In [53]:
```
# Import the numpy and pandas package
import numpy as np
import pandas as pd

# Read the given CSV file, and view some sample records
advertising = pd.read_csv("Company_data.csv")
advertising
```

Out[53]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [93]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.linear_model import LogisticRegression,LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import load_iris
from sklearn.model_selection import GridSearchCV

from sklearn.datasets import load_digits
from sklearn.model_selection import cross_val_score
```

In [54]:
```
# Shape of our dataset
advertising.shape

# Info our dataset
advertising.info()
```

```
# Describe our dataset
advertising.describe()
```

```
<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
```

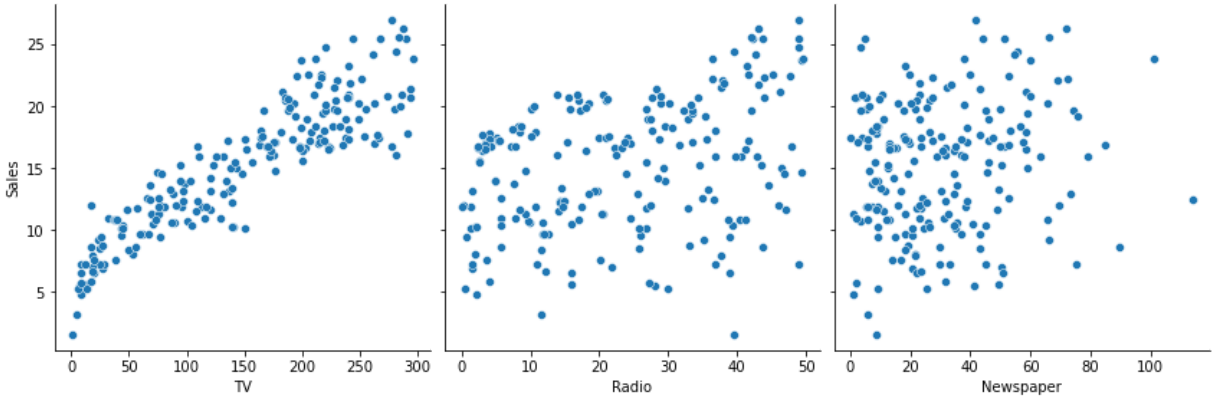
Out[54]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [55]:

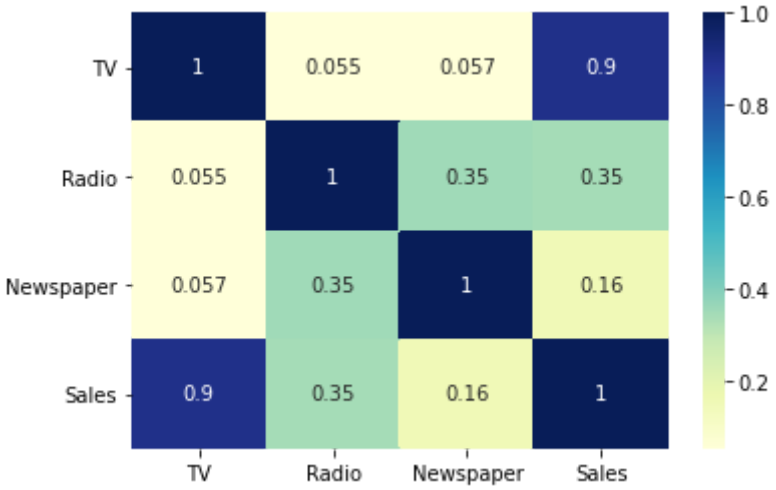
```
# Import matplotlib and seaborn libraries to visualize the data
import matplotlib.pyplot as plt
import seaborn as sns

# Using pairplot we'll visualize the data for correlation
sns.pairplot(advertising, x_vars=['TV', 'Radio','Newspaper'],
             y_vars='Sales', size=4, aspect=1, kind='scatter')
plt.show()
```

```
C:\Users\Prakhar\anaconda3new\lib\site-packages\seaborn\axisgrid.py:1969: UserWarnin
g: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



In [56]:

```
# Visualizing the data using heatmap
sns.heatmap(advertising.corr(), cmap="YlGnBu", annot = True)
plt.show()
```



In [57]:
```
# Creating X and y
X = advertising['TV']
y = advertising['Sales']
```

In [70]:
```
# Splitting the varaibles as training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8,
                                                    test_size = 0.2, random_state =
```

In [71]:
```
# Splitting the data into train and test
from sklearn.model_selection import train_test_split
X_train_lm, X_test_lm, y_train_lm, y_test_lm = train_test_split(X, y, train_size = 0
                                                                test_size = 0.2, ran
```

In [72]:
```
# Shape of the train set without adding column
X_train_lm.shape

# Adding additional column to the train and test data
X_train_lm = X_train_lm.values.reshape(-1,1)
X_test_lm = X_test_lm.values.reshape(-1,1)

print(X_train_lm.shape)
print(X_test_lm.shape)
```

```
(160, 1)
(40, 1)
```

In [73]:
```
# Shape of the train set without adding column
X_train_lm.shape

# Adding additional column to the train and test data
X_train_lm = X_train_lm.reshape(-1,1)
X_test_lm = X_test_lm.reshape(-1,1)

print(X_train_lm.shape)
print(X_test_lm.shape)
```

```
(160, 1)
(40, 1)
```

In [88]:
```
from sklearn.linear_model import LinearRegression
```

```
# Creating an object of Linear Regression
lm = LinearRegression()

# Fit the model using .fit() method
lm.fit(X_train_lm, y_train_lm)
lm.score(X_test_lm,y_test_lm)
```

Out[88]: 0.7281352744078883

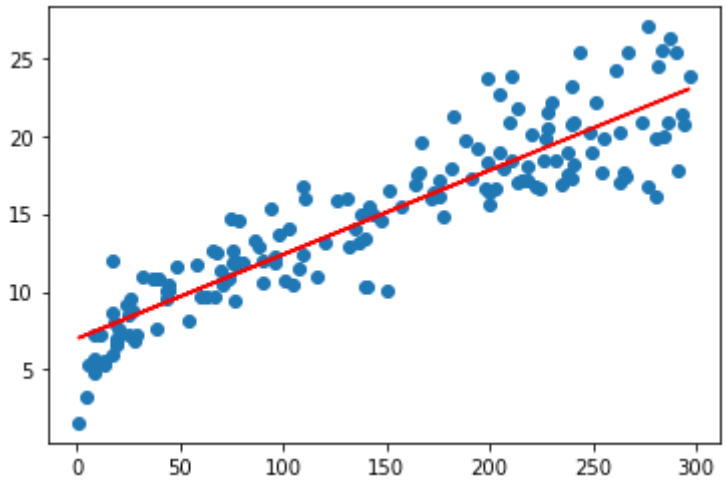
In [75]:
```
# Intercept value
print("Intercept :",lm.intercept_)

# Slope value
print('Slope :',lm.coef_)
```

```
Intercept : 6.995532914307688
Slope : [0.05410548]
```

In [76]:
```
# Making Predictions of y_value
y_train_pred = lm.predict(X_train_lm)
y_test_pred = lm.predict(X_test_lm)

# Comparing the r2 value of both train and test data
print(r2_score(y_train,y_train_pred))
print(r2_score(y_test,y_test_pred))
```

```
0.8216142794949134
0.7281352744078883
```

In [77]:
```
# Visualizing the regression line
plt.scatter(X_train, y_train)
plt.plot(X_train, 6.995 + 0.054*X_train, 'r')
plt.show()
```



In [ ]:
```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
model_params={
    'linear':{
        'model':LinearRegression(),
            'params': {
            'normalize':[True,False]
              }
          },
          'lasso':{
```

```
        'model':linear_model.Lasso(),
            'params':{
            'alpha':[1,2],
            'selection':['random','cyclic']

                  }
           },
          'dec_tree':{
        'model':DecisionTreeRegressor(),
            'params':{
            'criterion':['mse','friedman_mse'],
            'splitter':['best', 'random'],
            'C': [1,5,10]
                  }
            }

}


scores=[]
cv=ShuffleSplit(n_splits=5,test_size=0.2,random_state=0)
for model_name,mp in model_params.items():
  clf=GridSearchCV(mp['model'],mp['params'],cv=cv)
  clf.fit(X_test_lm,y_test_lm)
  scores.append({'model':model_name,'best_score':clf.best_score_,'best_params':clf.b


scores
```

In [ ]:

```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(clf, X_test_lm,y_test_lm, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores.mean())
```