G. H. Raisoni College Of Engineering And Management, Wagholi Pune

2021- 2022

Group C :-Assignment no :-18

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	13-12-2021
Subject Name /Code	Python for Data Science / UCSP204		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

VIJAYANT
PAGE NO.:
DATE:

Group C : Assignment No 18

Aim :-

Study and implement any one type of regression mode using appropriate dataset.

Theory :-

- **Linear regression :→**

It is an approach for predicting a response using a single featore.

It is assumed that the two variables are linearly related.

In statistics : Linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables.

Conclusion :→

Hence we conclude that using Linear Regression algorithm we can find the best salary for our candidate.

In [1]:

```
print("****************************************")
print("SCOB77_Pratham pitty_Group B Assignment (18)")
print("****************************************")
```

```
****************************************
SCOB77_Pratham pitty_Group B Assignment (18)
****************************************
```

In [2]:

```
# conventional way to import pandas
import pandas as pd
```

In [3]:

```
# read CSV file from the 'data' subdirectory using a relative path
data = pd.read_csv('data/Advertising.csv', index_col=0)

# display the first 5 rows
data.head()
```

Out[3]:

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

In [5]:

```
# display the last 5 rows
data.tail()
```

Out[5]:

	TV	Radio	Newspaper	Sales
196	38.2	3.7	13.8	7.6
197	94.2	4.9	8.1	9.7
198	177.0	9.3	6.4	12.8
199	283.6	42.0	66.2	25.5
200	232.1	8.6	8.7	13.4

In [6]:

```
# check the shape of the DataFrame (rows, columns)
data.shape
```

Out[6]:

```
(200, 4)
```

Visualizing data using seaborn

In [7]:

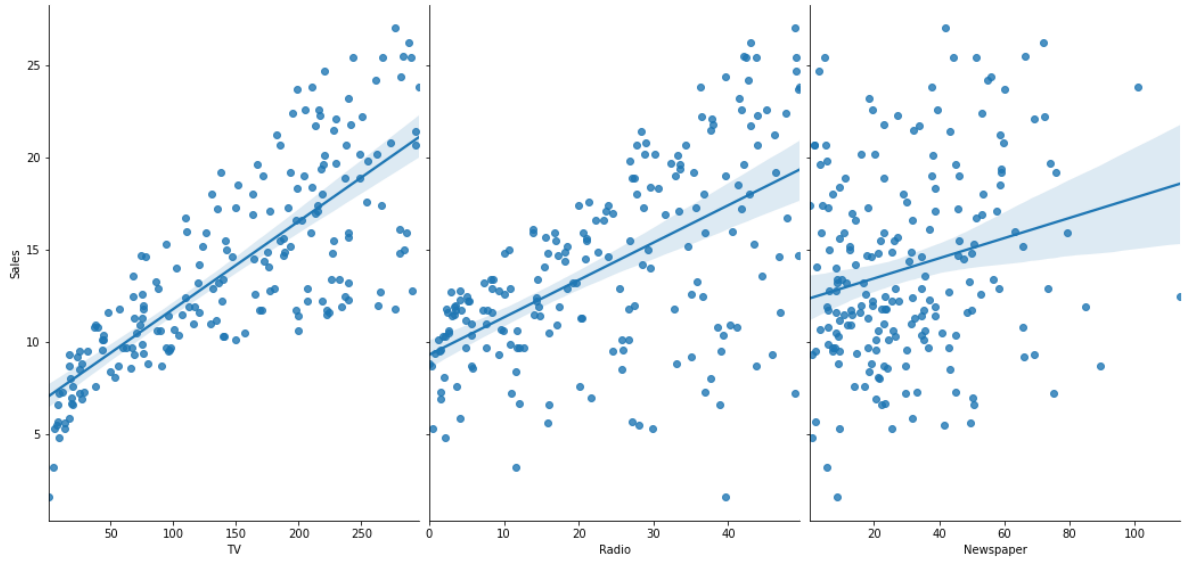
```
# conventional way to import seaborn
import seaborn as sns

# allow plots to appear within the notebook
%matplotlib inline
```

In [8]:

```
# visualize the relationship between the features and the response using scatterplots
sns.pairplot(data, x_vars=['TV','Radio','Newspaper'], y_vars='Sales', height=7, aspect=0.7,
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x7fb0288ec550>
```



Linear regression

In [11]:

```
# create a Python list of feature names
feature_cols = ['TV', 'Radio', 'Newspaper']

# use the list to select a subset of the original DataFrame
X = data[feature_cols]

# equivalent command to do this in one line
X = data[['TV', 'Radio', 'Newspaper']]

# print the first 5 rows
X.head()
```

Out[11]:

	TV	Radio	Newspaper
1	230.1	37.8	69.2
2	44.5	39.3	45.1
3	17.2	45.9	69.3
4	151.5	41.3	58.5
5	180.8	10.8	58.4

In [12]:

```
# check the type and shape of X
print(type(X))
print(X.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(200, 3)
```

In [13]:

```
# select a Series from the DataFrame
y = data['Sales']

# equivalent command that works if there are no spaces in the column name
y = data.Sales

# print the first 5 values
y.head()
```

Out[13]:

```
1    22.1
2    10.4
3     9.3
4    18.5
5    12.9
Name: Sales, dtype: float64
```

In [14]:

```
# check the type and shape of y
print(type(y))
print(y.shape)
```

```
<class 'pandas.core.series.Series'>
(200,)
```

Splitting X and y into training and testing sets

In [15]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

In [16]:

```
# default split is 75% for training and 25% for testing
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(150, 3)
(150,)
(50, 3)
(50,)
```

Linear regression in scikit-learn

In [17]:

```
# import model
from sklearn.linear_model import LinearRegression

# instantiate
linreg = LinearRegression()

# fit the model to the training data (learn the coefficients)
linreg.fit(X_train, y_train)
```

Out[17]:

```
LinearRegression()
```

Interpreting model coefficients

In [18]:

```
# print the intercept and coefficients
print(linreg.intercept_)
print(linreg.coef_)
```

```
2.8769666223179318
[0.04656457 0.17915812 0.00345046]
```

In [19]:

```
# pair the feature names with the coefficients
list(zip(feature_cols, linreg.coef_))
```

Out[19]:

```
[('TV', 0.04656456787415029),
 ('Radio', 0.17915812245088839),
 ('Newspaper', 0.003450464711180378)]
```

Making predictions

In [20]:

```
# make predictions on the testing set
y_pred = linreg.predict(X_test)
```

We need an **evaluation metric** in order to compare our predictions with the actual values!

Model evaluation metrics for regression

Evaluation metrics for classification problems, such as **accuracy**, are not useful for regression problems. Instead, we need evaluation metrics designed for comparing continuous values.

Let's create some example numeric predictions, and calculate **three common evaluation metrics** for regression problems:

In [21]:

```
# define true and predicted response values
true = [100, 50, 30, 20]
pred = [90, 50, 50, 30]
```

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

In [22]:

```
# calculate MAE by hand
print((10 + 0 + 20 + 10)/4.)

# calculate MAE using scikit-learn
from sklearn import metrics
print(metrics.mean_absolute_error(true, pred))
```

```
10.0
10.0
```

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In [23]:

```
# calculate MSE by hand
print((10**2 + 0**2 + 20**2 + 10**2)/4.)

# calculate MSE using scikit-learn
print(metrics.mean_squared_error(true, pred))
```

```
150.0
150.0
```

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

In [24]:

```
# calculate RMSE by hand
import numpy as np
print(np.sqrt((10**2 + 0**2 + 20**2 + 10**2)/4.))

# calculate RMSE using scikit-learn
print(np.sqrt(metrics.mean_squared_error(true, pred)))
```

```
12.24744871391589
12.24744871391589
```

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

Computing the RMSE for our Sales predictions

In [25]:

```
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
1.404651423032895
```

Feature selection

Does **Newspaper** "belong" in our model? In other words, does it improve the quality of our predictions?

Let's **remove it** from the model and check the RMSE!

In [26]:

```
# create a Python list of feature names
feature_cols = ['TV', 'Radio']

# use the list to select a subset of the original DataFrame
X = data[feature_cols]

# select a Series from the DataFrame
y = data.Sales

# split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

# fit the model to the training data (learn the coefficients)
linreg.fit(X_train, y_train)

# make predictions on the testing set
y_pred = linreg.predict(X_test)

# compute the RMSE of our predictions
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
1.3879034699382886
```

The RMSE **decreased** when we removed Newspaper from the model. (Error is something we want to minimize, so **a lower number for RMSE is better**.) Thus, it is unlikely that this feature is useful for predicting Sales, and should be removed from the model.

Resources

Linear regression:

- Longer notebook on linear regression (https://github.com/justmarkham/DAT4/blob/master/notebooks/08_linear_regression.ipynb) by me
- Chapter 3 of An Introduction to Statistical Learning (https://www.statlearning.com/) and related videos (https://www.dataschool.io/15-hours-of-expert-machine-learning-videos/) by Hastie and Tibshirani (Stanford)
- Quick reference guide to applying and interpreting linear regression (https://www.dataschool.io/applying-and-interpreting-linear-regression/) by me
- Introduction to linear regression (http://people.duke.edu/~rnau/regintro.htm) by Robert Nau (Duke)

Pandas:

- pandas Q&A video series (https://www.dataschool.io/easier-data-analysis-with-pandas/) by me
- Three-part pandas tutorial (http://www.gregreda.com/2013/10/26/intro-to-pandas-data-structures/) by Greg Reda
- read_csv (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html) and read_table (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_table.html) documentation

Seaborn:

- Official seaborn tutorial (http://seaborn.pydata.org/tutorial.html)
- Example gallery (http://seaborn.pydata.org/examples/index.html)

Comments or Questions?

- Email: kevin@dataschool.io (mailto:kevin@dataschool.io)
- Website: https://www.dataschool.io (https://www.dataschool.io)
- Twitter: @justmarkham (https://twitter.com/justmarkham)