Performing Linear Regression on Advertising dataset

Why do you use Regression Analysis?

Regression analysis estimates the relationship between two or more variables.

In [1]:
```
# import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt

# this allows plots to appear directly in the notebook
%matplotlib inline
```

Let's take a look at the data, ask some questions about that data, and then use Linear regression to answer those questions.

In [2]:
```
# read data into a DataFrame
data = pd.read_csv('Advertising.csv', index_col=0)
data.head()
data.columns = ['TV','Sales']
```

Indepenent variables

- TV: Advertising dollars spent on TV for a single product in a given market (in thousands of dollars)

**Target Variable* *

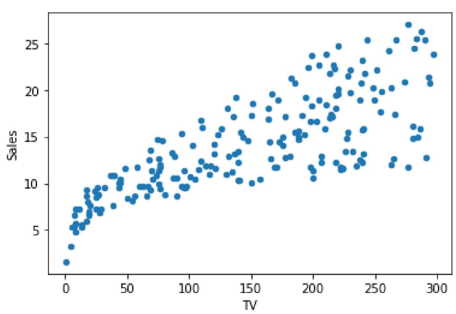
- Sales: sales of a single product in a given market (in thousands of widgets)

In [3]:
```
# print the shape of the DataFrame
data.shape
```

Out[3]:
```
(200, 2)
```

In [4]:
```
# visualize the relationship between the features and the response using scatterplots
data.plot(kind='scatter', x='TV', y='Sales')
```

Out[4]:
```
<AxesSubplot:xlabel='TV', ylabel='Sales'>
```



Questions About the Advertising Data

On the basis of this data, how should you spend advertising money in the future? These general questions might lead you to more specific questions:

1. Is there a relationship between TV ads and sales?
2. How strong is that relationship?
3. Given ad spending, can sales be predicted?

Exploring these questions below.

In [5]:
```
# create X and y
#taking only one variable for now
feature_cols = ['TV']
X = data[['TV']]
X
```

Out[5]:

	TV
1	230.1
2	44.5
3	17.2
4	151.5
5	180.8
...	...
196	38.2
197	94.2
198	177.0
199	283.6
200	232.1

200 rows × 1 columns

In [6]:
```
y = data.Sales
y
```

Out[6]:
```
1      22.1
2      10.4
3       9.3
4      18.5
5      12.9
       ...
196     7.6
197     9.7
198    12.8
199    25.5
200    13.4
Name: Sales, Length: 200, dtype: float64
```

In [7]:
```
# follow the usual sklearn pattern: import, instantiate, fit
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X, y)

# print intercept and coefficients
print(lm.intercept_)
print(lm.coef_)
```

```
7.032593549127693
[0.04753664]
```

Interpreting Model Coefficients

How do you interpret the TV coefficient (β_1)?

- A "unit" increase in TV ad spending was **associated with** a 0.047537 "unit" increase in Sales.
- Or more clearly: An additional $1,000 spent on TV ads was **associated with** an increase in sales of 47.537 widgets.

Note that if an increase in TV ad spending was associated with a **decrease** in sales, β_1 would be **negative.**

Using the Model for Prediction

Let's say that there was a new market where the TV advertising spend was **$50,000**. How would you predict the sales in that market?

$$y = \beta_0 + \beta_1 x$$
$$y = 7.032594 + 0.047537 \times 50$$

Manually calculate the prediction 7.032594 + 0.047537*50= 9.409444

In [8]:
```
# manually calculate the prediction
7.032594 + 0.047537*50
```

Out[8]:
```
9.409444
```

Thus, you would predict Sales of **9,409 widgets** in that market.

In [9]:
```
# you have to create a DataFrame since the Statsmodels formula interface expects it
X_new = pd.DataFrame({'TV': [50]})
X_new.head()
```

Out[9]:

	TV
0	50

In [10]:
```
# use the model to make predictions on a new value
lm.predict(X_new)
```

Out[10]:
```
array([9.40942557])
```

In [11]:
```
data['TV'].min()
```

Out[11]:
```
0.7
```

Plotting the Least Squares Line

Let's make predictions for the **smallest and largest observed values of x**, and then use the predicted values to plot the least squares line:

In [12]:
```
# create a DataFrame with the minimum and maximum values of TV
X_new = pd.DataFrame({'TV': [data['TV'].min(), data['TV'].max()]})
X_new.head()
```

Out[12]:

	TV
0	0.7
1	296.4

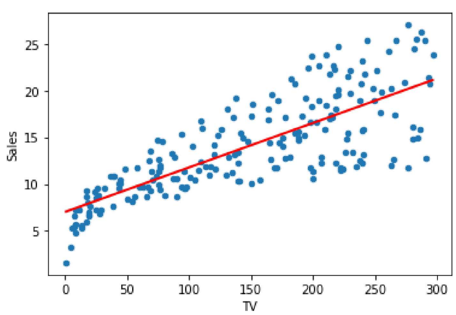
In [13]:
```
# make predictions for those x values and store them
preds = lm.predict(X_new)
preds
```

Out[13]:
```
array([ 7.0658692 , 21.12245377])
```

In [14]:
```
# first, plot the observed data
data.plot(kind='scatter', x='TV', y='Sales')

# then, plot the least squares line
plt.plot(X_new, preds, c='red', linewidth=2)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x2c8d86e8970>]
```



How Well Does the Model Fit the data?

The most common way to evaluate the overall fit of a linear model is by the R-squared value. R-squared is the proportion of variance explained, meaning the proportion of variance in the observed data that is explained by the model, or the reduction in error over the null model. (The null model just predicts the mean of the observed response, and thus it has an intercept and no slope.)

R-squared is between 0 and 1, and higher is better because it means that more variance is explained by the model. Here's an example of what R-squared "looks like":

Type *Markdown* and LaTeX: α^2

In [15]:
```
predictions = lm.predict(X)
print(sqrt(mean_squared_error(y, predictions)))
```

```
3.2423221486546887
```

In [17]:
```
r2 = r2_score(y, predictions)
r2
```

Out[17]:
```
0.611875050850071
```

There is no correct value for MSE. Simply put, the lower the value the better and 0 means the model is perfect. Since there is no correct answer, the MSE's basic value is in selecting one prediction model over another.

Similarly, there is also no correct answer as to what R2 should be. 100% means perfect correlation. Yet, there are models with a low R2 that are still good models.

In [ ]:

```

```