index

January 24, 2022

1 Ordinary Least Squares in Statsmodels (OLS) - Lab

1.1 Introduction

Previously, you looked at all the requirements for running an OLS simple linear regression using Statsmodels. You worked with the height-weight data set to understand the process and all of the necessary steps that must be performed. In this lab , you'll explore a slightly more complex example to study the impact of spending on different advertising channels on total sales.

1.2 Objectives

You will be able to: * Perform a linear regression using statsmodels * Evaluate a linear regression model by using statistical performance metrics pertaining to overall model and specific parameters * Determine if a particular set of data exhibits the assumptions of linear regression

1.3 Let's get started

In this lab, you'll work with the "Advertising Dataset", which is a very popular dataset for studying simple regression. The dataset is available on Kaggle, but we have downloaded it for you. It is available in this repository as `advertising.csv`. You'll use this dataset to answer this question:

> Which advertising channel has the strongest relationship with sales volume, and can be used to model and predict the sales?

1.4 Step 1: Read the dataset and inspect its columns and 5-point statistics

[1]:
```
# Load necessary libraries and import the data
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


%matplotlib inline
```

[26]:
```
# Check the columns and first few rows
df = pd.read_csv("advertising.csv")
```

[27]:

```python
# Get the 5-point statistics for data
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  200 non-null    int64
 1   TV          200 non-null    float64
 2   radio       200 non-null    float64
 3   newspaper   200 non-null    float64
 4   sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

[30]:

```python
df.drop(columns = "Unnamed: 0", inplace = True)
```

[31]:

```python
# Describe the contents of this dataset
df.describe()
```

[31]:

	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

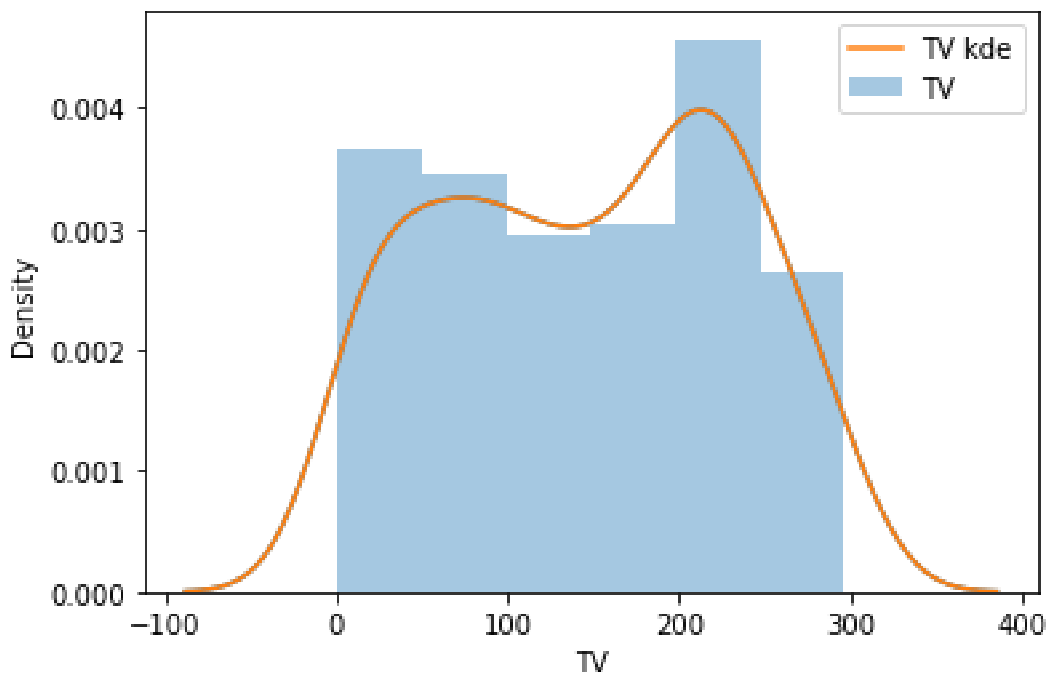
[39]:

```python
## From GitHub

# Describe the contents of this dataset

# In every record, we have three predictors showing the advertising budget
# spent on TV, newspaper and radio and a target variable (sales).

# The target variable shows the sales figure for each marketing campaign
# along with money spent on all three channels.

# Looking at means for predictors, most of the budget is spent on
# TV marketing, and the least is spent on radio.
```

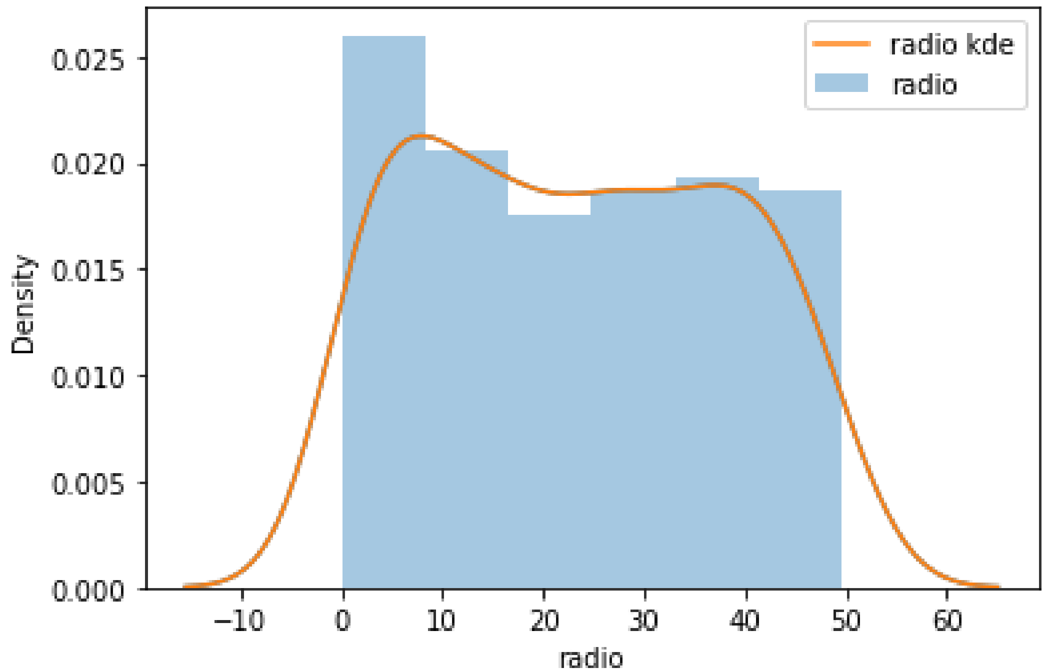
1.5 Step 2: Plot histograms with kde overlay to check the distribution of the predictors

[38]:
```
# For all the variables, check distribution by creating a histogram with kde

for item in df.columns:
    sns.distplot(df[item], label = item)
    sns.kdeplot(df[item], label = f"{item} kde")
    plt.legend()
    plt.show();
```

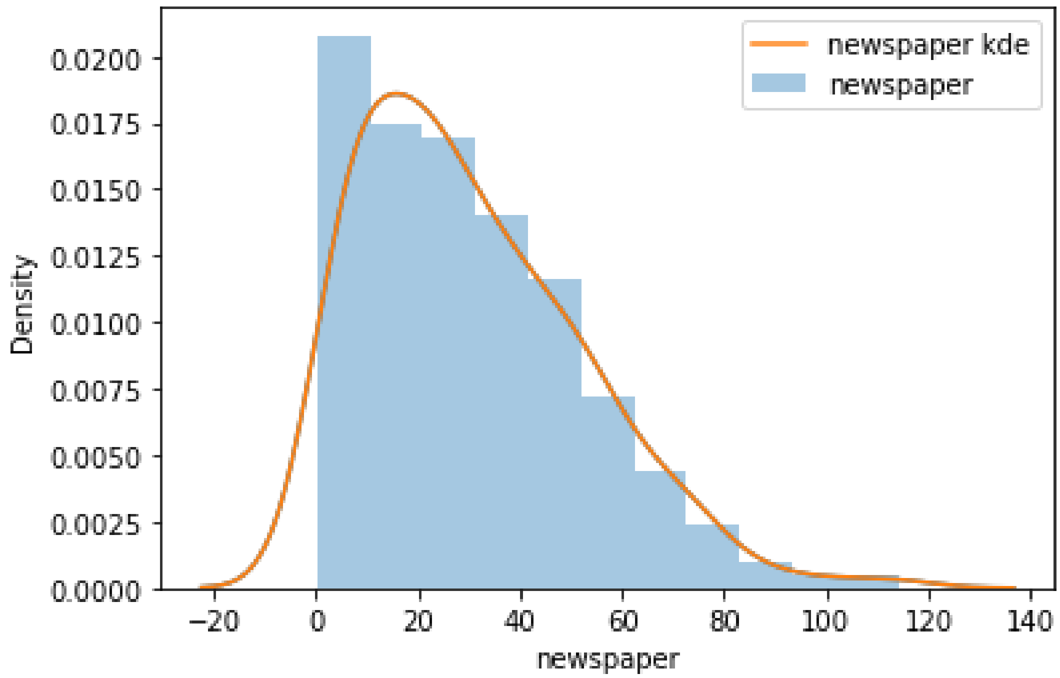
```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



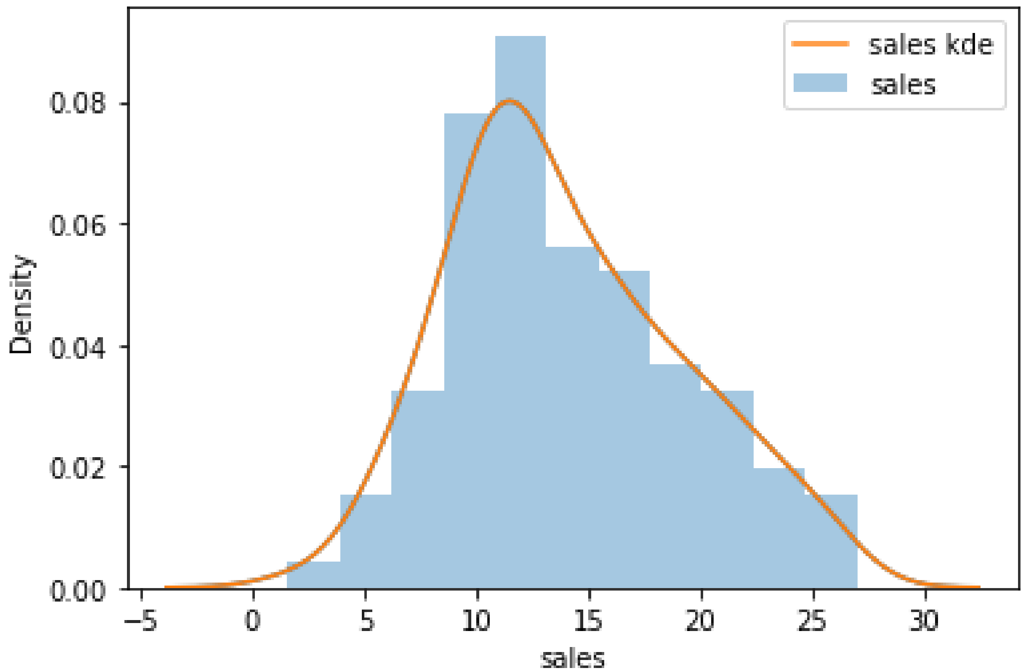

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



[ ]:
```
# Record your observations here

## From GitHub

# Record your observations here

# No variable is "perfectly" normal, but these do tend to follow an overall
# normal pattern.

# We see major skew in the newspaper predictor which could be problematic
# towards analysis.

# TV and radio are still pretty symmetrical distributions and can be used as
# predictors.

# The target variable "sales" is normally distributed with just a gentle skew
```

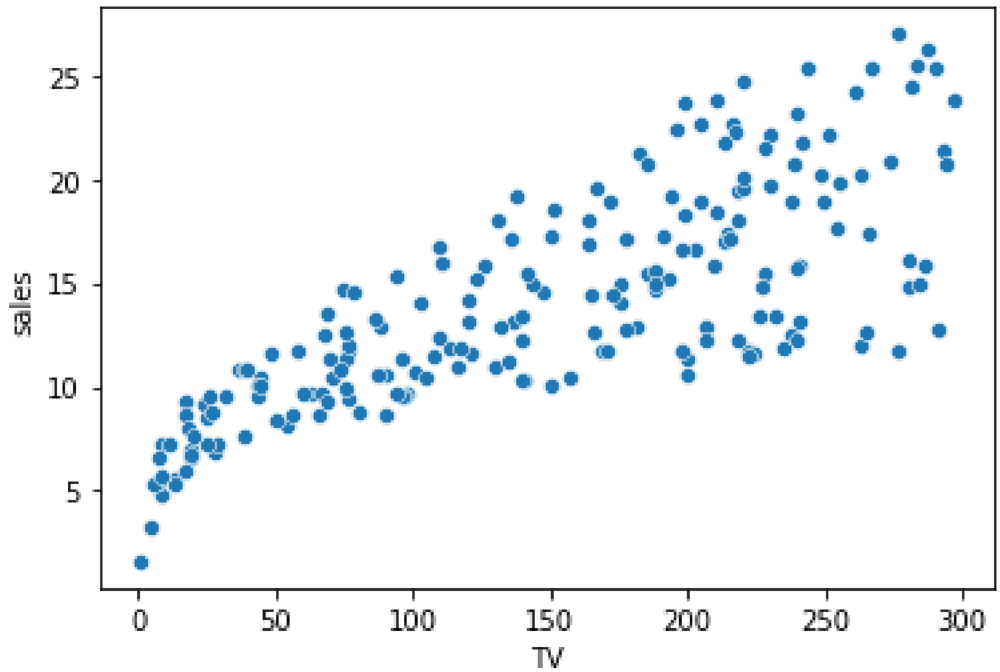
1.6 Step 3: Test for the linearity assumption

Use scatterplots to plot each predictor against the target variable

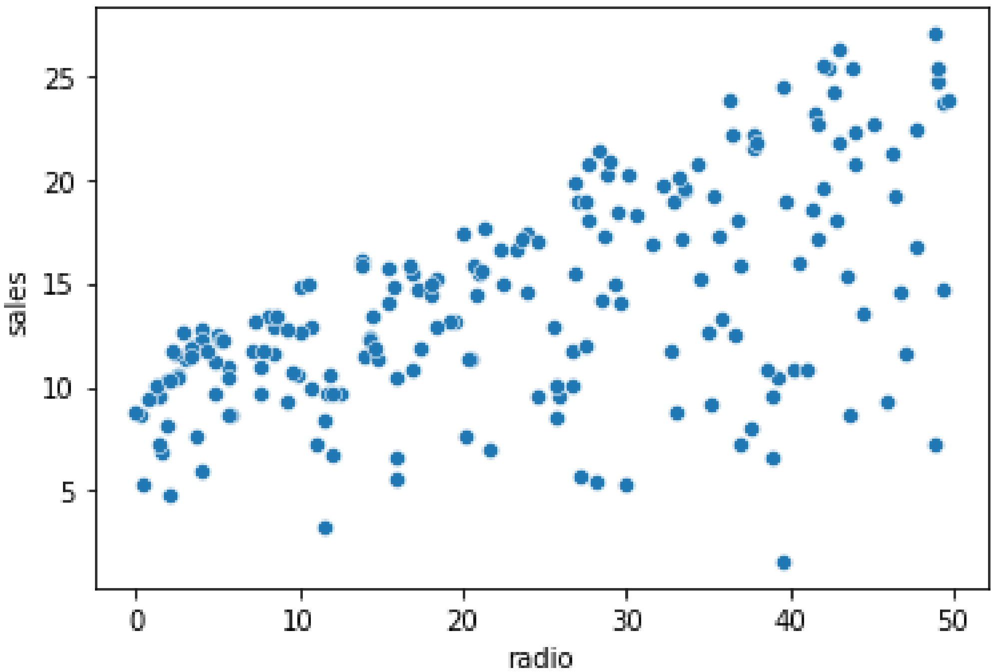
[41]:
```
# visualize the relationship between the preditors and the target using␣
↪scatterplots
for item in df.columns:
    sns.scatterplot(df[item], df["sales"])
```

```
    plt.show();
```

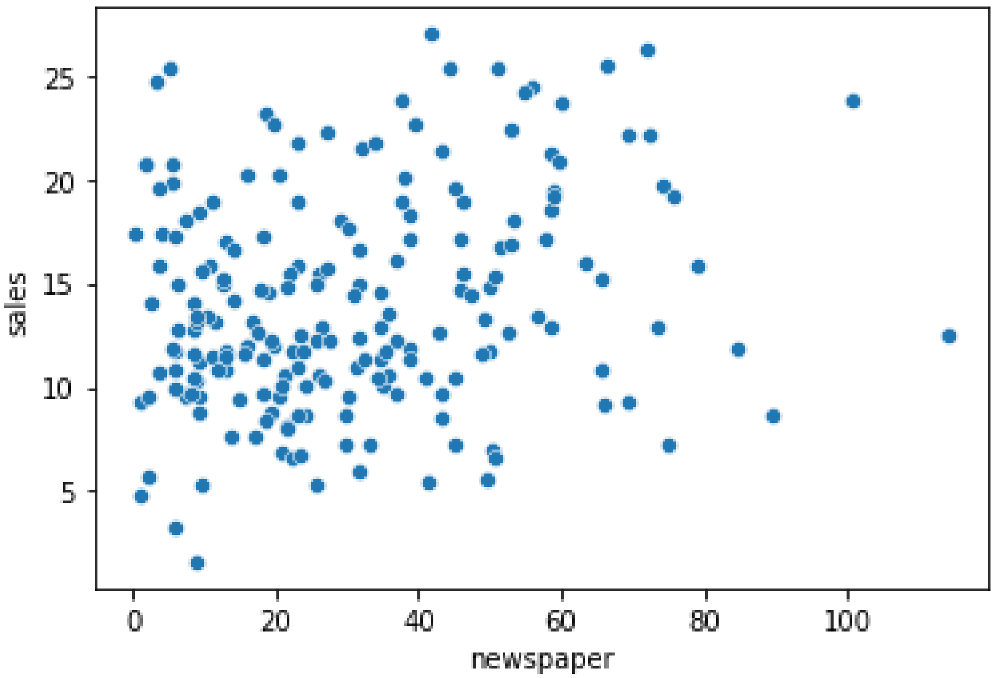
```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



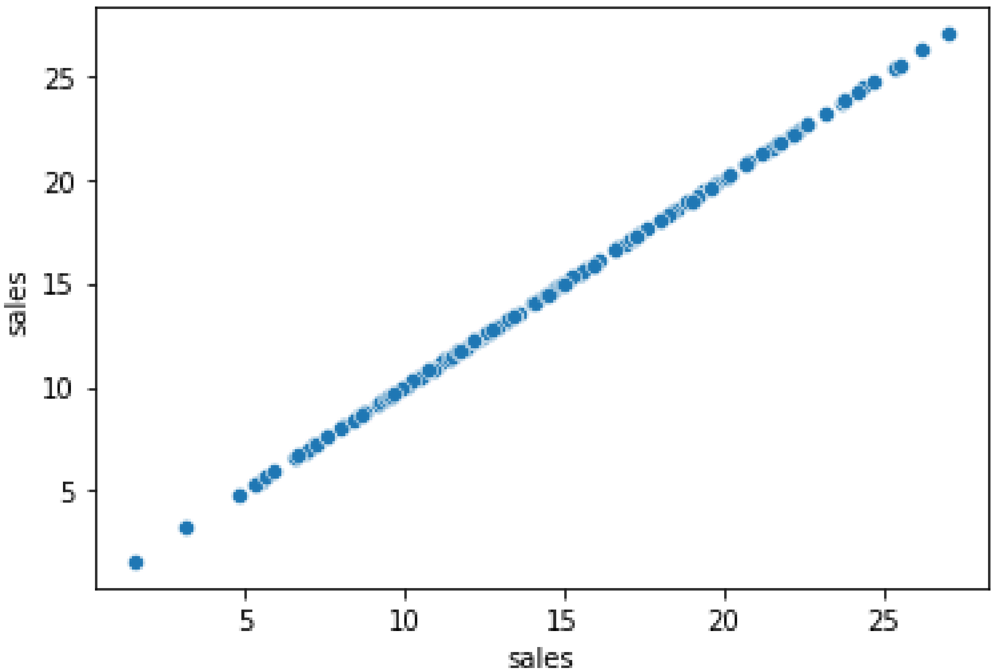

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



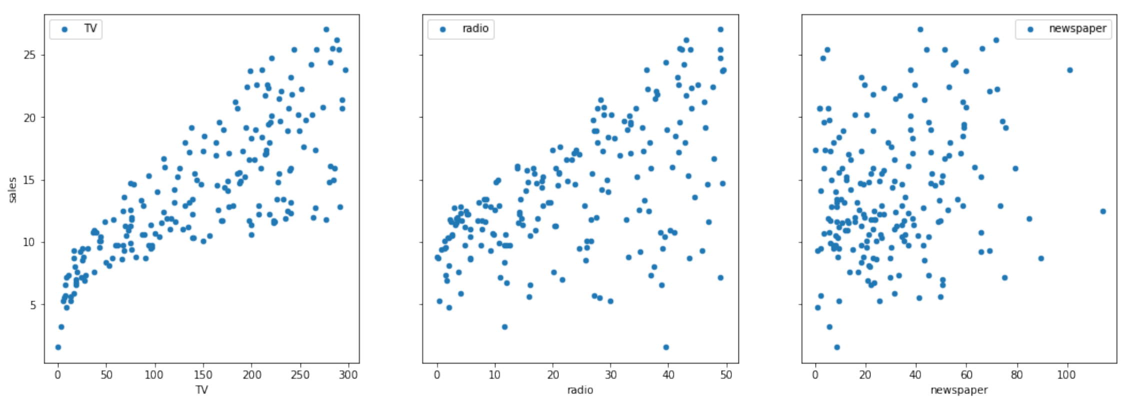

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```




```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



[43]:
```python
### From GitHub:

fig, axs = plt.subplots(1, 3, sharey=True, figsize=(18, 6))
for idx, channel in enumerate(['TV', 'radio', 'newspaper']):
    df.plot(kind='scatter', x=channel, y='sales', ax=axs[idx], label=channel)
plt.legend()
plt.show()
```



[44]:
```python
# Record yor observations on linearity here
```

```
# TV-Sales is very nonlinear but linear model might work even though
# the relation looks nonlinear.

# Radio-Sales is mostly nonlinear but linear model might work too

# newspaper-sales looks linear
```

1.6.1 Conclusion so far

Based on above initial checks, we can confidently say that TV and radio appear to be good predictors for our regression analysis. Newspaper is very heavily skewed and also doesnt show any clear linear relationship with the target. > We'll move ahead with our analysis using TV and radio, and rule out newspaper because we believe it violates OLS assumptions

Note: Kurtosis can be dealt with using techniques like log normalization to "push" the peak towards the center of distribution. You'll learn about this later on.

1.7 Step 4: Run a simple regression in Statsmodels with TV as a predictor

[49]:
```
# import libraries

import statsmodels.api as sm
import statsmodels.formula.api as smf

# build the formula
formula = "sales ~ TV"

# create a fitted model in one line
model = smf.ols(formula = formula, data = df).fit()
```

1.8 Step 5: Get Regression Diagnostics Summary

[50]:
```
model.summary()
```

[50]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            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, 24 Jan 2022   Prob (F-statistic):           1.47e-42
Time:                        01:01:22   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.
==============================================================================

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

[ ]:

Note here that the coefficients represent associations, not causations

1.9 Step 6: Draw a prediction line with data points on a scatter plot for X (TV) and Y (Sales)

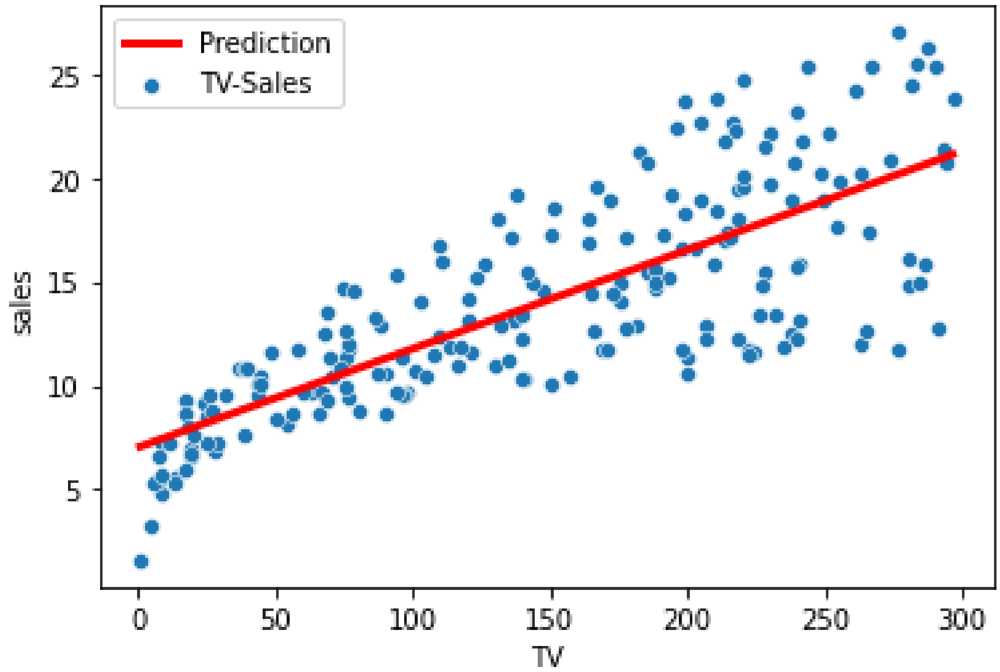
Hint: You can use the `model.predict()` function to predict the start and end point of of regression line for the minimum and maximum values in the 'TV' variable.

[81]:

```python
# create a DataFrame with the minimum and maximum values of TV
tv = pd.DataFrame.from_dict({"TV": [df["TV"].max(), df["TV"].min()]})

# make predictions for those x values and store them
prediction = model.predict(tv)
sns.scatterplot(df["TV"], df["sales"], label = "TV-Sales");
sns.lineplot(tv["TV"], prediction, color = "Red", lw = 3, label = "Prediction")
# first, plot the observed data and the least squares line
```

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

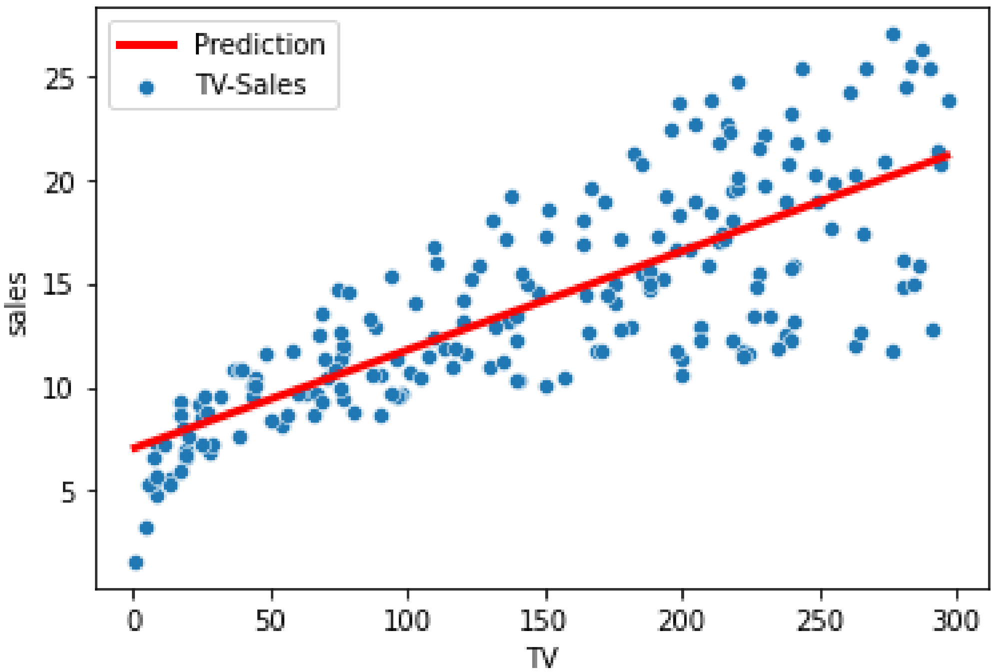
[81]: <AxesSubplot:xlabel='TV', ylabel='sales'>



[96]:
```
tv = df["TV"].to_frame()

# make predictions for those x values and store them
prediction = model.predict(tv)

sns.scatterplot(df["TV"], df["sales"], label = "TV-Sales");
sns.lineplot(tv["TV"], prediction, color = "Red", lw = 3, label = "Prediction")
# first, plot the observed data and the least squares line
```

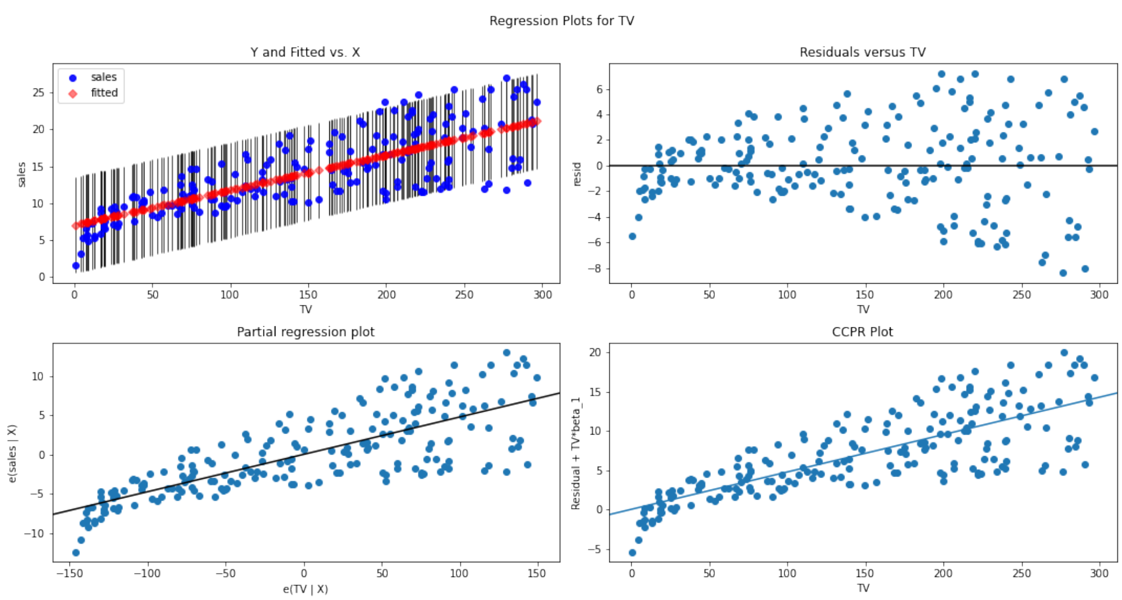
```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

[96]: <AxesSubplot:xlabel='TV', ylabel='sales'>



1.10 Step 7: Visualize the error term for variance and heteroscedasticity

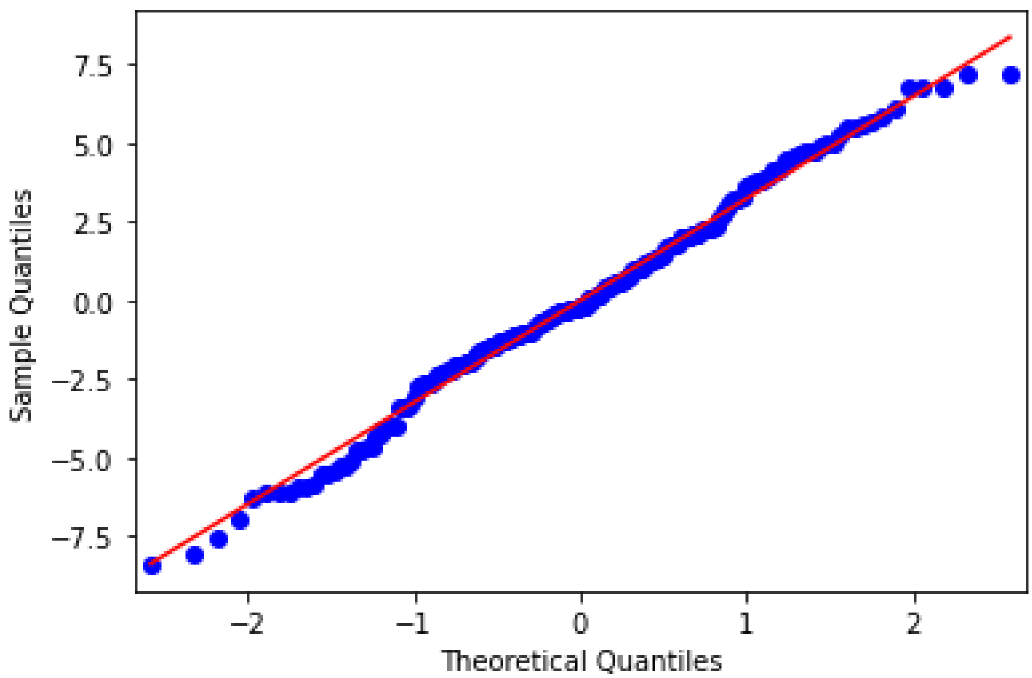
[98]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "TV", fig=fig)
plt.show()
```



[99]:
```
# Record Your observations on heteroscedasticity

# From the first and second plot in the first row, we see a cone-shape
# which is a sign of heteroscedasticity, i.e., the residuals are
# heteroscedastic.

# This violates an assumption.
```

1.11 Step 8: Check the normality assumptions by creating a QQ-plot

[105]:
```
# Code for QQ-plot here
import statsmodels.api as sm
import matplotlib.pyplot as plt

fig = sm.qqplot(model.resid, line = 's')
plt.show()
```

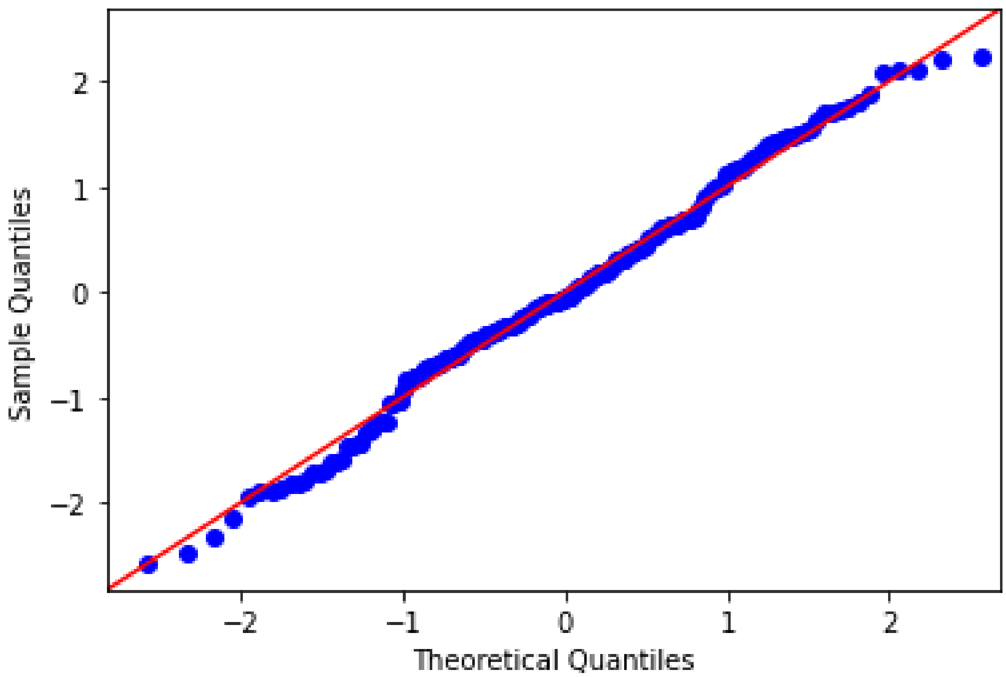


[106]:
```
# Record Your observations on the normality assumption
## QQ Plot shows that the residuals are noremally distributed
```

[107]:
```
## From GitHub Solution

import scipy.stats as stats
residuals = model.resid
```

```
fig = sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True)
fig.show()
```

```
<ipython-input-107-7c8010780ec1>:6: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



1.12 Step 9: Repeat the above for radio and record your observations

[108]:
```
# code for model, prediction line plot, heteroscedasticity check
# and QQ normality check here

formula = "sales ~ radio"
model = model = smf.ols(formula = formula, data = df).fit()
```

[110]:
```
model.summary()
```

[110]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.332
Model:                            OLS   Adj. R-squared:                  0.329
```

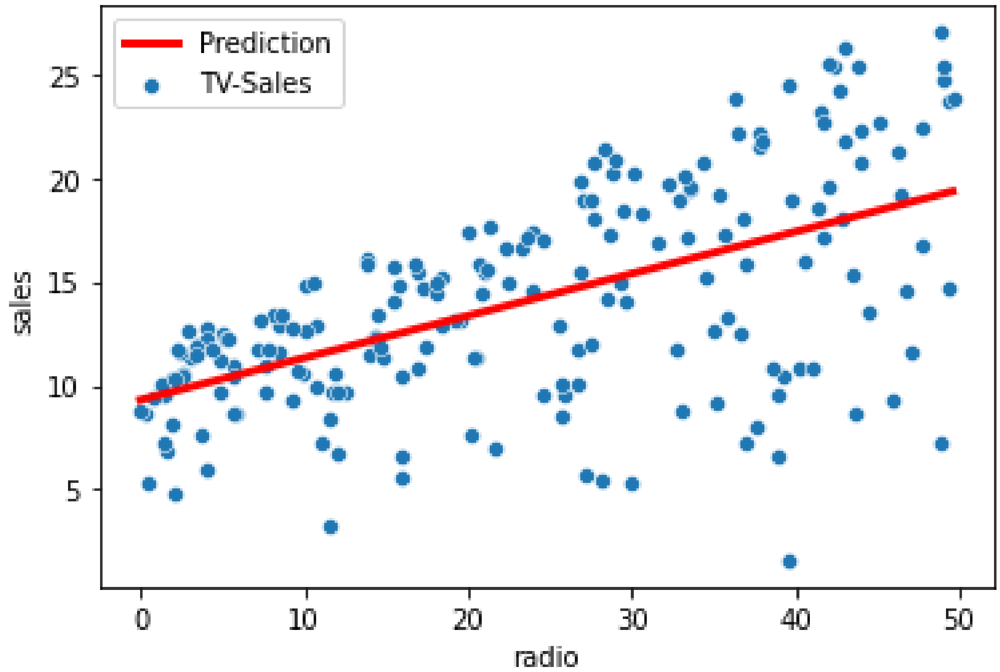
```
Method:                 Least Squares   F-statistic:                     98.42
Date:                Mon, 24 Jan 2022   Prob (F-statistic):           4.35e-19
Time:                        01:26:46   Log-Likelihood:                -573.34
No. Observations:                 200   AIC:                             1151.
Df Residuals:                     198   BIC:                             1157.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.3116      0.563     16.542      0.000       8.202      10.422
radio          0.2025      0.020      9.921      0.000       0.162       0.243
==============================================================================
Omnibus:                       19.358   Durbin-Watson:                   1.946
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               21.910
Skew:                          -0.764   Prob(JB):                     1.75e-05
Kurtosis:                       3.544   Cond. No.                         51.4
==============================================================================

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

[112]:

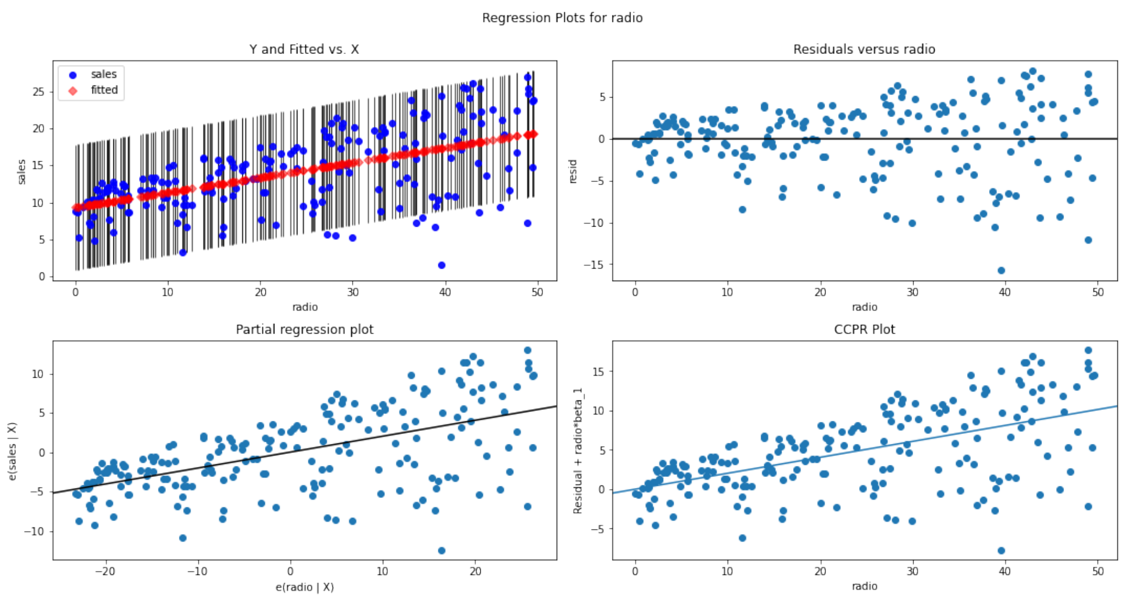
```
radio = df["radio"].to_frame()

# make predictions for those x values and store them
prediction = model.predict(radio)

sns.scatterplot(df["radio"], df["sales"], label = "TV-Sales");
sns.lineplot(radio["radio"], prediction, color = "Red", lw = 3, label =␣
 ↪"Prediction")
# first, plot the observed data and the least squares line
```

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

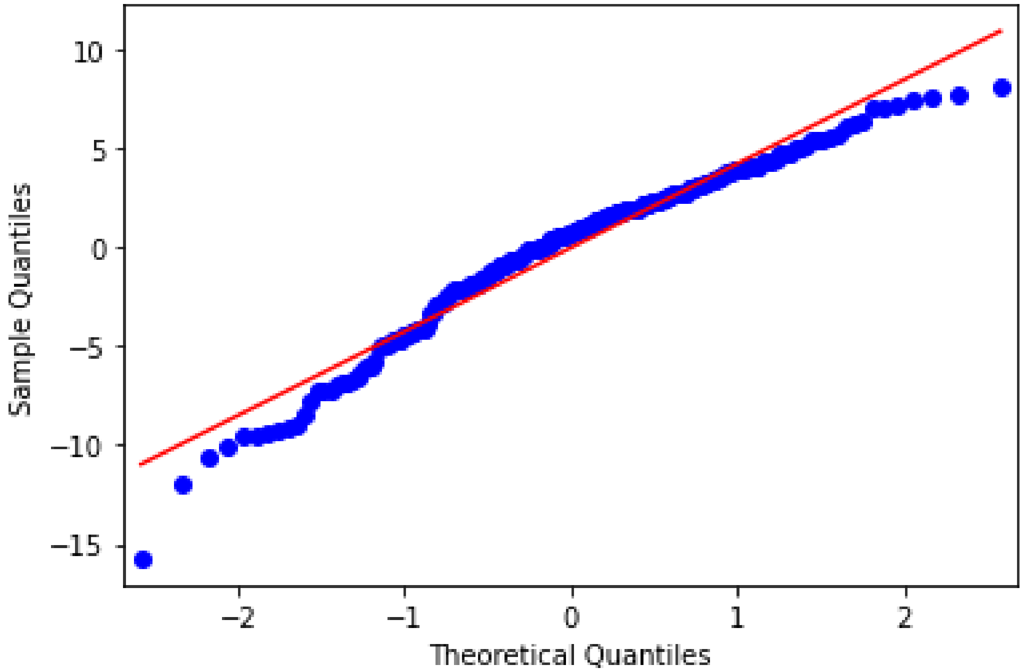
[112]: <AxesSubplot:xlabel='radio', ylabel='sales'>



[114]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "radio", fig=fig)
plt.show()
```



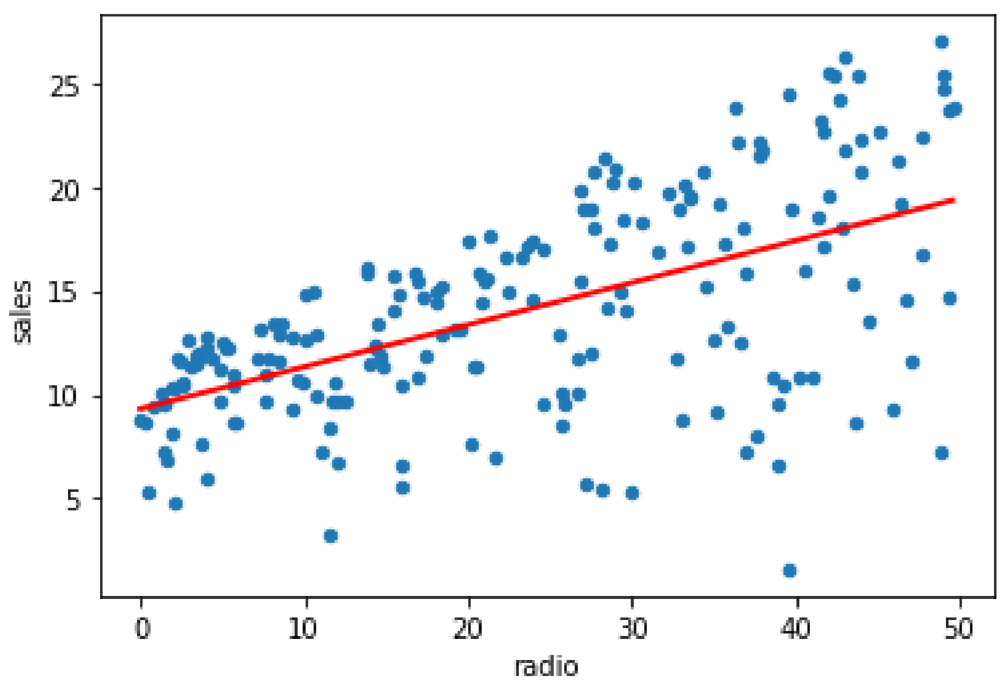
[115]:
```python
import statsmodels.api as sm
import matplotlib.pyplot as plt

fig = sm.qqplot(model.resid, line = 's')
plt.show()
```

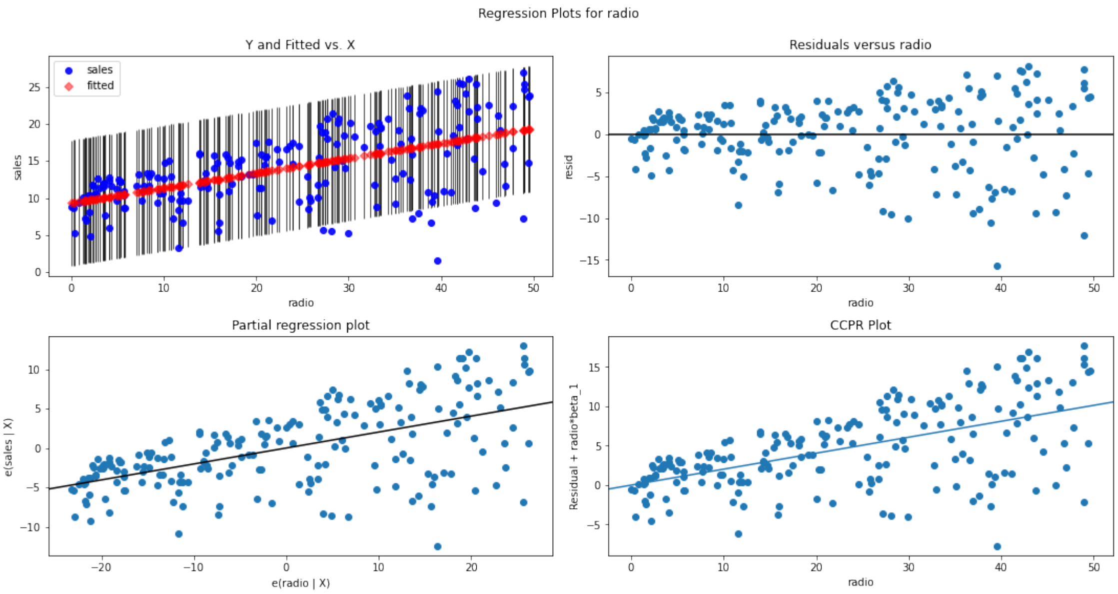


[120]:
```python
# Record your observations here for goodnes of fit

## From GitHub Solution

# Record your observations here for goodnes of fit


# As a predictor, radio performs worse than TV.
# It has higher amount of skewness and kurtosis than TV. After running the
# model, it also became clear that the residuals QQ plot looks off, so the
# normality assumption is not fulfilled.

# A very low R_squared explaining only 33% of variance in the target variable.

# A "unit" increase in radio spending is associated with a 0.2025 "unit"
# increase in Sales. OR An additional 1,000 spent on TV is associated with
# an increase in sales of 202.5

# There is obvious heteroscedasticity as with the case of TV.
```

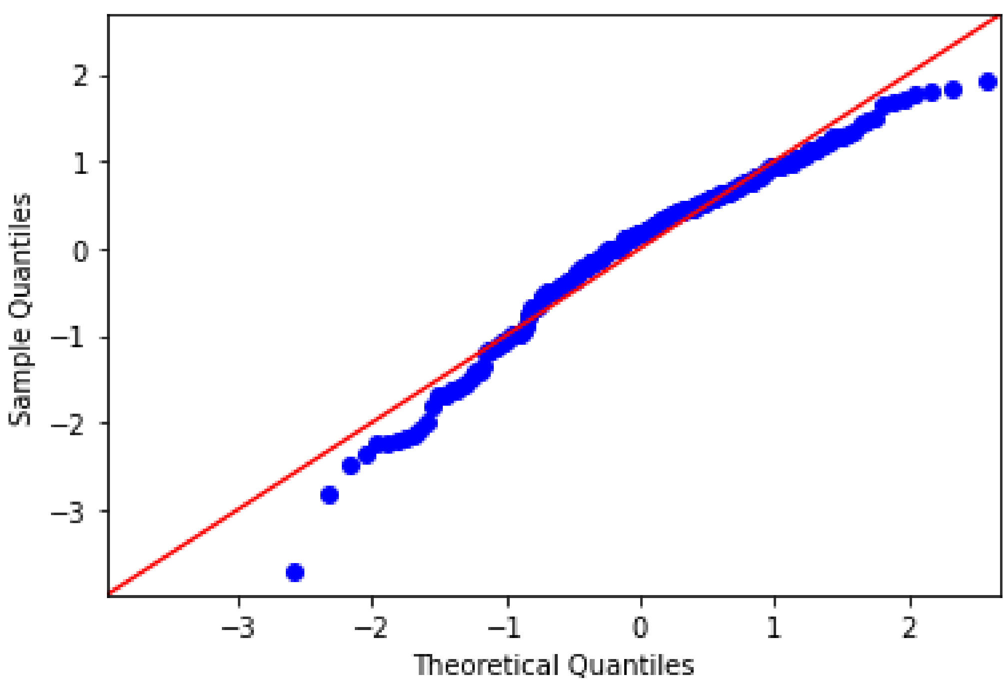
[121]:
```python
### From GitHub Solution

f = 'sales~radio'
model = smf.ols(formula=f, data=df).fit()
print ('R-Squared:',model.rsquared)
print (model.params)
X_new = pd.DataFrame({'radio': [df.radio.min(), df.radio.max()]});
preds = model.predict(X_new)
df.plot(kind='scatter', x='radio', y='sales');
plt.plot(X_new, preds, c='red', linewidth=2);
plt.show()
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "radio", fig=fig)
plt.show()
import scipy.stats as stats
residuals = model.resid
fig = sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True)
fig.show()
```

```
R-Squared: 0.33203245544529547
Intercept    9.311638
radio        0.202496
dtype: float64
```






```
<ipython-input-121-84e2c707cb6c>:18: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



1.13 The Answer

Based on the above analysis, you can conclude that none of the two chosen predictors is ideal for modeling a relationship with the sales volumes. `Newspaper` clearly violated the linearity assumption. `TV` and `radio` did not provide a high value for the coefficient of determination, where TV performed slightly better than the radio. There is obvious heteroscdasticity in the residuals for both variables.

> We can either look for further data, perform extra preprocessing or use more advanced techniques.

Remember there are lots of techniques we can employ to fix these data.

Whether we should call TV the "best predictor" or label all of them "equally useless", is a domain specific question and a marketing manager would have a better opinion on how to move forward with this situation.

In the following lesson, you'll look at the more details on interpreting the regression diagnostics and confidence in the model.

1.14 Summary

In this lab, you ran a complete regression analysis with a simple dataset. You used statsmodel to perform linear regression and evaluated your models using statistical metrics. You also looked for the regression assumptions before and after the analysis phase. Finally, you created some visualizations of your models and checked their goodness of fit.