& e \times e^{50log((1+1\%)x)}\\ =& e \times (1.01x)^{50}\\ =& e \times x^{50} \times 1.01^{50}\\ =& y \times 1.01^{50}\end{aligned}$$

They reach the same result: x increases 1%, y increases \$ 1.01^{50} -1\$, or approximately 50%.

Inversely, 1% increase in y means $1\%/50 = 0.02\%$ increase in x.

3. Recall that in Lecture 2b, we regress medv on PTRATIO and RM using the Boston housing data. Repeat this regression, but normalize both features before running the regression. Compared with what you see in the Lecture, do you get a different R2 , or do you get the same one?

[1]:
```
from sklearn.datasets import load_boston
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
```

[2]:
```
Boston=load_boston()
```

[3]:
```
Boston_Data=pd.DataFrame(Boston.data,columns=Boston.feature_names)
Boston_Data.head()
```

[3]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT
0     15.3  396.90   4.98
1     17.8  396.90   9.14
2     17.8  392.83   4.03
3     18.7  394.63   2.94
4     18.7  396.90   5.33
```

[4]:
```
X=Boston_Data[["PTRATIO","RM"]].values
y=Boston.target
```

[5]:
```
# We scale both features using normalization of min-max
mm=MinMaxScaler()
X=mm.fit_transform(X)
```

[6]:
```
lr=LinearRegression()
lr.fit(X,y)
```

```
print("R square after normalization:",lr.score(X,y))
```

```
R square after normalization: 0.5612534621272918
```

The R Square of regression on normalized features is the same as original regression, proving normalization doesn't affect r-square because scaling doesn't influence the correlation between features and dependent variable.

4. Split the Boston housing data into two parts with 30% as test data. Use random_state = 1 in this split. Because this is a regression problem, you don't want to use stratify = y part of the code from our Python tutorial. Regress medv on LSTAT and RM using the training data. Compute R2 on both the training data and the test data.

[7]:
```
from sklearn.model_selection import train_test_split
```

[8]:
```
X=Boston_Data[["LSTAT","RM"]].values
y=Boston.target

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1)
```

[9]:
```
lr=LinearRegression()
lr.fit(X_train,y_train);
```

[10]:
```
print("R Square for training set:",lr.score(X_train,y_train))
print("R Square for test set:",lr.score(X_test,y_test))
```

```
R Square for training set: 0.6099162694401526
R Square for test set: 0.6843090583339466
```

5. Use the first 30 rows of the training set you got in problem 4 as the new training set, regress medv on LSTAT and RM again. How much is the R2 on this training set? How much is R2 if you evaluate this new regression line's performance using the test data in problem 4?

[11]:
```
new_X=X[:30,:]
new_y=y[:30]
```

[12]:
```
lr=LinearRegression()
lr.fit(new_X,new_y)

print("R Square of selected 30-row training set:",lr.score(new_X,new_y))
print("R Square of test data on regression of trained 30-row training set:",lr.
 ↪score(X_test,y_test))
```

```
R Square of selected 30-row training set: 0.6911690180235888
R Square of test data on regression of trained 30-row training set:
0.7085118171135107
```

[ ]:
```

```