In [1]:
```
import statsmodels.api as sm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [51]:
```
df=pd.read_csv('C:\\Users\\acile\\Documents\\UVU\\ASpring23\\INFO4130\\Module6\\Advert
```

1. EDA

In [6]:
```
df.shape
```

Out[6]:
```
(200, 4)
```

In [7]:
```
df.head()  # if want to drop a column then df.drop(["columnname"], axis=1, inplace=Tru
```

Out[7]:

	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 [8]:
```
df.isnull().sum() # if wish to drop nulls then df.dropna(inplace=True)
```

Out[8]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
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: 7.8 KB
```

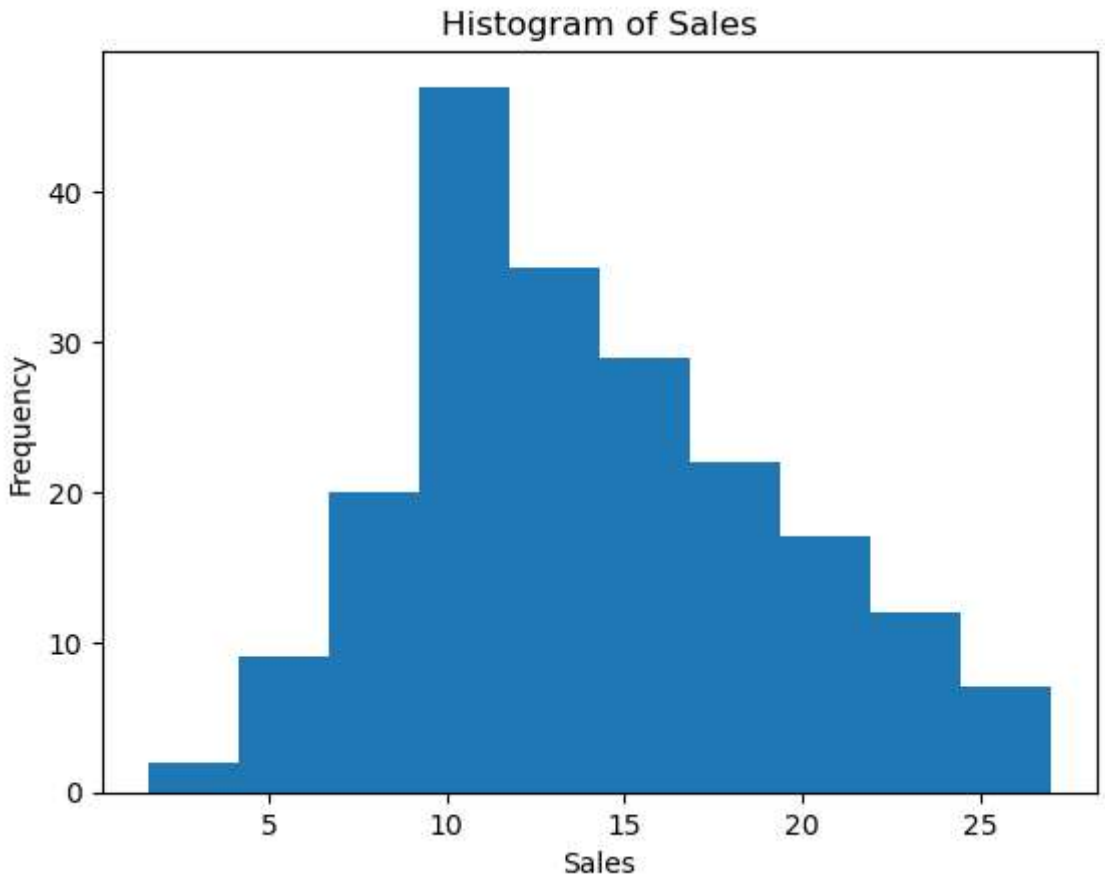
In [10]:
```
df.describe()
```

Out[10]:

	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 [11]:
```
plt.hist(df["sales"]) # makes a histogram of our target variable
plt.xlabel("Sales")
plt.ylabel("Frequency")
plt.title("Histogram of Sales")
```

Out[11]:
```
Text(0.5, 1.0, 'Histogram of Sales')
```



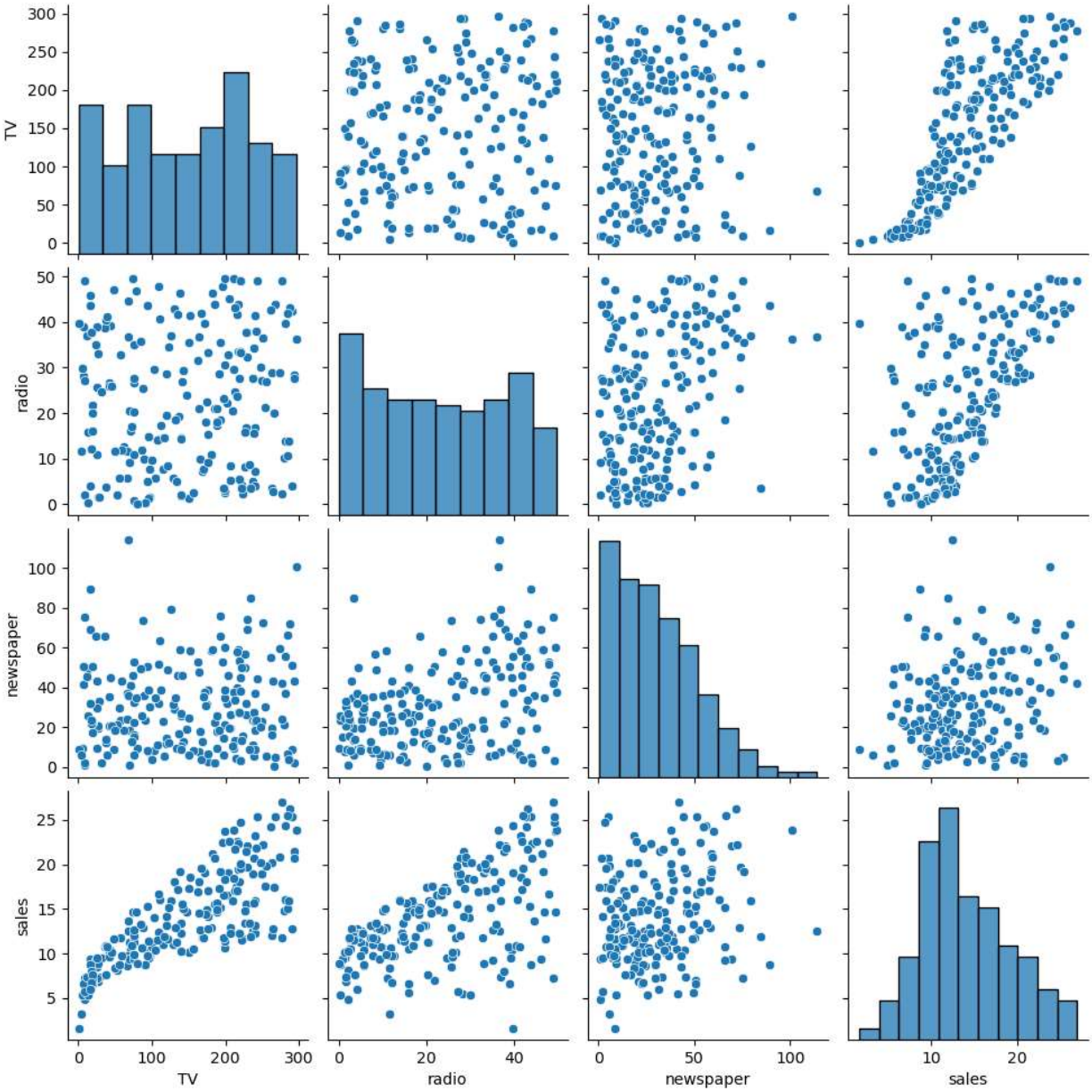
In [12]:
```
df.corr() # makes a correlation matrix (look at corr to target and multicolliniarity w
# https://www.statology.org/multicollinearity-regression/
```

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

In [13]:
```
sns.pairplot(data=df) # makes a scatterplot matrix
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x17cbe2a2670>
```



2. Linear Regression with all data using statmodels

You can analyze each input feature individually (nice exploration but usually skip these steps and do the full model)

In [15]:
```
# https://www.statsmodels.org/stable/examples/notebooks/generated/ols.html
```

In [16]:
```
y=df.sales #outcome or target
x=df.TV #pedictor
x=sm.add_constant(x) #adds a constant term to the predictor
```

In [17]:
```
lrmodel = sm.OLS(y,x).fit()  # or lrmodel = sm.OLS('y~x').fit() if using R-style code
```

In [18]:
```
print(lrmodel.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:                Sat, 25 Feb 2023   Prob (F-statistic):           1.47e-42
Time:                        17:49:20   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]
------------------------------------------------------------------------------
const          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 spec
ified.
```

In [19]:
```
print(lrmodel.params)
```

```
const    7.032594
TV       0.047537
dtype: float64
```

In [20]:
```
print("R2: ", lrmodel.rsquared)
```

```
R2:  0.611875050850071
```

In [21]:
```
r=df.radio
r=sm.add_constant(r) #adds a constant term to the predictor
rlrmodel=sm.OLS(y,r).fit()
print(rlrmodel.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:                Sat, 25 Feb 2023   Prob (F-statistic):           4.35e-19
Time:                        17:49:41   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]
------------------------------------------------------------------------------
const          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 spec
ified.
```

In [22]:
```
n=df.newspaper
n=sm.add_constant(n) #adds a constant term to the predictor
nlrmodel=sm.OLS(y,n).fit()
print(nlrmodel.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.052
Model:                            OLS   Adj. R-squared:                  0.047
Method:                 Least Squares   F-statistic:                     10.89
Date:                Sat, 25 Feb 2023   Prob (F-statistic):            0.00115
Time:                        17:49:47   Log-Likelihood:                -608.34
No. Observations:                 200   AIC:                             1221.
Df Residuals:                     198   BIC:                             1227.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.3514      0.621     19.876      0.000      11.126      13.577
newspaper      0.0547      0.017      3.300      0.001       0.022       0.087
==============================================================================
Omnibus:                        6.231   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.044   Jarque-Bera (JB):                5.483
Skew:                           0.330   Prob(JB):                       0.0645
Kurtosis:                       2.527   Cond. No.                         64.7
==============================================================================

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

Create full model with all 3 predictors

In [23]:
```
y=df.sales #outcome or target
x=df[['TV','radio', 'newspaper']] #pedictor
x=sm.add_constant(x) #adds a constant term to the predictor
```

In [24]:
```
lrmodel = sm.OLS(y,x).fit()
print(lrmodel.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     570.3
Date:                Sat, 25 Feb 2023   Prob (F-statistic):           1.58e-96
Time:                        17:50:10   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]
------------------------------------------------------------------------------
const          2.9389      0.312      9.422      0.000       2.324       3.554
TV             0.0458      0.001     32.809      0.000       0.043       0.049
radio          0.1885      0.009     21.893      0.000       0.172       0.206
newspaper     -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.
==============================================================================

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

drop newspaper with bad p-value of .86, create new model

In [26]:
```
y=df.sales #outcome or target
x=df[['TV','radio']] #pedictor
x=sm.add_constant(x) #adds a constant term to the predictor
```

In [27]:
```
lrmodel = sm.OLS(y,x).fit()
print(lrmodel.summary())
```

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:	Sat, 25 Feb 2023	Prob (F-statistic):	4.83e-98
Time:	17:50:37	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]
const	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.

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

Model looks good: R2 is large at 89%, F is large (not 0), the p-values for TV and Radio are below .01.

Formula: is Sales = 2.9211 + TV x .0458 + Radio x .188 *** This formula is what your client is paying you for.

Weaknesses: the data are a bit not normally distributed (but not terrible) - skew should be 0 but is -1.323 thus is negatively skewed (left tail long) Kurtosis should be 3 but is 6.2 which indicates a "heavy-tailed" distribution, which indicates some outliers

Measure of fit performance (we will use RMSE)

In [29]:
```
from statsmodels.tools.eval_measures import rmse
```

In [30]:
```
ypredLR = lrmodel.predict(x)
```

In [31]:
```
rmse(y,ypredLR) #RMSE Root Mean Squared Error
```

Out[31]:
```
1.668703059366193
```

3. Regression Tree using all data using sklearn DecisionTreeRegressor

In [32]:
```
# documentation at https://scikit-learn.org/stable/modules/generated/sklearn.tree.Deci
```

In [33]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [34]:
```
regtreemodel2 = DecisionTreeRegressor(criterion = "squared_error", max_depth = 4, min_
```

In [35]:
```
y=df.sales #outcome or target
x=df[['TV','radio','newspaper']] #pedictor
```

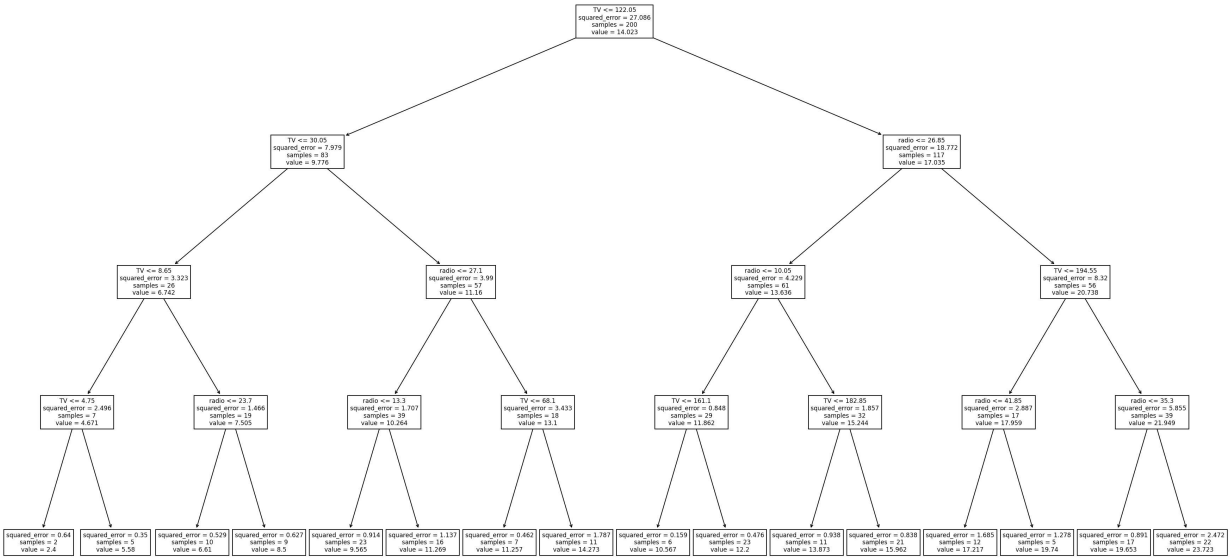
In [36]:
```
regtreemodel2.fit(x,y)
```

Out[36]:
```
DecisionTreeRegressor(max_depth=4, min_samples_split=5, random_state=5)
```

visualize in tree form

In [37]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(30,16), dpi=150)
plot_tree(regtreemodel2, feature_names=x.columns) # true is to the left on the tree, f
```

Out[37]:
```
[Text(0.5, 0.9, 'TV <= 122.05\nsquared_error = 27.086\nsamples = 200\nvalue = 14.02
3'),
 Text(0.25, 0.7, 'TV <= 30.05\nsquared_error = 7.979\nsamples = 83\nvalue = 9.776'),
 Text(0.125, 0.5, 'TV <= 8.65\nsquared_error = 3.323\nsamples = 26\nvalue = 6.742'),
 Text(0.0625, 0.3, 'TV <= 4.75\nsquared_error = 2.496\nsamples = 7\nvalue = 4.671'),
 Text(0.03125, 0.1, 'squared_error = 0.64\nsamples = 2\nvalue = 2.4'),
 Text(0.09375, 0.1, 'squared_error = 0.35\nsamples = 5\nvalue = 5.58'),
 Text(0.1875, 0.3, 'radio <= 23.7\nsquared_error = 1.466\nsamples = 19\nvalue = 7.50
5'),
 Text(0.15625, 0.1, 'squared_error = 0.529\nsamples = 10\nvalue = 6.61'),
 Text(0.21875, 0.1, 'squared_error = 0.627\nsamples = 9\nvalue = 8.5'),
 Text(0.375, 0.5, 'radio <= 27.1\nsquared_error = 3.99\nsamples = 57\nvalue = 11.1
6'),
 Text(0.3125, 0.3, 'radio <= 13.3\nsquared_error = 1.707\nsamples = 39\nvalue = 10.26
4'),
 Text(0.28125, 0.1, 'squared_error = 0.914\nsamples = 23\nvalue = 9.565'),
 Text(0.34375, 0.1, 'squared_error = 1.137\nsamples = 16\nvalue = 11.269'),
 Text(0.4375, 0.3, 'TV <= 68.1\nsquared_error = 3.433\nsamples = 18\nvalue = 13.1'),
 Text(0.40625, 0.1, 'squared_error = 0.462\nsamples = 7\nvalue = 11.257'),
 Text(0.46875, 0.1, 'squared_error = 1.787\nsamples = 11\nvalue = 14.273'),
 Text(0.75, 0.7, 'radio <= 26.85\nsquared_error = 18.772\nsamples = 117\nvalue = 17.0
35'),
 Text(0.625, 0.5, 'radio <= 10.05\nsquared_error = 4.229\nsamples = 61\nvalue = 13.63
6'),
 Text(0.5625, 0.3, 'TV <= 161.1\nsquared_error = 0.848\nsamples = 29\nvalue = 11.86
2'),
 Text(0.53125, 0.1, 'squared_error = 0.159\nsamples = 6\nvalue = 10.567'),
 Text(0.59375, 0.1, 'squared_error = 0.476\nsamples = 23\nvalue = 12.2'),
 Text(0.6875, 0.3, 'TV <= 182.85\nsquared_error = 1.857\nsamples = 32\nvalue = 15.24
4'),
 Text(0.65625, 0.1, 'squared_error = 0.938\nsamples = 11\nvalue = 13.873'),
 Text(0.71875, 0.1, 'squared_error = 0.838\nsamples = 21\nvalue = 15.962'),
 Text(0.875, 0.5, 'TV <= 194.55\nsquared_error = 8.32\nsamples = 56\nvalue = 20.73
8'),
 Text(0.8125, 0.3, 'radio <= 41.85\nsquared_error = 2.887\nsamples = 17\nvalue = 17.9
59'),
 Text(0.78125, 0.1, 'squared_error = 1.685\nsamples = 12\nvalue = 17.217'),
 Text(0.84375, 0.1, 'squared_error = 1.278\nsamples = 5\nvalue = 19.74'),
 Text(0.9375, 0.3, 'radio <= 35.3\nsquared_error = 5.855\nsamples = 39\nvalue = 21.94
9'),
 Text(0.90625, 0.1, 'squared_error = 0.891\nsamples = 17\nvalue = 19.653'),
 Text(0.96875, 0.1, 'squared_error = 2.472\nsamples = 22\nvalue = 23.723')]
```



visualize in text form

In [38]:
```
from sklearn import tree
print(tree.export_text(regtreemodel2, feature_names = ['TV','radio', 'newspaper']))
```

```
|--- TV <= 122.05
|   |--- TV <= 30.05
|   |   |--- TV <= 8.65
|   |   |   |--- TV <= 4.75
|   |   |   |   |--- value: [2.40]
|   |   |   |--- TV >  4.75
|   |   |   |   |--- value: [5.58]
|   |   |--- TV >  8.65
|   |   |   |--- radio <= 23.70
|   |   |   |   |--- value: [6.61]
|   |   |   |--- radio >  23.70
|   |   |   |   |--- value: [8.50]
|   |--- TV >  30.05
|   |   |--- radio <= 27.10
|   |   |   |--- radio <= 13.30
|   |   |   |   |--- value: [9.57]
|   |   |   |--- radio >  13.30
|   |   |   |   |--- value: [11.27]
|   |   |--- radio >  27.10
|   |   |   |--- TV <= 68.10
|   |   |   |   |--- value: [11.26]
|   |   |   |--- TV >  68.10
|   |   |   |   |--- value: [14.27]
|--- TV >  122.05
|   |--- radio <= 26.85
|   |   |--- radio <= 10.05
|   |   |   |--- TV <= 161.10
|   |   |   |   |--- value: [10.57]
|   |   |   |--- TV >  161.10
|   |   |   |   |--- value: [12.20]
|   |   |--- radio >  10.05
|   |   |   |--- TV <= 182.85
|   |   |   |   |--- value: [13.87]
|   |   |   |--- TV >  182.85
|   |   |   |   |--- value: [15.96]
|   |--- radio >  26.85
|   |   |--- TV <= 194.55
|   |   |   |--- radio <= 41.85
|   |   |   |   |--- value: [17.22]
|   |   |   |--- radio >  41.85
|   |   |   |   |--- value: [19.74]
|   |   |--- TV >  194.55
|   |   |   |--- radio <= 35.30
|   |   |   |   |--- value: [19.65]
|   |   |   |--- radio >  35.30
|   |   |   |   |--- value: [23.72]
```

more on tree visualization
https://mljar.com/blog/visualize-decision-tree/

Measure of fit performance (we will use RMSE)

In [46]:
```
y_predRT = regtreemodel2.predict(x)
```

In [47]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y, y_predRT)
```

Out[47]:
```
1.0599842062407965
```

In [48]:
```
np.sqrt(mean_squared_error(y, y_predRT))  #RMSE, closer to zero is better
```

Out[48]:
```
1.029555343942615
```

In [49]:
```
from sklearn.model_selection import cross_val_score
cross_val_score(regtreemodel2, x, y, cv=10)
```

Out[49]:
```
array([0.94304499, 0.90861469, 0.95206686, 0.92055261, 0.84629713,
       0.86857687, 0.92412929, 0.85356716, 0.91351403, 0.9506752 ])
```

In [50]:
```
cvscores = cross_val_score(regtreemodel2, x, y, cv=10)
np.average(cvscores)
```

Out[50]:
```
0.9081038838616384
```