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

%matplotlib inline
```

In [2]:
```
adv_data=pd.read_csv(r"C:\Users\s323\Desktop\Data Science\Advertising.csv",index_c
```

In [3]:
```
adv_data.shape
```

Out[3]:
```
(200, 4)
```

In [4]:
```
adv_data.head()
```

Out[4]:

	TV Ad Budget ($)	Radio Ad Budget ($)	Newspaper Ad Budget ($)	Sales ($)
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]:
```
adv_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   TV Ad Budget ($)        200 non-null    float64
 1   Radio Ad Budget ($)     200 non-null    float64
 2   Newspaper Ad Budget ($) 200 non-null    float64
 3   Sales ($)               200 non-null    float64
dtypes: float64(4)
memory usage: 7.8 KB
```

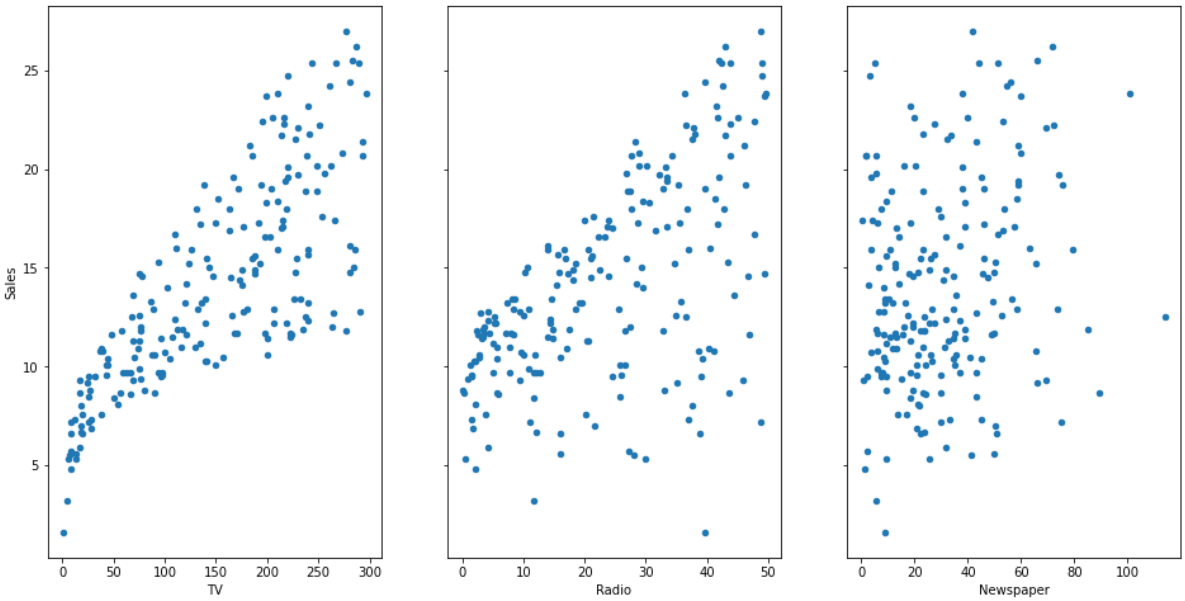
In [6]:
```
adv_data.isnull().sum()
```

Out[6]:
```
TV Ad Budget ($)           0
Radio Ad Budget ($)        0
Newspaper Ad Budget ($)    0
Sales ($)                  0
dtype: int64
```

In [7]:
```
 adv_data.columns = ["TV","Radio","Newspaper","Sales"]
```

In [8]:
```
# axs= using for multiple axis
fig,axs=plt.subplots(1,3,sharey=True)
# 1 row, 3 column and 3rd argument is for
adv_data.plot(kind="scatter",x="TV",y="Sales",ax=axs[0],figsize=(16,8))
adv_data.plot(kind="scatter",x="Radio",y="Sales",ax=axs[1])
adv_data.plot(kind="scatter",x="Newspaper",y="Sales",ax=axs[2])
```

Out[8]:
```
<AxesSubplot:xlabel='Newspaper', ylabel='Sales'>
```



In [9]:
```
# apply linear regression model to see relationship between sales
feature_cols=['TV']
x = adv_data[feature_cols]
y = adv_data.Sales
# x = indepedent variable and y= dependent variable
```

In [10]:
```
from sklearn.linear_model import LinearRegression
# intialising the model
lm = LinearRegression()
# fitting the model
lm.fit(x,y)
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
#intercept and coeffiecnt helps us in making linear equation
print(lm.intercept_)
print(lm.coef_)
```

```
7.032593549127693
[0.04753664]
```

In [12]:
```
# it means a unit increase in TV ad spending is associated with 0.04753664 increase
# now we have linear equation, we can use it for prediction
```

Lets say there was a new market where Tv advertising spends were 50k dollars, wat would we predict for the sales in that market... use the previous results to calculate new value for sales

In [13]:
```
# linear equation: y=bx+c ( b= coeffiecent, c= intercept of y)
7.032594 + 0.047537*50
```

Out[13]:
```
9.409444
```

- Let's predict the new X value

In [14]:
```
X_new = pd.DataFrame({"TV":[50]})
X_new.head()
```

Out[14]:

	TV
0	50

In [15]:
```
lm.predict(X_new)
```

Out[15]:
```
array([9.40942557])
```

In [16]:
```
# let's predict the value for smallest and largest observed value for X and then us
X_new = pd.DataFrame({"TV":[adv_data.TV.min(),adv_data.TV.max()]})
X_new.head()
```

Out[16]:

	TV
0	0.7
1	296.4

In [17]:
```
# least square line
preds = lm.predict(X_new)
preds
```

Out[17]:
```
array([ 7.0658692 , 21.12245377])
```

In [18]:
```
adv_data.plot(kind="scatter",x="TV",y="Sales")
#plotting the least square line
plt.plot(X_new,preds,c="red",linewidth=2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File ~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in Index.get_
loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File ~\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._libs.ind
ex.IndexEngine.get_loc()

File ~\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:142, in pandas._libs.ind
ex.IndexEngine.get_loc()

TypeError: '(slice(None, None, None), None)' is an invalid key

During handling of the above exception, another exception occurred:

InvalidIndexError                         Traceback (most recent call last)
Input In [18], in <cell line: 3>()
      1 adv_data.plot(kind="scatter",x="TV",y="Sales")
      2 #plotting the least square line
----> 3 plt.plot(X_new,preds,c="red",linewidth=2)

File ~\Anaconda3\lib\site-packages\matplotlib\pyplot.py:2757, in plot(scalex, scal
ey, data, *args, **kwargs)
   2755 @_copy_docstring_and_deprecators(Axes.plot)
   2756 def plot(*args, scalex=True, scaley=True, data=None, **kwargs):
-> 2757     return gca().plot(
   2758         *args, scalex=scalex, scaley=scaley,
   2759         **({"data": data} if data is not None else {}), **kwargs)

File ~\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:1632, in Axes.plot(sel
f, scalex, scaley, data, *args, **kwargs)
   1390 """
   1391 Plot y versus x as lines and/or markers.
   1392
   (...)
   1629 (``'green'``) or hex strings (``'#008000'``).
   1630 """
   1631 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1632 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633 for line in lines:
   1634     self.add_line(line)

File ~\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:312, in _process_plot_
var_args.__call__(self, data, *args, **kwargs)
    310     this += args[0],
    311     args = args[1:]
--> 312 yield from self._plot_args(this, kwargs)

File ~\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:487, in _process_plot_
var_args._plot_args(self, tup, kwargs, return_kwargs)
    484         kw[prop_name] = val
    486 if len(xy) == 2:
--> 487     x = _check_1d(xy[0])
    488     y = _check_1d(xy[1])
    489 else:

File ~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py:1327, in _check_1d
(x)
   1321 with warnings.catch_warnings(record=True) as w:
   1322     warnings.filterwarnings(
   1323         "always",
```

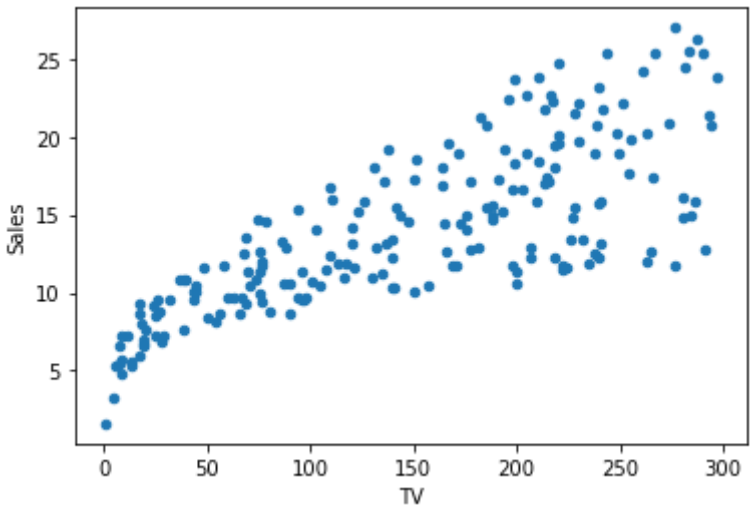
```
   1324         category=Warning,
   1325         message='Support for multi-dimensional indexing')
-> 1327     ndim = x[:, None].ndim
   1328     # we have definitely hit a pandas index or series object
   1329     # cast to a numpy array.
   1330     if len(w) > 0:

File ~\Anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFrame.__getit
em__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:3628, in Index.get_
loc(self, key, method, tolerance)
   3623         raise KeyError(key) from err
   3624     except TypeError:
   3625         # If we have a listlike key, _check_indexing_error will raise
   3626         #  InvalidIndexError. Otherwise we fall through and re-raise
   3627         #  the TypeError.
-> 3628         self._check_indexing_error(key)
   3629         raise
   3631 # GH#42269

File ~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:5637, in Index._che
ck_indexing_error(self, key)
   5633 def _check_indexing_error(self, key):
   5634     if not is_scalar(key):
   5635         # if key is not a scalar, directly raise an error (the code below
   5636         # would convert to numpy arrays and raise later any way) - GH29926
-> 5637         raise InvalidIndexError(key)

InvalidIndexError: (slice(None, None, None), None)
```



In [ ]:
```
# to check for null hypotheris = 95%
import statsmodels.formula.api as smf
lm = smf.ols(formula ="Sales ~ TV",data=adv_data).fit()
```

In [ ]:
```
lm.conf_int()
```

In [ ]:
```
lm.pvalues
# pvalues show that the possibloity that the coeffiecents value is actually 0 but
```

How well this model fits the data- the most common way for a linear model is by using R2 value

In [ ]:
```
# print the r2 value
lm.rsquared
```

Multiple linear model, earlier we use feature as TV now we will use other all

In [ ]:
```
feature_cols=["TV","Radio","Newspaper"]
x=adv_data[feature_cols]
y=adv_data.Sales
```

In [ ]:
```
from sklearn import model_selection
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.3,r
```

In [ ]:
```
#apply linear regression
lm=LinearRegression()
lm.fit(x,y)
print(lm.intercept_)
print(lm.coef_)
```

In [ ]:
```
lm=LinearRegression()
lm.fit(x_train,y_train)
```

In [ ]:
```
print(lm.intercept_)
print(lm.coef_)
```

In [ ]:
```
predictions=lm.predict(x_test)
print(sqrt(mean_squared_error(y_test,predictions)))
```

In [ ]:
```
lm = smf.ols(formula="Sales ~ TV + Radio + Newspaper", data=adv_data).fit()
lm.conf_int()
lm.summary()
```

In [ ]:
```
lm = smf.ols(formula="Sales ~ TV + Radio", data=adv_data).fit()
lm.rsquared
```

In [ ]:
```
lm = smf.ols(formula="Sales ~ TV + Radio+ Newspaper", data=adv_data).fit()
lm.rsquared
```

Let's fit a new cateogorical variable size

In [ ]:
```
import numpy as np
np.random.seed(12345)

nums=np.random.rand(len(adv_data))
mask_large=nums>0.5

adv_data["size"]="small"
adv_data.loc[mask_large,"Size"]="large"
adv_data.head()
```

In [ ]:
```
adv_data["Islarge"]=adv_data.Size.map({"small":0, "large":1})
adv_data.head()
```

In [ ]:
```
feature_cols=["TV","Radio","Newspaper","Islarge"]
x=adv_data[feature_cols]
y=adv_data.Sales
```

In [ ]:
```
lm=LinearRegression()
lm.fit(x,y)
zip(feature_cols,lm.coef_)
```

In [ ]:
```

```