Interactive exercise week #9a

Liping Wu 300-958-061 11/9/2020

In this exercise we will do the following:

- Build a linear regression model using:
  - The ols method and the statsmodel.formula.api library
  - The scikit-learn package

Pre-requisites:

1- Install Anoconda
2- We will be using a lot of Public datasets these datasets are available at https://goo.gl/zjS4C6 under a folder named "Datasets for Predictive Modelling with Python", the datasets are organized in the order of the text book chapters: Python: Advanced Predictive Analytics, chapter # 5 files are required

Steps for building a linear regression model:

1- Open your spider IDE
2- Load the 'Adertising.csv' file into a dataframe name *the dataframe data_firstname_adv where first name is your first name* carry out the following activities:
   a. Display the column names
   b. Display the shape of the data frame i.e number of rows and number of columns
   c. Display the main statistics of the data
   d. Display the types of columns
   e. Display the first five records

   Following is the code, *make sure you update the path to the correct path where you placed the files and update the data frame name correctly*:

```
import pandas as pd
import os
path = "D:/CentennialWu/2020Fall/COMP309Data/Assignments/Lab07Wk09/"
filename = 'Advertising.csv'
fullpath = os.path.join(path,filename)
print(fullpath)
data_liping_adv = pd.read_csv(fullpath)
data_liping_adv.columns.values
data_liping_adv.shape
```

data_liping_adv.describe()
data_liping_adv.dtypes
data_liping_adv.head(5)

```
In [17]: data_liping_adv.columns.values
Out[17]: array(['TV', 'Radio', 'Newspaper', 'Sales'], dtype=object)

In [18]: data_liping_adv.shape
Out[18]: (200, 4)

In [19]: data_liping_adv.describe()
Out[19]:
               TV       Radio   Newspaper       Sales
count  200.000000  200.000000  200.000000  200.000000
mean   147.042500   23.264000   30.554000   14.022500
std     85.854236   14.846809   21.778621    5.217457
min      0.700000    0.000000    0.300000    1.600000
25%     74.375000    9.975000   12.750000   10.375000
50%    149.750000   22.900000   25.750000   12.900000
75%    218.825000   36.525000   45.100000   17.400000
max    296.400000   49.600000  114.000000   27.000000

In [20]: data_liping_adv.dtypes
Out[20]:
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object

In [21]: data_liping_adv.head(5)
Out[21]:
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3    9.3
3  151.5   41.3       58.5   18.5
4  180.8   10.8       58.4   12.9

In [22]:
```

3- Let us check if there is a correlation between advertisement costs on TV and the resultant sales. Remember the formula:

$$correlation\,coefficient(h)=\frac{\Sigma\left((x-xm)*(y-ym)\right)}{\sqrt{\Sigma(x-xm)^{2*}\,\Sigma(y-ym)^{2}}}$$

a. Use the numpy package to build a function to calculate the correlation between each input variable TV,Radio & Newspaper and the output Sales
b. Run the below code snippet , you should get a result the following results:
   0.782224424861606
   0.5762225745710553
   0.22829902637616525

Following is the code, *make sure you update the path to the correct path where you placed the files and the dataframe name.*

```
import numpy as np
def corrcoeff(df,var1,var2):
   df['corrn']=(df[var1]-np.mean(df[var1]))*(df[var2]-np.mean(df[var2]))
   df['corrd1']=(df[var1]-np.mean(df[var1]))**2
```

```
    df['corrd2']=(df[var2]-np.mean(df[var2]))**2
    corrcoeffn=df.sum()['corrn']
    corrcoeffd1=df.sum()['corrd1']
    corrcoeffd2=df.sum()['corrd2']
    corrcoeffd=np.sqrt(corrcoeffd1*corrcoeffd2)
    corrcoeff=corrcoeffn/corrcoeffd
    return corrcoeff
print(corrcoeff(data_liping_adv,'TV','Sales'))
print(corrcoeff(data_liping_adv,'Radio','Sales'))
print(corrcoeff(data_liping_adv,'Newspaper','Sales'))
```

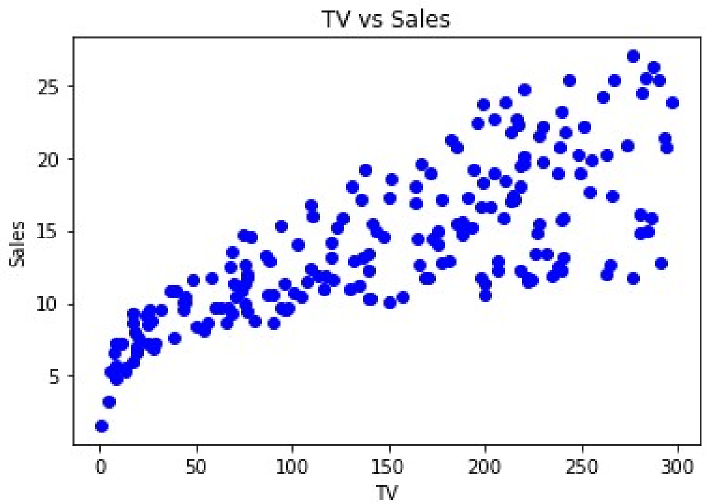
```
In [14]: import numpy as np
    ...: def corrcoeff(df,var1,var2):
    ...:     df['corrn']=(df[var1]-np.mean(df[var1]))*(df[var2]-np.mean(df[var2]))
    ...:     df['corrd1']=(df[var1]-np.mean(df[var1]))**2
    ...:     df['corrd2']=(df[var2]-np.mean(df[var2]))**2
    ...:     corrcoeffn=df.sum()['corrn']
    ...:     corrcoeffd1=df.sum()['corrd1']
    ...:     corrcoeffd2=df.sum()['corrd2']
    ...:     corrcoeffd=np.sqrt(corrcoeffd1*corrcoeffd2)
    ...:     corrcoeff=corrcoeffn/corrcoeffd
    ...:     return corrcoeff

In [15]: print(corrcoeff(data_liping_adv,'TV','Sales'))
    ...: print(corrcoeff(data_liping_adv,'Radio','Sales'))
    ...: print(corrcoeff(data_liping_adv,'Newspaper','Sales'))
0.782224424861606
0.5762225745710553
0.22829902637616525
```

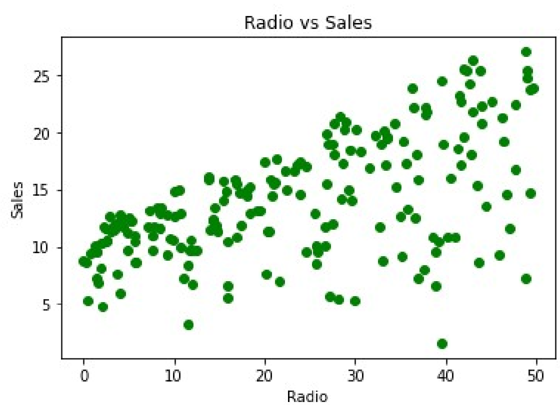
4- Use the matplotlib module to visualize the relationships between each of the inputs and the output (sales), i.e. generate three scattered plots.

Following is the code, *make sure you update the path to the correct path where you placed the files and use the correct dataframe name:*

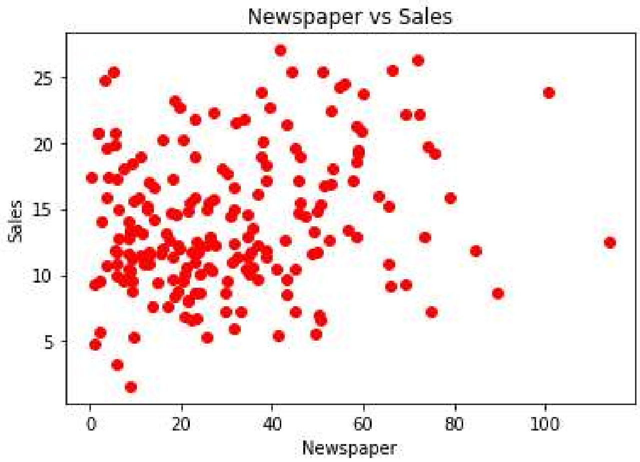
```
import matplotlib.pyplot as plt
plt.plot(data_liping_adv['TV'], data_liping_adv['Sales'],'ro', color='blue')
plt.xlabel("TV")
plt.ylabel("Sales")
plt.title('TV vs Sales')
plt.show()
```




```
plt.plot(data_liping_adv['Radio'], data_liping_adv['Sales'],'ro', color='green')
plt.xlabel("Radio")
plt.ylabel("Sales")
plt.title('Radio vs Sales')
```




```
plt.plot(data_liping_adv['Newspaper'], data_liping_adv['Sales'],'ro', color ='red')
plt.xlabel("Newspaper")
plt.ylabel("Sales")
plt.title('Newspaper vs Sales')
```



4. Use the ols method and the statsmodel.formula.api library to build a linear regression model with TV costs as the predictor (input) and sales as the predicted i.e. estimate the parameters of the model. You should get the following results:

   Intercept   7.032594
   TV   0.047537
   Following is the code, *make sure you update the path to the correct path where you placed the files and use the correct dataframe name:*

   import statsmodels.formula.api as smf
   model1=smf.ols(formula='Sales~TV',data= data_liping_adv).fit()
   model1.params

```
In [47]: model1.params
Out[47]:
Intercept    7.032594
TV           0.047537
dtype: float64
```

   model1=smf.ols(formula='Sales~Radio',data=data_liping_adv).fit()
   model1.params

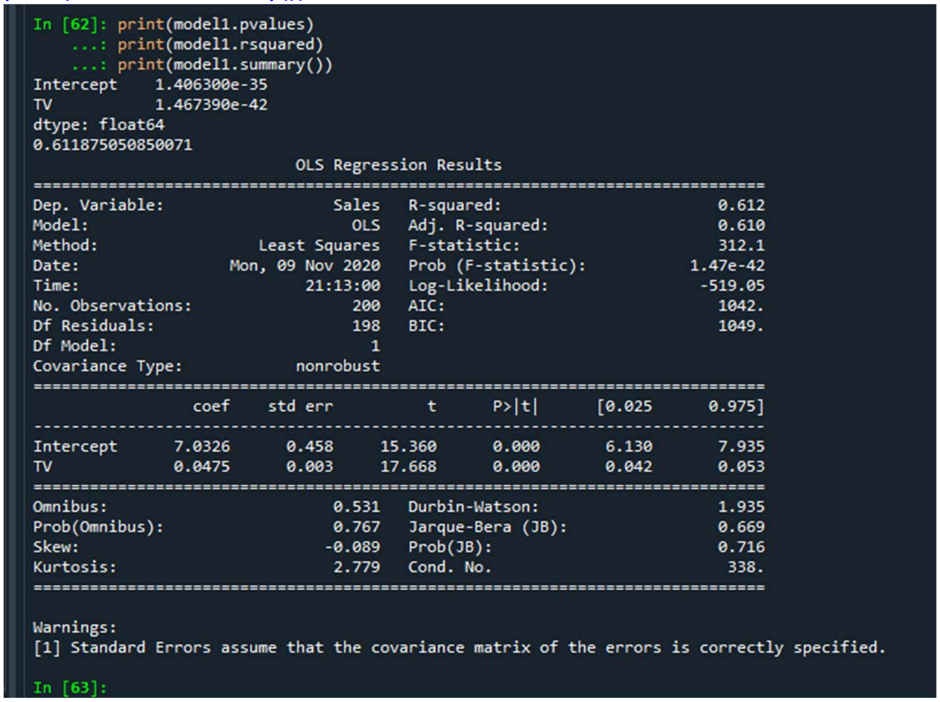
```
In [48]: model1=smf.ols(formula='Sales~Radio',data=data_liping_adv).fit()
    ...: model1.params
Out[48]:
Intercept    9.311638
Radio        0.202496
dtype: float64
```

   model1=smf.ols(formula='Sales~Newspaper',data=data_liping_adv).fit()
   model1.params

```
In [49]: model1=smf.ols(formula='Sales~Newspaper',data=data_liping_adv).fit()
    ...: model1.params
Out[49]:
Intercept    12.351407
Newspaper     0.054693
dtype: float64
```

5- Generate the p-values and the R-squared and model summary, run the following lines of code

```
print(model1.pvalues)
print(model1.rsquared)
print(model1.summary())
```



```
In [62]: print(model1.pvalues)
    ...: print(model1.rsquared)
    ...: print(model1.summary())
Intercept    1.406300e-35
TV           1.467390e-42
dtype: float64
0.611875050850071
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Mon, 09 Nov 2020   Prob (F-statistic):           1.47e-42
Time:                        21:13:00   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================

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

In [63]:
```

6- Re-build the model with two predictors TV and Radio as input variables and print the parameters, p-values, rsquared and summary. Then:

a. Create a new data frame with 2 new values for TV and Radio
b. Predict using the new values
c. Change the values and run the prediction again
d. Change the values again to two values already existing in the dataset and run the prediction again

7- Based on the output our new formula is: $Sales = 2.92 + 0.045 * TV + 0.18 * Radio$

Following is the code, *make sure you update the path to the correct path where you placed the files and use the correct dataframe name:*

```
import statsmodels.formula.api as smf
model3=smf.ols(formula='Sales~TV+Radio',data= data_liping_adv).fit()
print(model3.params)
print(model3.rsquared)
print(model3.summary())
```

```
In [63]: import statsmodels.formula.api as smf
    ...: model3=smf.ols(formula='Sales~TV+Radio',data= data_liping_adv).fit()
    ...: print(model3.params)
    ...: print(model3.rsquared)
    ...: print(model3.summary())
Intercept    2.921100
TV           0.045755
Radio        0.187994
dtype: float64
0.8971942610828956
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     859.6
Date:                Mon, 09 Nov 2020   Prob (F-statistic):           4.83e-98
Time:                        21:21:23   Log-Likelihood:                -386.20
No. Observations:                 200   AIC:                             778.4
Df Residuals:                     197   BIC:                             788.3
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.9211      0.294      9.919      0.000       2.340       3.502
TV             0.0458      0.001     32.909      0.000       0.043       0.048
Radio          0.1880      0.008     23.382      0.000       0.172       0.204
==============================================================================
Omnibus:                       60.022   Durbin-Watson:                   2.081
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              148.679
Skew:                          -1.323   Prob(JB):                     5.19e-33
Kurtosis:                       6.292   Cond. No.                         425.
==============================================================================

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

So: $Sales = 2.92 + 0.045 * TV + 0.18 * Radio$

Predicte a new value

\# give TV=50, Radio =40
X_new2 = pd.DataFrame({'TV': [50],'Radio' : [40]})

\# predict for a new observation
sales_pred2=model3.predict(X_new2)
print(sales_pred2)

```
In [77]:
    ...: X_new2 = pd.DataFrame({'TV': [50], 'Radio' : [40] })
    ...: # predict for a new observation
    ...: sales_pred2=model3.predict(X_new2)
    ...: print(sales_pred2)
0    12.72861
dtype: float64
```

Notice in this exercise we used all the data for training, this is not the best approach, it is better to split the data randomly into test and train.

8- In this step we will build the model using scikit-learn package, this is the more commonly used package to build data science projects. This method is more elegant as it has more in-built methods to perform the regular processes associated with regression. Carry out the following:

a. Import the necessary modules
b. Split the dataset into 80% for training and 20% for testing
c. Print out the parameters
d. Test the model using the Train/Test

Following is the code, *make sure you update the path to the correct path where you placed the files and use the correct dataframe name:*

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

feature_cols = ['TV', 'Radio']
# feature_cols = ['TV', 'Radio', 'Newspaper'] if more column invols
X = data_liping_adv[feature_cols]
Y = data_liping_adv['Sales']
trainX,testX,trainY,testY = train_test_split(X,Y, test_size = 0.20)
trainX.describe()
trainX.columns.values
trainX.shape

trainY.describe()
trainY.shape

testX.describe()
testX.columns.values
testX.shape

testY.describe()
testY.shape

#print test values
for z in zip(testX.index, testY):
    print (z)
```

```
    ...: #print test values
    ...: for z in zip(testX.index, testY):
    ...:     print (z)
(176, 20.2)
(148, 10.9)
(8, 4.8)
(192, 5.9)
(163, 18.0)
(125, 10.6)
(23, 15.5)
(97, 15.5)
(83, 13.6)
(15, 22.4)
(25, 12.0)
(195, 7.6)
(99, 17.2)
(153, 19.0)
(116, 12.2)
(134, 10.8)
(93, 22.2)
(21, 12.5)
(4, 12.9)
(43, 12.9)
(150, 16.1)
(167, 12.2)
(52, 22.6)
(122, 11.6)
(40, 16.6)
(81, 12.3)
(143, 10.4)
(106, 7.2)
(151, 11.6)
(139, 20.7)
(1, 10.4)
(3, 18.5)
(160, 14.4)
(28, 18.9)
(171, 14.5)
(130, 1.6)
(35, 12.8)
(147, 25.4)
(88, 12.9)
(104, 20.7)
```

```
# to get Linear Regression
lm = LinearRegression()
lm.fit(trainX, trainY)
print (lm.intercept_)
print (lm.coef_)
for z in zip(feature_cols, lm.coef_):
    print (z)
```

```
In [185]:
     ...: lm = LinearRegression()
     ...: lm.fit(trainX, trainY)
     ...: print (lm.intercept_)
     ...: print (lm.coef_)
     ...: for z in zip(feature_cols, lm.coef_):
     ...:     print (z)
3.008005505418147
[0.0464221  0.18318345]
('TV', 0.04642210232095002)
('Radio', 0.1831834514171443)
```

to predict TestY base on the Linear Regression

```
lm.score(trainX, trainY)
predictY = lm.predict(testX)
for p in zip(testX.index, testY, predictY):
    print (p)
```

```
   ...: lm.score(trainX, trainY)
   ...: predictY = lm.predict(testX)
   ...: for p in zip(testX.index, testY, predictY):
   ...:     print (p)
(176, 20.2, 20.07139595473989)
(148, 10.9, 12.154338485725162)
(8, 4.8, 3.7919208333543204)
(192, 5.9, 4.5575178161487795)
(163, 18.0, 17.339170247044386)
(125, 10.6, 9.217577554527292)
(23, 15.5, 16.701971794240777)
(97, 15.5, 15.438304704321835)
(83, 13.6, 14.33494089223405)
(15, 22.4, 20.816734931529567)
(25, 12.0, 15.853518285555912)
(195, 7.6, 5.459108584321871)
(99, 17.2, 16.923023663305507)
(153, 19.0, 18.232494654257515)
(116, 12.2, 12.089485503759553)
(134, 10.8, 11.791862305762972)
(93, 22.2, 21.341506954470272)
(21, 12.5, 14.962848198639119)
(4, 12.9, 13.37950288035107)
(43, 12.9, 14.15147946752672)
(150, 16.1, 18.584939601607125)
(167, 12.2, 13.560650212759763)
(52, 22.6, 20.69249837176665)
(122, 11.6, 13.8461967087121)
(40, 16.6, 16.493472192012845)
(81, 12.3, 14.891077792792256)
(143, 10.4, 8.907903081267243)
(106, 7.2, 6.183576029030485)
(151, 11.6, 10.16382087815711)
(139, 20.7, 19.63320574177444)
(1, 10.4, 12.272898699394194)
(3, 18.5, 17.606430550570135)
(160, 14.4, 14.33143862643234)
(28, 18.9, 19.522096096275124)
(171, 14.5, 14.472975471832742)
(130, 1.6, 10.294565653161726)
(35, 12.8, 17.253962800928612)
(147, 25.4, 23.273849909313263)
(88, 12.9, 11.778255151495213)
(104, 20.7, 20.348942661876492)
```

9- Feature selection: using the scikit , in order to check which predictors are best as input variable to the model run the following code sinpet and don't forget to change the path name:

```
from sklearn.feature_selection import RFE
from sklearn.svm import SVR
feature_cols = ['TV', 'Radio','Newspaper']
X = data_liping_adv[feature_cols]
Y = data_liping_adv['Sales']
estimator = SVR(kernel="linear")
selector = RFE(estimator,2,step=1)
selector = selector.fit(X, Y)
print(selector.support_)
print(selector.ranking_)
```

```
[ True  True False]
[1 1 2]
```

for s in zip(selector.support_,selector.ranking_):
    print(s)

```
In [193]: for s in zip(selector.support_,selector.ranking_ ):
     ...:     print(s)
(True, 1)
(True, 1)
(False, 2)
```