Falak_Jain_HW1

February 10, 2022

HW1 - Falak Jain

[16]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
```

1.

(i) Yes

(ii) 4 hours per week on average

(iii) Yes

2.

Model: log y = 1 + 70 log x

- This implies that for a 1% increase in x, there is a roughly 101% increase in y
- If x changes: the regression equation is 1 + 70 log(x_new)
- Old equation : log(y_old) = 1 + 70log(x_old)

Interpretation:

- log(y_new) = 1 + 70log(x_new)
- Subtracting the old and new equations we get,
- log(y_new) - log(y_old) = 70log(x_new) - 70log(x_old)
- log(y_new/y_old) = 70log(x_new/x_old)
- y_new/y_old = (x_new/x_old)^70
- Therefore, if there is a 1% increase in x,
- y_new/y_old = 1.01^70 = 2.01

Therefore there is an increase of 101% in y

3.

```
[4]: housing = pd.read_csv('housing.csv')
     housing.head()
```

```
[4]:       crim    zn  river     rm  ptratio  medv
     0  0.00632  18.0      0  6.575     15.3  24.0
     1  0.02731   0.0      0  6.421     17.8  21.6
     2  0.02729   0.0      0  7.185     17.8  34.7
     3  0.03237   0.0      0  6.998     18.7  33.4
     4  0.06905   0.0      0  7.147     18.7  36.2
```

```
[3]: mms = MinMaxScaler()
     X = mms.fit_transform(X)
```

Before Min Max Scaling

```
[5]: X = housing[['ptratio','rm']].values
     y = housing['medv'].values
```

```
[11]: linear_model = LinearRegression()
      linear_model.fit(X,y)
      r_sq = linear_model.score(X,y)
      print('Coefficient of determination: ', r_sq)
```

```
Coefficient of determination:  0.5612534621272917
```

After Min Max Scaling

```
[12]: X = housing[['ptratio','rm']].values.astype(float)
      y = housing['medv'].values
      mms = MinMaxScaler()
      X_scaled = mms.fit_transform(X)
      X_norm = pd.DataFrame(X_scaled,columns = ['ptratio','rm'])
      X_norm
```

```
[12]:       ptratio        rm
      0    0.287234  0.577505
      1    0.553191  0.547998
      2    0.553191  0.694386
      3    0.648936  0.658555
      4    0.648936  0.687105
      ..        ...       ...
      501  0.893617  0.580954
      502  0.893617  0.490324
      503  0.893617  0.654340
      504  0.893617  0.619467
      505  0.893617  0.473079

      [506 rows x 2 columns]
```

[14]:
```python
linear_model = LinearRegression()
X_norm_values = X_norm[['ptratio','rm']].values
linear_model.fit(X_norm_values,y)
r_sq = linear_model.score(X_norm_values,y)
print('Coefficient of determination: ', r_sq)
```

```
Coefficient of determination:  0.5612534621272917
```

We get the same R-sq value as Lecture 2b

4.

[5]:
```python
import statsmodels.api as sm
X = sm.add_constant(housing[['zn','river','rm','ptratio','medv']].values)
y = housing['crim'].values
ols = sm.OLS(y,X)
ols_result = ols.fit()
ols_result.summary()
```

[5]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.170
Model:                            OLS   Adj. R-squared:                  0.161
Method:                 Least Squares   F-statistic:                     20.43
Date:                Sun, 30 Jan 2022   Prob (F-statistic):           1.41e-18
Time:                        19:13:20   Log-Likelihood:                -1759.3
No. Observations:                 506   AIC:                             3531.
Df Residuals:                     500   BIC:                             3556.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -4.3579      5.449     -0.800      0.424     -15.064       6.348
x1            -0.0178      0.017     -1.054      0.293      -0.051       0.015
x2             0.4869      1.415      0.344      0.731      -2.294       3.268
x3             1.2903      0.698      1.850      0.065      -0.080       2.661
x4             0.4466      0.195      2.288      0.023       0.063       0.830
x5            -0.3644      0.058     -6.237      0.000      -0.479      -0.250
==============================================================================
Omnibus:                      557.602   Durbin-Watson:                   1.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            32052.496
Skew:                           5.097   Prob(JB):                         0.00
Kurtosis:                      40.635   Cond. No.                         544.
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

- R-sq = 0.17, adj-R-sq = 0.161
- We can reject the null for ptratio and medv predictors

[6]:
```python
import statsmodels.api as sm
X = sm.add_constant(housing[['ptratio','medv']].values)
y = housing['crim'].values
ols = sm.OLS(y,X)
ols_result = ols.fit()
ols_result.summary()
```

[6]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.162
Model:                            OLS   Adj. R-squared:                  0.159
Method:                 Least Squares   F-statistic:                     48.75
Date:                Sun, 30 Jan 2022   Prob (F-statistic):           4.43e-20
Time:                        19:13:20   Log-Likelihood:                -1761.5
No. Observations:                 506   AIC:                             3529.
Df Residuals:                     503   BIC:                             3542.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.2931      4.087      0.316      0.752      -6.737       9.323
x1             0.4966      0.188      2.639      0.009       0.127       0.866
x2            -0.3038      0.044     -6.857      0.000      -0.391      -0.217
==============================================================================
Omnibus:                      562.630   Durbin-Watson:                   0.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            34204.153
Skew:                           5.151   Prob(JB):                         0.00
Kurtosis:                      41.939   Cond. No.                         349.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Interpretation of slopes:

- For every one unit increase in pupil teacher ratio, the crime rate increases by 0.4966 crimes per capita

- For every one thousand dollar increase in the median home price in a neighborhood, the crime rate reduces by 0.3038 crimes per capita

5.

[7]:
```
from sklearn.model_selection import train_test_split
X,y = housing[['river','rm']].values,housing.iloc[:,-1].values
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.
↪3,random_state = 2)
```

[8]:
```
linear_model = LinearRegression()
linear_model.fit(X_train,y_train)
r_sq = linear_model.score(X_train,y_train)
print('In-Sample Coefficient of determination: ', r_sq)
r_sq = linear_model.score(X_test,y_test)
print('Out-of-Sample Coefficient of determination: ', r_sq)
```

```
In-Sample Coefficient of determination:  0.4652498065943518
Out-of-Sample Coefficient of determination:  0.5582472793500367
```