Data

This sample data displays sales (in thousands of units) for a particular product as a function of advertising budgets (in thousands of dollars) for TV, radio, and newspaper media.

Performing Linear Regression on Advertising dataset

Indepenent variables

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

Target Variable

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

Question

Previously, we explored **Is there a relationship between *total* advertising spend and *sales*?** as well as predicting the total sales for some value of total spend. Now we want to expand this to **What is the relationship between each advertising channel (TV,Radio,Newspaper) and sales?**

Multiple Linear Regression

In [1]:
```
## Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
# read data into a DataFrame
df = pd.read_csv("Advertising.csv")
df.head()
```

Out[2]:

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

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

In [4]:
```
df.columns
```

Out[4]:
```
Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

In [5]:
```
df.dtypes
```

Out[5]:
```
TV           float64
radio        float64
newspaper    float64
sales        float64
dtype: object
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

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 ads and sales?
2. How strong is that relationship?
3. Which ad types contribute to sales?
4. What is the effect of each ad type of sales?
5. Given ad spending, can sales be predicted?

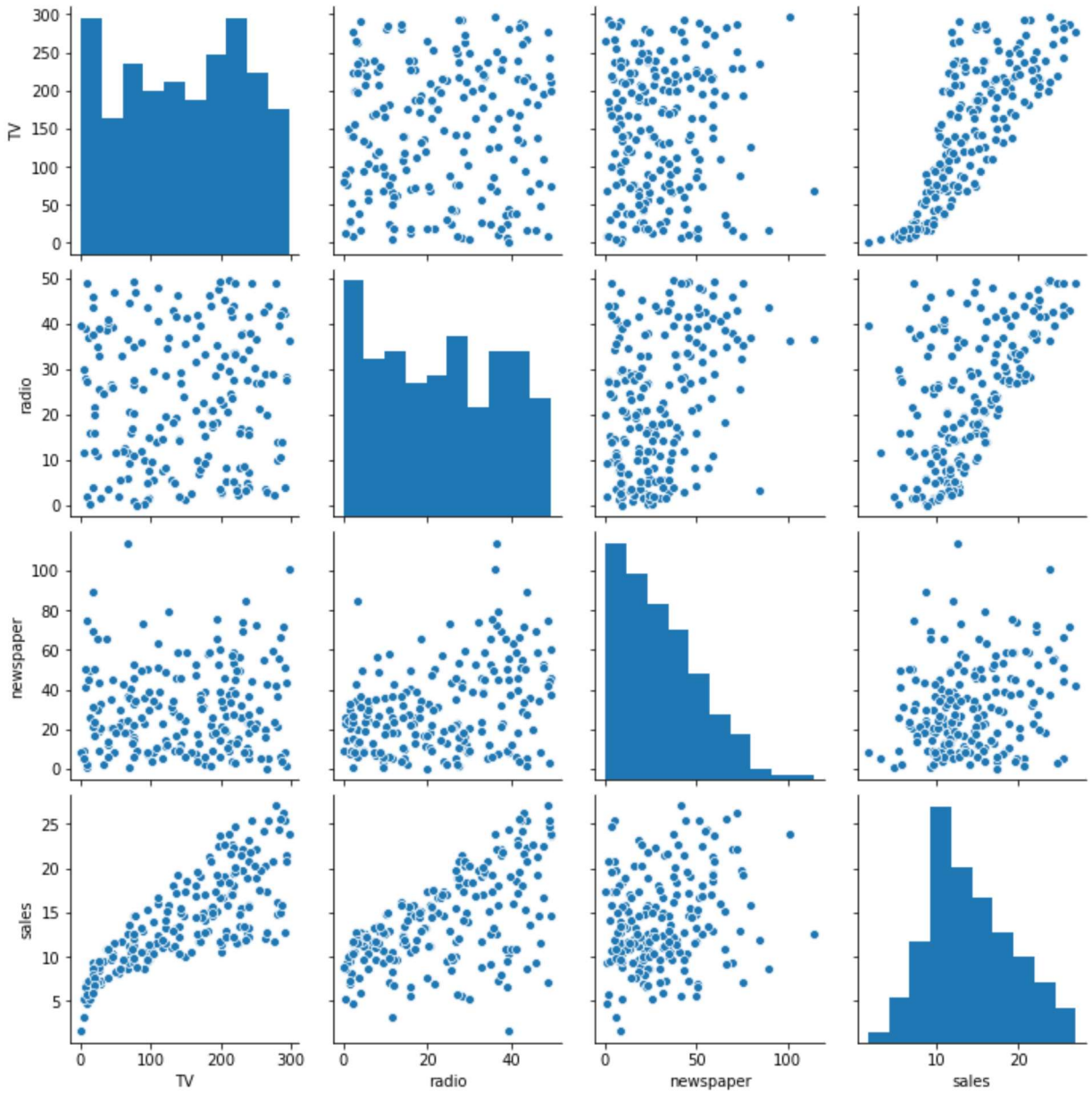
EDA

In [8]:
```
df.describe()
```

Out[8]:

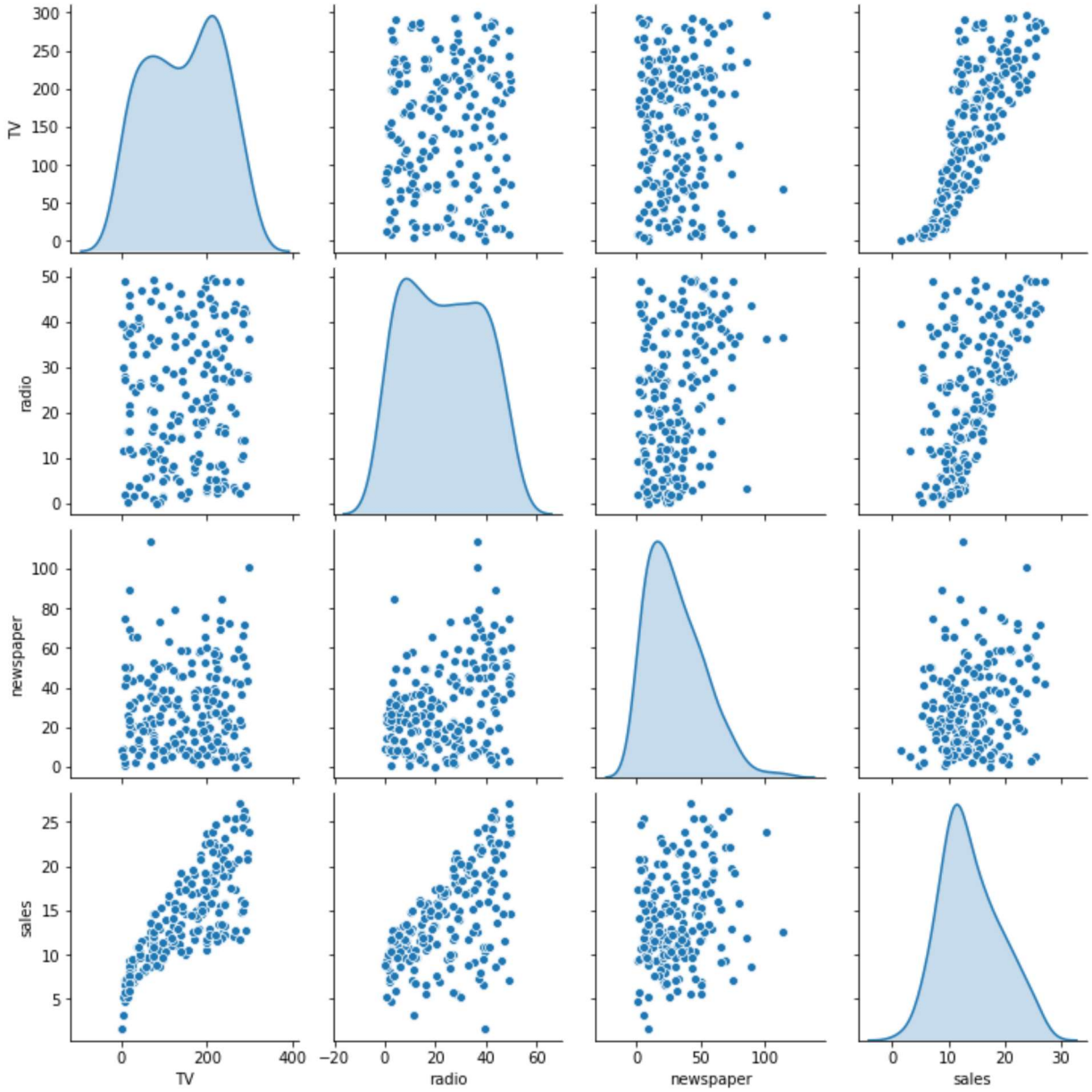
	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

In [9]:
```
sns.pairplot(df)
plt.show()
```



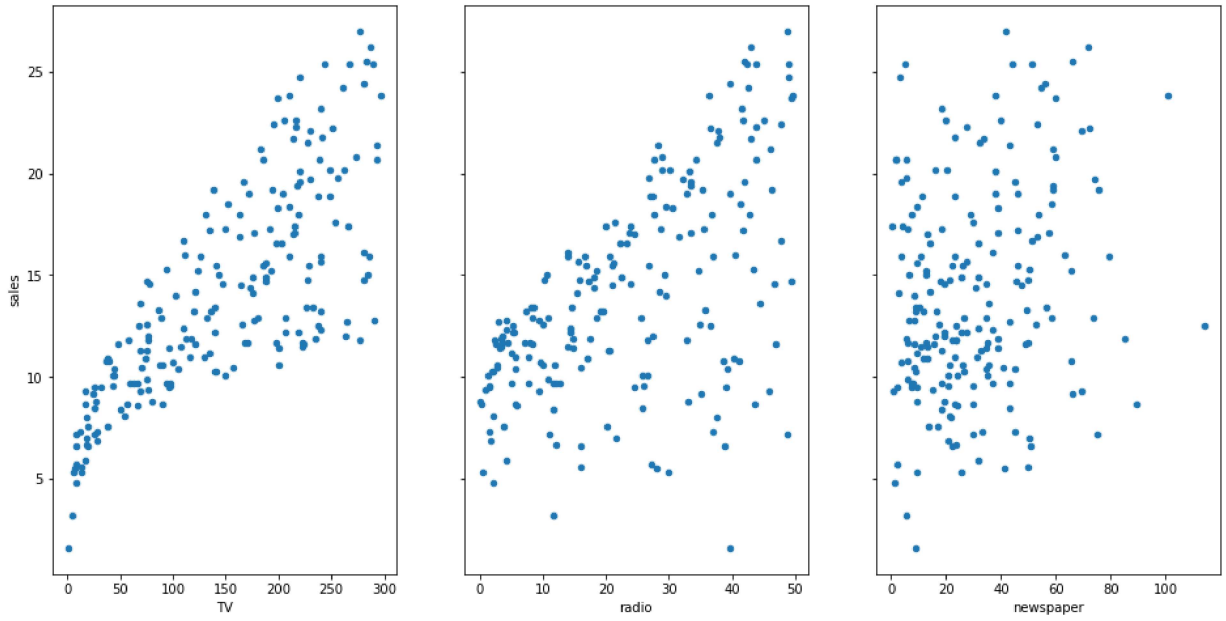
In [10]:
```
# Relationships between features
sns.pairplot(df,diag_kind='kde')
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x19dd3f6f040>
```



In [11]:
```
# visualize the relationship between the features and the response using scatterp
fig, axs = plt.subplots(1, 3, sharey=True)
df.plot(kind='scatter', x='TV', y='sales', ax=axs[0], figsize=(16, 8))
df.plot(kind='scatter', x='radio', y='sales', ax=axs[1])
df.plot(kind='scatter', x='newspaper', y='sales', ax=axs[2])
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19dd4905070>
```



In [12]:
```
df.corr()
```

Out[12]:

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

Least Squares Line

Simple linear regression can easily be extended to include multiple features. This is called **multiple linear regression**:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Each x represents a different feature, and each feature has its own coefficient. In this case:

$$y = \beta_0 + \beta_1 \times TV + \beta_2 \times Radio + \beta_3 \times Newspaper$$

In [13]:
```
# X = df[['TV','radio','newspaper']]
# y = df['sales']
```

In [14]:
```
X = df.drop('sales',axis=1)
y = df['sales']
```

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_st
```

Modelling

In [16]:
```
# follow the usual sklearn pattern: import, instantiate, fit

from sklearn.linear_model import LinearRegression #import

model = LinearRegression()   #instantiate

model.fit(X_train,y_train) #fit
```

Out[16]:
```
LinearRegression()
```

In [17]:
```
model.intercept_ , model.coef_
```

Out[17]:
```
(2.7089490925159083, array([0.04405928, 0.1992875 , 0.00688245]))
```

Predictions

In [18]:
```
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)
```

Evaluation

In [19]:
```
train_res = y_train - train_predictions
test_res = y_test - test_predictions
```

In [20]:
```
model.score(X_train,y_train)  # Train R2
```

Out[20]:
```
0.9055159502227753
```

In [21]:
```
model.score(X_test,y_test)   # Test R2
```

Out[21]:
```
0.8609466508230368
```

In [22]:
```
from sklearn.metrics import r2_score
r2_score(y_test,test_predictions)
```

Out[22]:
```
0.8609466508230368
```

In [23]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model,X,y,cv=5)
scores

# Average of the MSE scores (we set back to positive)
abs(scores.mean())
```

Out[23]:
```
0.8871063495438436
```

In [24]:
```
from sklearn.metrics import mean_absolute_error
MAE = mean_absolute_error(y_test,test_predictions)
MAE
```

Out[24]:
```
1.5116692224549089
```

In [25]:
```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,test_predictions)
MSE
```

Out[25]:
```
3.7967972367152205
```

In [26]:
```
RMSE = np.sqrt(MSE)
RMSE
```

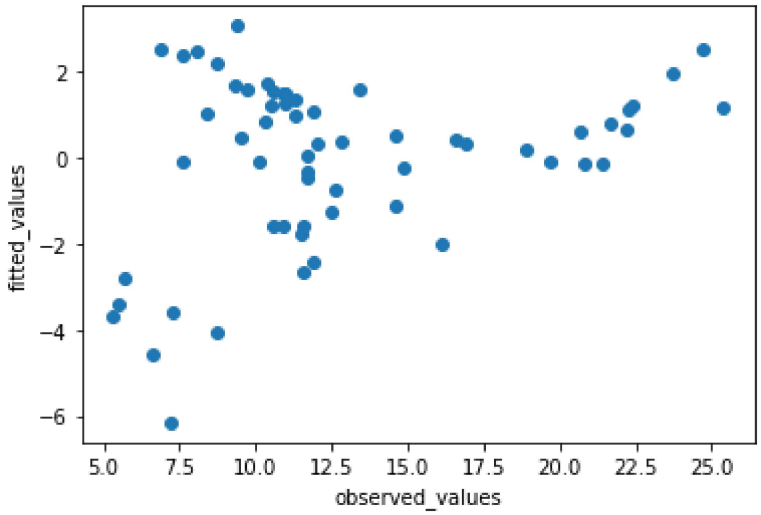
Out[26]:
```
1.9485372043446387
```

Diagnosis Test

It's also important to plot out residuals, this helps us understand if Linear Regression was a valid model choice.

1. Linearity (Observed values VS Fitted values)

In [27]:
```
plt.scatter(y_test,test_res)
plt.xlabel("observed_values")
plt.ylabel("fitted_values")
plt.show()
```

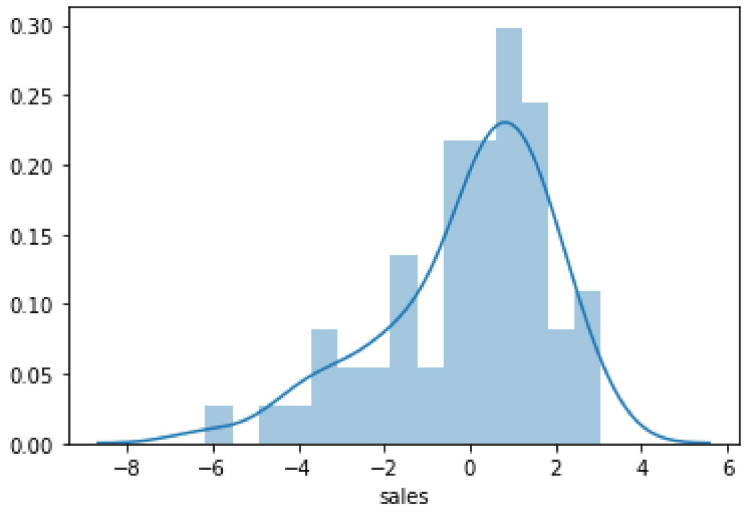


2. Normality of Residuals

It's also important to plot out residuals and check for normal distribution

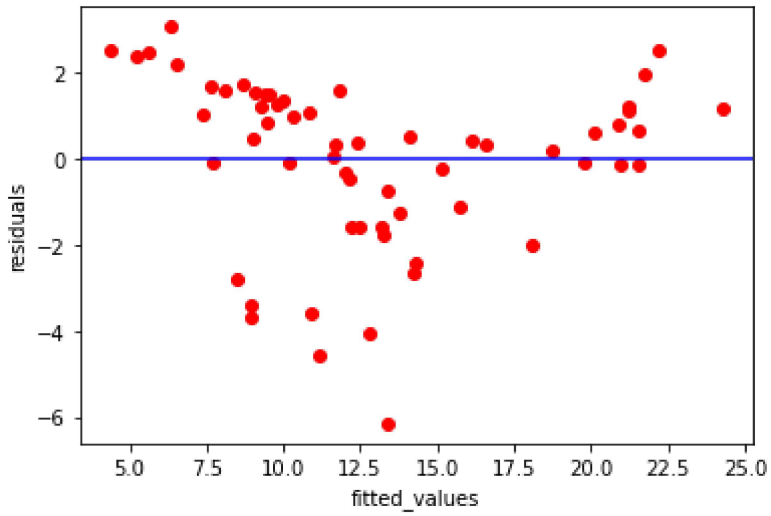
In [28]:
```
sns.distplot(test_res,bins=15,kde=True)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19dd4cb2700>
```



3. Homoscadesicity (y_hat vs residuals)

In [29]:
```
# Homoscedasticity (Residuals VS Fitted Values)
plt.scatter(test_predictions,test_res,c="r")
plt.axhline(y=0,color='blue')
plt.xlabel("fitted_values")
plt.ylabel("residuals")
plt.show()
```



Variables Significance

In [30]:
```
#importing required libraries
import statsmodels.formula.api as smf

#model
model1=smf.ols("y~X",data=df).fit()

# P-values for the variables and R-squared value for prepared model
model1.summary()
```

Out[30]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.897
Model:	OLS	**Adj. R-squared:**	0.896
Method:	Least Squares	**F-statistic:**	570.3
Date:	Mon, 20 Sep 2021	**Prob (F-statistic):**	1.58e-96
Time:	21:06:39	**Log-Likelihood:**	-386.18
No. Observations:	200	**AIC:**	780.4
Df Residuals:	196	**BIC:**	793.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.9389	0.312	9.422	0.000	2.324	3.554
X[0]	0.0458	0.001	32.809	0.000	0.043	0.049
X[1]	0.1885	0.009	21.893	0.000	0.172	0.206
X[2]	-0.0010	0.006	-0.177	0.860	-0.013	0.011

Omnibus:	60.414	**Durbin-Watson:**	2.084
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	151.241
Skew:	-1.327	**Prob(JB):**	1.44e-33
Kurtosis:	6.332	**Cond. No.**	454.

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

What are a few key things you learn from this output?

- TV and Radio have significant **p-values**, whereas Newspaper does not. Thus, reject the null hypothesis for TV and Radio (that there is no association between those features and Sales), and fail to reject the null hypothesis for Newspaper.
- TV and Radio ad spending are both **positively associated** with Sales, whereas Newspaper ad spending was **slightly negatively associated** with Sales. (However, this is irrelevant since as you have failed to reject the null hypothesis for Newspaper.)

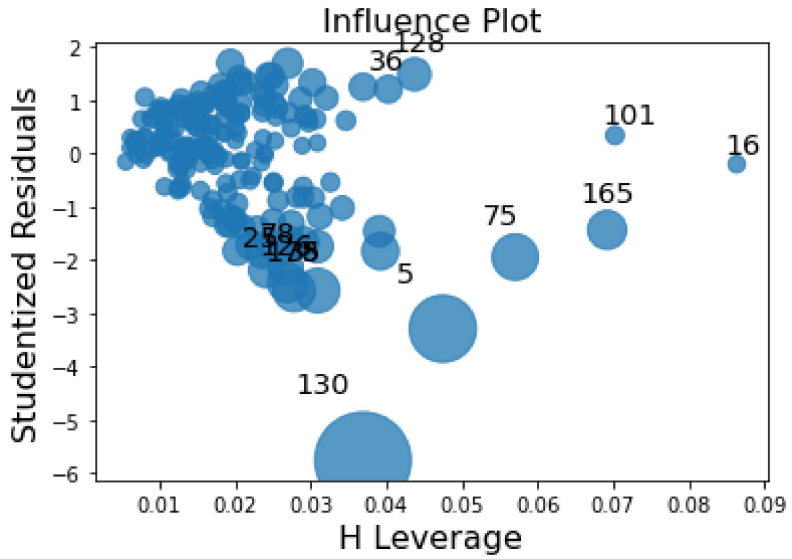
- This model has a higher **R-squared** (0.897) than the previous model, which means that this model provides a better fit to the data than a model that only includes Totalspend.

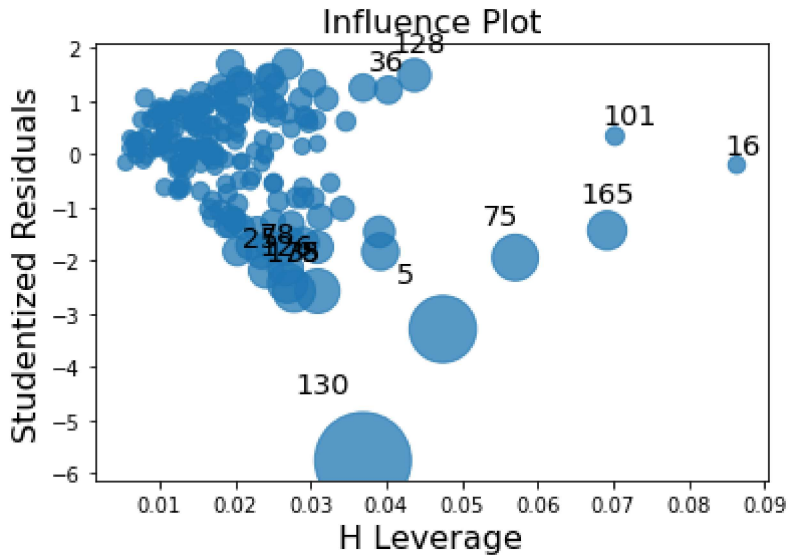
Checking whether data has any influential values

Influence index plots

In [31]:
```
import statsmodels.api as sm
sm.graphics.influence_plot(model1)
```

Out[31]:





- index 130 is showing high influence so we can exclude that entire row
- Studentized Residuals = Residual/standard deviation of residuals

In [32]:
```
df_new=df.drop(df.index[[130]],axis=0)
df_new
```

Out[32]:

	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
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

199 rows × 4 columns

In [33]:
```
lm = smf.ols(formula='sales ~ TV + radio + newspaper', data=df_new).fit()
lm.summary()
```

Out[33]:

OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.910
Model:	OLS	**Adj. R-squared:**	0.908
Method:	Least Squares	**F-statistic:**	653.7
Date:	Mon, 20 Sep 2021	**Prob (F-statistic):**	1.88e-101
Time:	21:06:41	**Log-Likelihood:**	-369.13
No. Observations:	199	**AIC:**	746.3
Df Residuals:	195	**BIC:**	759.4
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.0931	0.290	10.654	0.000	2.520	3.666
TV	0.0448	0.001	34.425	0.000	0.042	0.047
radio	0.1939	0.008	24.130	0.000	0.178	0.210
newspaper	-0.0043	0.005	-0.777	0.438	-0.015	0.007

Omnibus:	21.217	**Durbin-Watson:**	2.157
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	24.630
Skew:	-0.835	**Prob(JB):**	4.48e-06
Kurtosis:	3.423	**Cond. No.**	456.

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

VIF

In [34]:
```
# calculating VIF's value of model
1/(1-0.897)
```

Out[34]:
```
9.708737864077671
```

- if the VIF of model > 4 (threshold value=4)....then there exists a problem and reinvestigate the problem

In [35]:
```
# calculating VIF's values of independent variables
rsq_TV = smf.ols('TV~radio+newspaper',data=df).fit().rsquared
vif_TV = 1/(1-rsq_TV)

rsq_radio = smf.ols('radio~TV+newspaper',data=df).fit().rsquared
vif_radio = 1/(1-rsq_radio)

rsq_newspaper = smf.ols('newspaper~radio+TV',data=df).fit().rsquared
vif_newspaper = 1/(1-rsq_newspaper)
```

In [36]:
```
# Storing vif values in a data frame
d1 = {'Variables':['TV','radio','newspaper'],'VIF':[vif_TV,vif_radio,vif_newspape
Vif_frame = pd.DataFrame(d1)
Vif_frame
```

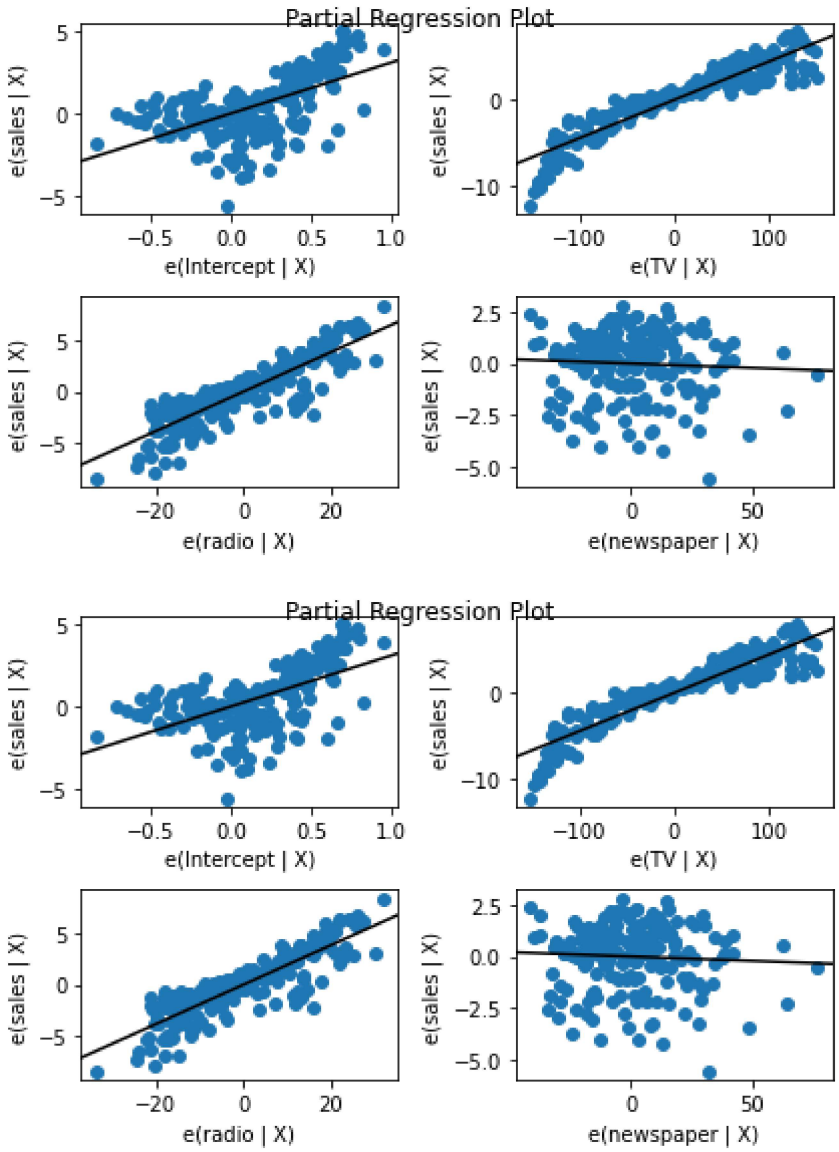
Out[36]:

	Variables	VIF
0	TV	1.004611
1	radio	1.144952
2	newspaper	1.145187

Added variable plot

In [37]:
```
sm.graphics.plot_partregress_grid(lm)
```

Out[37]:



- added varible plot is not showing any significance for newspaper

Feature Selection

How do you decide **what features have to be included** in a linear model? Here's one idea:

- Try different models, and only keep predictors in the model if they have small p-values.

- Check whether the R-squared value goes up when you add new predictors.

What are the **drawbacks** in this approach?

- Linear models rely upon a lot of **assumptions** (such as the features being independent), and if those assumptions are violated (which they usually are), R-squared and p-values are less reliable.
- Using a p-value cutoff of 0.05 means that if you add 100 predictors to a model that are **pure noise**, 5 of them (on average) will still be counted as significant.
- R-squared is susceptible to **overfitting**, and thus there is no guarantee that a model with a high R-squared value will generalize. Below is an example:

R-squared will always increase as you add more features to the model, even if they are unrelated to the response. Thus, selecting the model with the highest R-squared is not a reliable approach for choosing the best linear model.

There is alternative to R-squared called **adjusted R-squared** that penalizes model complexity (to control for overfitting), but it generally under-penalizes complexity (http://scott.fortmann-roe.com/docs/MeasuringError.html).

So is there a better approach to feature selection? **Cross-validation.** It provides a more reliable estimate of out-of-sample error, and thus is a better way to choose which of your models will best **generalize** to out-of-sample data. There is extensive functionality for cross-validation in scikit-learn, including automated methods for searching different sets of parameters and different models. Importantly, cross-validation can be applied to any model, whereas the methods described above only apply to linear models.

In [38]:
```
# only include TV and Radio in the model
lm = smf.ols(formula='sales ~ TV + radio', data=df).fit()
lm.summary()
```

Out[38]:

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, 20 Sep 2021	**Prob (F-statistic):**	4.83e-98
Time:	21:06:44	**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.

Retraining Model on Full Data

If we're satisfied with the performance on the test data, before deploying our model to the real world, we should retrain on all our data. (If we were not satisfied, we could update parameters or choose another model, something we'll discuss later on).

In [39]:
```
final_model = smf.ols(formula='sales ~ TV + radio', data=df).fit()
```

In [40]:
```
final_model.params
```

Out[40]:
```
Intercept    2.921100
TV           0.045755
radio        0.187994
dtype: float64
```

Interpreting the coefficients:

- Holding all other features fixed, a 1 unit (A thousand dollars) increase in TV Spend is associated with an increase in sales of 0.045 "sales units", in this case 1000s of units .
- This basically means that for every $1000 dollars spend on TV Ads, we could expect 45 more units sold.

- Holding all other features fixed, a 1 unit (A thousand dollars) increase in Radio Spend is associated with an increase in sales of 0.188 "sales units", in this case 1000s of units .
- This basically means that for every $1000 dollars spend on Radio Ads, we could expect 188 more units sold.

Note! In this case all our units were the same for each feature (1 unit = $1000 of ad spend). But in other datasets, units may not be the same, such as a housing dataset could try to predict a sale price with both a feature for number of bedrooms and a feature of total area like square footage. In this case it would make more sense to *normalize* the data, in order to clearly compare features and results. We will cover normalization later on.