In [1]:
```
# Problem Statement- Build a model which predicts sales based on the money spent
on different platforms for marketing.
```

In [ ]:
```
# We will import the libraies which will be using in building prediction  model
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
# We will import the data Advertising from Github.
#The case study of Sales channel includes the detailed study of TV, radio and ne
wspaper channel
```

In [4]:
```
df=pd.read_csv('https://raw.githubusercontent.com/dsrscientist/DSData/master/Adv
ertising.csv',index_col=0)
```

In [ ]:
```
# Data Inspection
```

In [5]:
```
df.head()
```

Out[5]:

| | 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 [7]:
```
df.shape
```

Out[7]:
```
(200, 4)
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
TV           200 non-null float64
radio        200 non-null float64
newspaper    200 non-null float64
sales        200 non-null float64
dtypes: float64(4)
memory usage: 7.8 KB
```

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

Out[9]:

| | 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 [ ]:
```
#Data Cleaning¶
```

In [15]:
```
# To  find missing values
df.isnull().sum()
```

Out[15]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [57]:
```
     # No Null values found
```

In [14]:
```
# Check data type
df.dtypes
```

Out[14]:
```
TV           float64
radio        float64
newspaper    float64
sales        float64
dtype: object
```

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

Out[12]:

| | 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 |

To Check Cooreltion

In [17]:
```
dfcorr=df.corr()
```

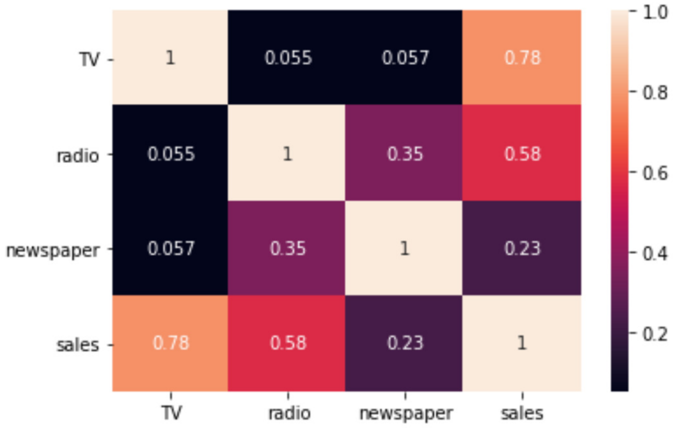
In [18]:
```
dfcorr
```

Out[18]:

| | 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 [19]:
```
sns.heatmap(dfcorr,annot=True)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x200f3d284e0>

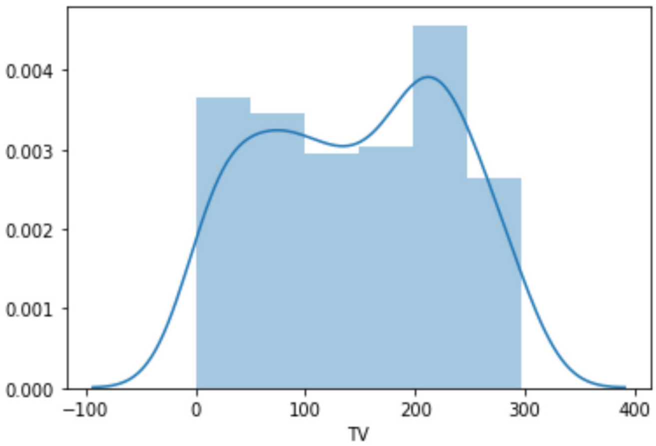



In [20]:
```
# Tv advertisement has high impact on sales whereas newspaper has lowest.
```

In [ ]:
```
#Exploratory Data Analysis
```

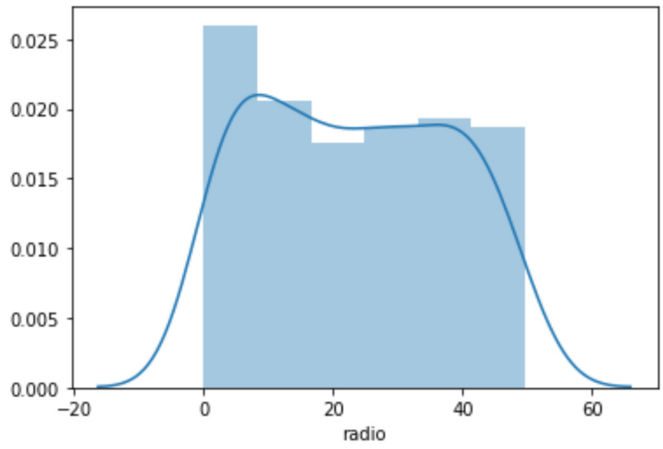
In [21]:
```
# To check distribution of skewness
sns.distplot(df['TV'])
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x200f3e422b0>



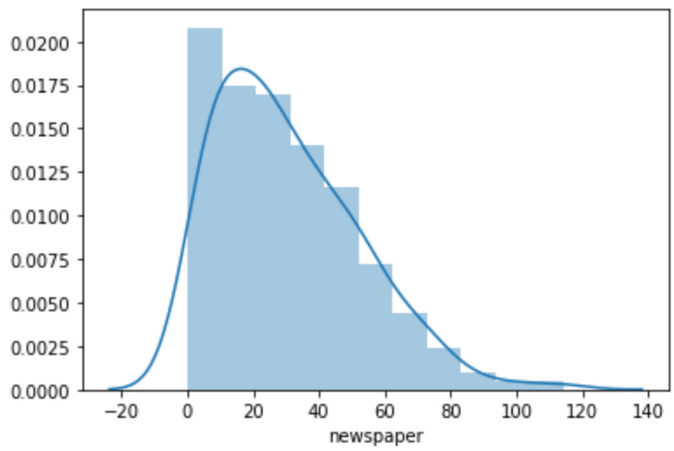
In [22]:
```
sns.distplot(df['radio'])
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x200f4121048>



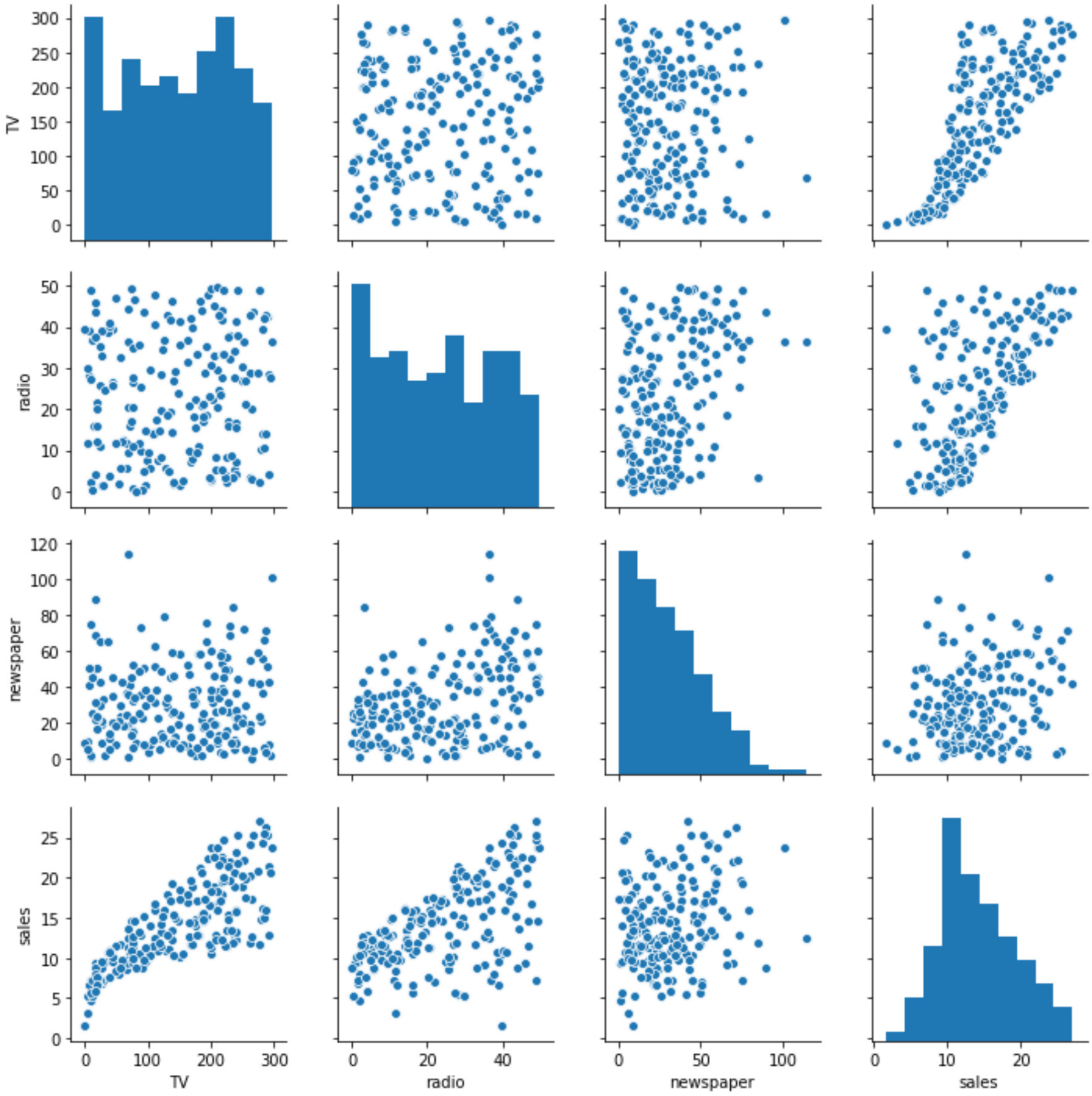

In [23]:
```
sns.distplot(df['newspaper'])
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x200f4165128>



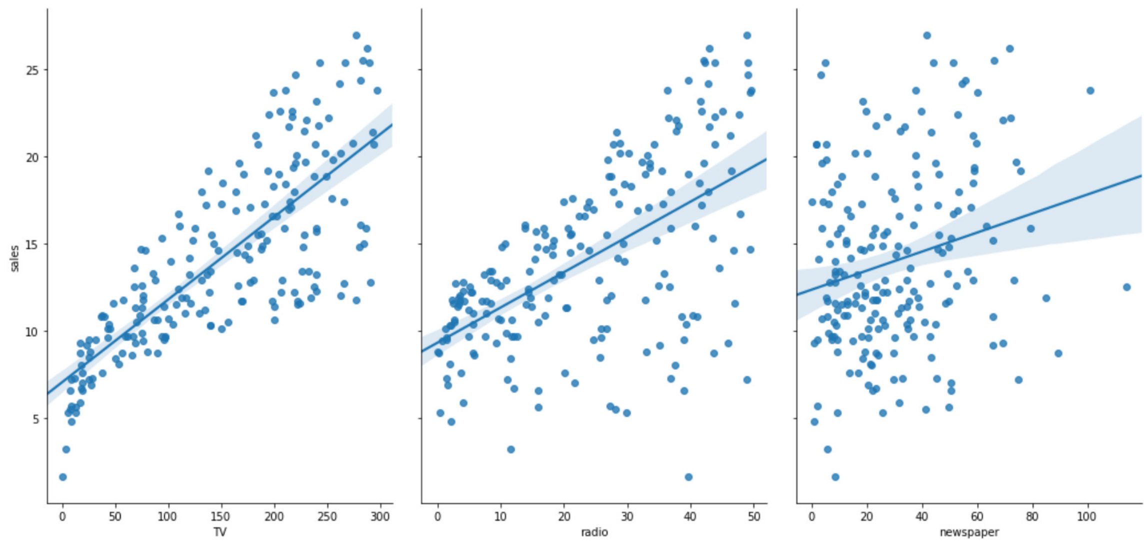
In [24]:
```
sns.pairplot(df)
```

Out[24]: <seaborn.axisgrid.PairGrid at 0x200f421ce48>




In [27]:
```
# Let's see how Sales are related with other variables using scatter plot.
sns.pairplot(df, x_vars=['TV', 'radio', 'newspaper'], y_vars='sales', size=7, as
pect=0.7, kind='reg')
```

Out[27]: <seaborn.axisgrid.PairGrid at 0x200f4fc60b8>

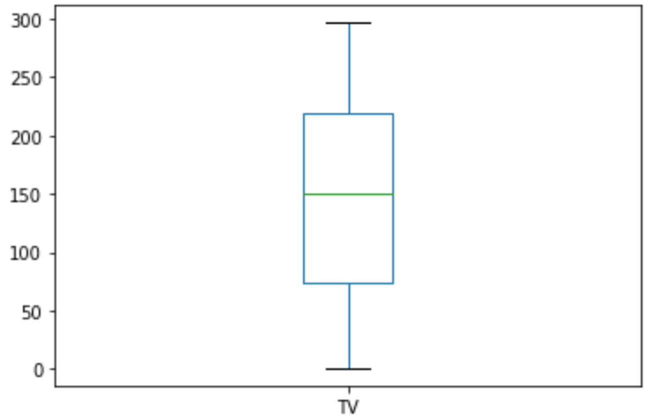


In [ ]:
```
# TV ads has strong impact on sales.
```

In [ ]:
```
# No negatively skewed data
```

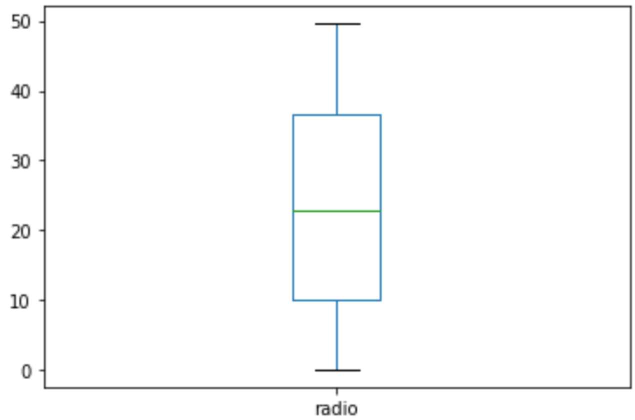
In [28]:
```
# Univariate analysis
# Lets check for columns
df['TV'].plot.box()
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x200f4e6b4e0>



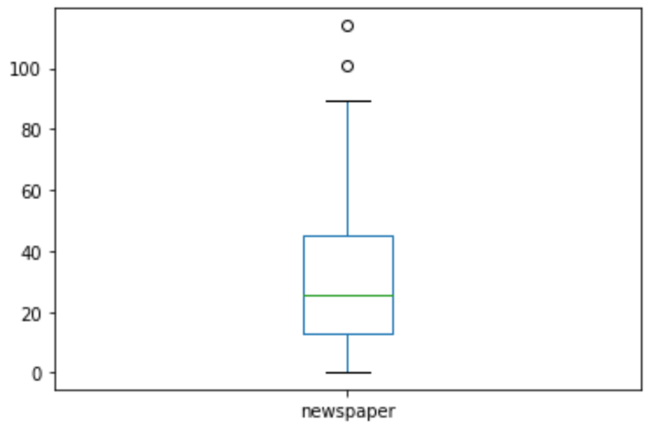

In [31]:
```
df['radio'].plot.box()
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x200f4f41b70>




In [32]:
```
df['newspaper'].plot.box() # newspaper has outlier
```

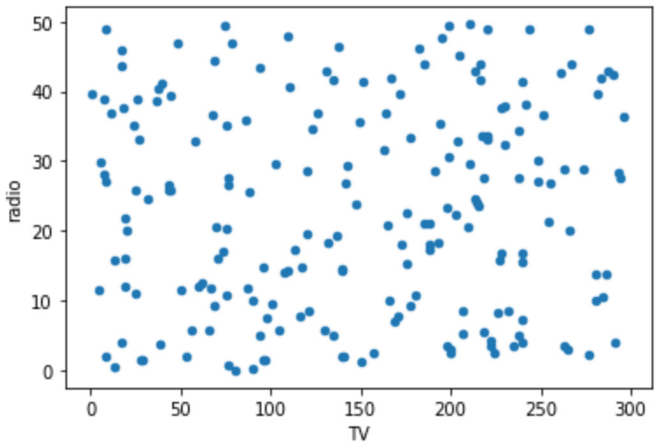
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x200f4f8b6a0>




In [78]:
```
# Bivariate analysis
```

In [30]:
```
df.plot.scatter('TV','radio')
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x200f4eca780>
```




In [33]:
```
# There are no considerable outliers present in the data.
```

Out[33]:
```
TV           float64
radio        float64
newspaper    float64
sales        float64
dtype: object
```

In [10]:
```
# Using z score method to remove outlier
z=np.abs(zscore(df))
df1=df[(z<3).all(axis=1)]
```

In [11]:
```
df.shape,df1.shape
```

Out[11]:
```
((200, 4), (198, 4))
```

In [12]:
```
# Checcking skewness of data
skw=df1.skew()
skw
```

Out[12]:
```
TV          -0.082332
radio        0.114842
newspaper    0.650112
sales        0.407130
dtype: float64
```

In [13]:
```
df1.head()
```

Out[13]:

| | 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 [16]:
```
# Model Building
# We will separate target and rest columns
df_x=df1.drop(columns='sales')
y=df1['sales']
```

In [17]:
```
# Scaling of data
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(df_x)
x=pd.DataFrame(x,columns=df_x.columns)
```

In [18]:
```
x.shape,y.shape
```

Out[18]:
```
((198, 3), (198,))
```

In [19]:
```
x.columns
```

Out[19]:
```
Index(['TV', 'radio', 'newspaper'], dtype='object')
```

In [ ]:
```
# Defina a function to find the best r state basis which we will choose the best
model
```

In [20]:
```
# Finding best r_state
def maxr2_score(lr,x,y):
    max_r_score=0
    for r_state in range(42,101):
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random
_state=r_state)
        lr.fit(x_train,y_train)
        pred=lr.predict(x_test)
        r2_scr=r2_score(y_test,pred)
        print('r2 score corresponding to random state',r_state," is ",r2_scr)
        if r2_scr>max_r_score:
            max_r_score=r2_scr
            final_r_state=r_state
    print('max r2 score corresponding to ', final_r_state," is ",max_r_score)
    return final_r_state
```

In [21]:
```
# Lets use LinearRegression
lr=LinearRegression()
r_state=maxr2_score(lr,x,y)
```

```
r2 score corresponding to random state 42  is  0.8989454779619588
r2 score corresponding to random state 43  is  0.8751484803695657
r2 score corresponding to random state 44  is  0.8117000632290441
r2 score corresponding to random state 45  is  0.8747422037972019
r2 score corresponding to random state 46  is  0.8726271917983103
r2 score corresponding to random state 47  is  0.8922946750347811
r2 score corresponding to random state 48  is  0.866237899668671
r2 score corresponding to random state 49  is  0.8351144932670392
r2 score corresponding to random state 50  is  0.8429785403092693
r2 score corresponding to random state 51  is  0.8969749608189316
r2 score corresponding to random state 52  is  0.8925145995117169
r2 score corresponding to random state 53  is  0.8524939615808071
r2 score corresponding to random state 54  is  0.8776799027550092
r2 score corresponding to random state 55  is  0.8977109139390169
r2 score corresponding to random state 56  is  0.8436665154822788
r2 score corresponding to random state 57  is  0.8652060367319568
r2 score corresponding to random state 58  is  0.930113922012518
r2 score corresponding to random state 59  is  0.9184306555712066
r2 score corresponding to random state 60  is  0.8823000359749914
r2 score corresponding to random state 61  is  0.8532975619375343
r2 score corresponding to random state 62  is  0.8571507573921291
r2 score corresponding to random state 63  is  0.8972874269449632
r2 score corresponding to random state 64  is  0.9267279065103222
r2 score corresponding to random state 65  is  0.8825466552554457
r2 score corresponding to random state 66  is  0.8744920660734806
r2 score corresponding to random state 67  is  0.9057874401170471
r2 score corresponding to random state 68  is  0.8617429493016013
r2 score corresponding to random state 69  is  0.906070694321867
r2 score corresponding to random state 70  is  0.9265656652996626
r2 score corresponding to random state 71  is  0.8071830117377917
r2 score corresponding to random state 72  is  0.9212755833741663
r2 score corresponding to random state 73  is  0.8849214355996005
r2 score corresponding to random state 74  is  0.9298308020128561
r2 score corresponding to random state 75  is  0.8756772369429391
r2 score corresponding to random state 76  is  0.851612626570941
r2 score corresponding to random state 77  is  0.9166905563751221
r2 score corresponding to random state 78  is  0.790234402720306
r2 score corresponding to random state 79  is  0.9014406236580841
r2 score corresponding to random state 80  is  0.9037187128234276
r2 score corresponding to random state 81  is  0.9199455602534609
r2 score corresponding to random state 82  is  0.839876824043998
r2 score corresponding to random state 83  is  0.8791148089196503
r2 score corresponding to random state 84  is  0.9292654473438203
r2 score corresponding to random state 85  is  0.7705544702687278
r2 score corresponding to random state 86  is  0.9047661303024037
r2 score corresponding to random state 87  is  0.8087421010289435
r2 score corresponding to random state 88  is  0.8837353565017301
r2 score corresponding to random state 89  is  0.8458421248110118
r2 score corresponding to random state 90  is  0.9477136584598765
r2 score corresponding to random state 91  is  0.8986272987144844
r2 score corresponding to random state 92  is  0.8937806772897253
r2 score corresponding to random state 93  is  0.8435303584836114
r2 score corresponding to random state 94  is  0.8679025522688202
r2 score corresponding to random state 95  is  0.8479116409360149
r2 score corresponding to random state 96  is  0.854981902005895
r2 score corresponding to random state 97  is  0.897572426046094
r2 score corresponding to random state 98  is  0.8382459156006744
r2 score corresponding to random state 99  is  0.9300192208914474
r2 score corresponding to random state 100  is  0.8556177767066226
max r2 score corresponding to  90  is  0.9477136584598765
```

In [27]:
```
# To find optimum value of n_neighbours for KNN model
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
neighbors={"n_neighbors":range(1,30)}
knr=KNeighborsRegressor()
gknr=GridSearchCV(knr,neighbors,cv=10)
gknr.fit(x,y)
gknr.best_params_
```

Out[27]:
```
{'n_neighbors': 4}
```

In [88]:
```
# Lets use KNN regression model
knr=KNeighborsRegressor(n_neighbors=4)
r_state=maxr2_score(knr,x,y)
```

```
r2 score corresponding to random state 42  is  0.9357149671313382
r2 score corresponding to random state 43  is  0.9327876926693458
r2 score corresponding to random state 44  is  0.9267724536317711
r2 score corresponding to random state 45  is  0.9086023350161672
r2 score corresponding to random state 46  is  0.9021706810143139
r2 score corresponding to random state 47  is  0.9450754310751618
r2 score corresponding to random state 48  is  0.9273540269700297
r2 score corresponding to random state 49  is  0.9192521343277706
r2 score corresponding to random state 50  is  0.943788389154849
r2 score corresponding to random state 51  is  0.9210612630483748
r2 score corresponding to random state 52  is  0.9658710017369763
r2 score corresponding to random state 53  is  0.8957877746975467
r2 score corresponding to random state 54  is  0.9363433815179443
r2 score corresponding to random state 55  is  0.9328271051139083
r2 score corresponding to random state 56  is  0.8929406220782272
r2 score corresponding to random state 57  is  0.9075507910153975
r2 score corresponding to random state 58  is  0.9549218036960777
r2 score corresponding to random state 59  is  0.9415261605183628
r2 score corresponding to random state 60  is  0.9225379125611011
r2 score corresponding to random state 61  is  0.9057745956644061
r2 score corresponding to random state 62  is  0.9391707061323228
r2 score corresponding to random state 63  is  0.9590143274430862
r2 score corresponding to random state 64  is  0.9517716615394614
r2 score corresponding to random state 65  is  0.9386162709421249
r2 score corresponding to random state 66  is  0.9236730908602018
r2 score corresponding to random state 67  is  0.9530756649182973
r2 score corresponding to random state 68  is  0.9195544759689912
r2 score corresponding to random state 69  is  0.9426817912248506
r2 score corresponding to random state 70  is  0.9458592170809582
r2 score corresponding to random state 71  is  0.9389247318352617
r2 score corresponding to random state 72  is  0.9386772605314783
r2 score corresponding to random state 73  is  0.9602599925142428
r2 score corresponding to random state 74  is  0.9455304780958627
r2 score corresponding to random state 75  is  0.9397009808244194
r2 score corresponding to random state 76  is  0.940726292576695
r2 score corresponding to random state 77  is  0.9186621925698901
r2 score corresponding to random state 78  is  0.8848819986682235
r2 score corresponding to random state 79  is  0.9332216011073982
r2 score corresponding to random state 80  is  0.9589310620554687
r2 score corresponding to random state 81  is  0.9568644308962696
r2 score corresponding to random state 82  is  0.9042495120953975
r2 score corresponding to random state 83  is  0.9168069750233252
r2 score corresponding to random state 84  is  0.9149434719251107
r2 score corresponding to random state 85  is  0.9078040393087551
r2 score corresponding to random state 86  is  0.9399426987082908
r2 score corresponding to random state 87  is  0.914747240675686
r2 score corresponding to random state 88  is  0.9404426956928879
r2 score corresponding to random state 89  is  0.960622470667016
r2 score corresponding to random state 90  is  0.9345308633477739
r2 score corresponding to random state 91  is  0.9442768514670933
r2 score corresponding to random state 92  is  0.9500259149690867
r2 score corresponding to random state 93  is  0.9134656395545542
r2 score corresponding to random state 94  is  0.9118833011840795
r2 score corresponding to random state 95  is  0.9074811059403959
r2 score corresponding to random state 96  is  0.946493914294995
r2 score corresponding to random state 97  is  0.9609474497215961
r2 score corresponding to random state 98  is  0.8842499494266973
r2 score corresponding to random state 99  is  0.9418554511016134
r2 score corresponding to random state 100  is  0.8966163403224618
max r2 score corresponding to  52  is  0.9658710017369763
```

In [89]:
```
# using Lasso
from sklearn.linear_model import Lasso
lsreg=Lasso()
parameters={'alpha':[0.001,0.01,0.1,1]}
clf=GridSearchCV(lsreg,parameters,cv=10)
clf.fit(x,y)
clf.best_params_
```

Out[89]:
```
{'alpha': 0.1}
```

In [90]:
```
# Lts check max r2 score
lsreg=Lasso(alpha=0.1)
r_state=maxr2_score(lsreg,x,y)
```

```
r2 score corresponding to random state 42  is  0.8980209932900431
r2 score corresponding to random state 43  is  0.8748274285562571
r2 score corresponding to random state 44  is  0.821964418062779
r2 score corresponding to random state 45  is  0.8790722914669622
r2 score corresponding to random state 46  is  0.8762496226404188
r2 score corresponding to random state 47  is  0.8957580511531971
r2 score corresponding to random state 48  is  0.8698158257018274
r2 score corresponding to random state 49  is  0.8368871187315301
r2 score corresponding to random state 50  is  0.8454831532120928
r2 score corresponding to random state 51  is  0.8998279979944519
r2 score corresponding to random state 52  is  0.8958286153769411
r2 score corresponding to random state 53  is  0.8518482932500415
r2 score corresponding to random state 54  is  0.876980803147764
r2 score corresponding to random state 55  is  0.8966769669622391
r2 score corresponding to random state 56  is  0.8415696924636991
r2 score corresponding to random state 57  is  0.8700531371333975
r2 score corresponding to random state 58  is  0.9289149990637448
r2 score corresponding to random state 59  is  0.9170299901628998
r2 score corresponding to random state 60  is  0.8801430063716252
r2 score corresponding to random state 61  is  0.8551202968872668
r2 score corresponding to random state 62  is  0.8598800808647138
r2 score corresponding to random state 63  is  0.8978866939351249
r2 score corresponding to random state 64  is  0.9284868429078285
r2 score corresponding to random state 65  is  0.8818699681447264
r2 score corresponding to random state 66  is  0.8798167550061156
r2 score corresponding to random state 67  is  0.9046713571516178
r2 score corresponding to random state 68  is  0.8669674499329241
r2 score corresponding to random state 69  is  0.9079746135093784
r2 score corresponding to random state 70  is  0.923546843087898
r2 score corresponding to random state 71  is  0.8107096369008978
r2 score corresponding to random state 72  is  0.9217914228387182
r2 score corresponding to random state 73  is  0.8859439188430773
r2 score corresponding to random state 74  is  0.9235677379405508
r2 score corresponding to random state 75  is  0.8759695712700271
r2 score corresponding to random state 76  is  0.8500553816373546
r2 score corresponding to random state 77  is  0.9212060581336318
r2 score corresponding to random state 78  is  0.7888367503863347
r2 score corresponding to random state 79  is  0.8990430881577671
r2 score corresponding to random state 80  is  0.9065711107408589
r2 score corresponding to random state 81  is  0.9199058472103132
r2 score corresponding to random state 82  is  0.8445756374212661
r2 score corresponding to random state 83  is  0.8756506933391401
r2 score corresponding to random state 84  is  0.9271708407927878
r2 score corresponding to random state 85  is  0.7738165835671043
r2 score corresponding to random state 86  is  0.907913585233876
r2 score corresponding to random state 87  is  0.8098659231652563
r2 score corresponding to random state 88  is  0.8808753791479955
r2 score corresponding to random state 89  is  0.8499507767597355
r2 score corresponding to random state 90  is  0.9440148550684346
r2 score corresponding to random state 91  is  0.8945218159905867
r2 score corresponding to random state 92  is  0.8931305476779863
r2 score corresponding to random state 93  is  0.8423456096853021
r2 score corresponding to random state 94  is  0.8652297008967426
r2 score corresponding to random state 95  is  0.8448396548568596
r2 score corresponding to random state 96  is  0.8571829228353596
r2 score corresponding to random state 97  is  0.8963948121208023
r2 score corresponding to random state 98  is  0.8391987536701183
r2 score corresponding to random state 99  is  0.925021849325458
r2 score corresponding to random state 100  is  0.8598766361943919
max r2 score corresponding to  90  is  0.9440148550684346
```

In [28]:
```
# we will use gradient boosting Technique
# for getting best prameters will use grid search
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor()
parameters={"learning_rate":[0.001,0.01,0.1,1],"n_estimators":[10,100,500,1000]}
clf=GridSearchCV(gbr,parameters,cv=5)
clf.fit(x,y)
clf.best_params_
```

Out[28]:
```
{'learning_rate': 0.1, 'n_estimators': 500}
```

In [29]:
```
gbr=GradientBoostingRegressor(learning_rate=0.1,n_estimators=500)
r_state=maxr2_score(gbr,x,y)
```

```
r2 score corresponding to random state 42  is  0.9853488603166238
r2 score corresponding to random state 43  is  0.9734496285828619
r2 score corresponding to random state 44  is  0.9656168839572921
r2 score corresponding to random state 45  is  0.973743289340044
r2 score corresponding to random state 46  is  0.976886117142699
r2 score corresponding to random state 47  is  0.9739841416021219
r2 score corresponding to random state 48  is  0.9774546972711915
r2 score corresponding to random state 49  is  0.9727611617081663
r2 score corresponding to random state 50  is  0.9836719505328788
r2 score corresponding to random state 51  is  0.9831800608394148
r2 score corresponding to random state 52  is  0.9861946090513255
r2 score corresponding to random state 53  is  0.9630376804422983
r2 score corresponding to random state 54  is  0.9853916873688009
r2 score corresponding to random state 55  is  0.9888142873845667
r2 score corresponding to random state 56  is  0.9676094877290173
r2 score corresponding to random state 57  is  0.9586057068663282
r2 score corresponding to random state 58  is  0.9886692871413602
r2 score corresponding to random state 59  is  0.9826813269829533
r2 score corresponding to random state 60  is  0.9727871540539551
r2 score corresponding to random state 61  is  0.9679882536001506
r2 score corresponding to random state 62  is  0.9712768477947066
r2 score corresponding to random state 63  is  0.9814097618219255
r2 score corresponding to random state 64  is  0.9800105124817331
r2 score corresponding to random state 65  is  0.9868210753415084
r2 score corresponding to random state 66  is  0.9674381715666975
r2 score corresponding to random state 67  is  0.9727314586180603
r2 score corresponding to random state 68  is  0.9774098165854012
r2 score corresponding to random state 69  is  0.9885115083040593
r2 score corresponding to random state 70  is  0.986832040919956
r2 score corresponding to random state 71  is  0.9789700558006583
r2 score corresponding to random state 72  is  0.9828122451566058
r2 score corresponding to random state 73  is  0.9745857696843457
r2 score corresponding to random state 74  is  0.9897173850597649
r2 score corresponding to random state 75  is  0.9778837826468862
r2 score corresponding to random state 76  is  0.9833036626990521
r2 score corresponding to random state 77  is  0.9811954807127101
r2 score corresponding to random state 78  is  0.9353062810298355
r2 score corresponding to random state 79  is  0.9880707726766614
r2 score corresponding to random state 80  is  0.987785710336341
r2 score corresponding to random state 81  is  0.9862212546187157
r2 score corresponding to random state 82  is  0.9624423010632969
r2 score corresponding to random state 83  is  0.9690774534008716
r2 score corresponding to random state 84  is  0.9837879342168804
r2 score corresponding to random state 85  is  0.9618077021784653
r2 score corresponding to random state 86  is  0.982800431654128
r2 score corresponding to random state 87  is  0.9584669260624604
r2 score corresponding to random state 88  is  0.9808831435907512
r2 score corresponding to random state 89  is  0.9770846391739656
r2 score corresponding to random state 90  is  0.9797625399673141
r2 score corresponding to random state 91  is  0.9810195967502829
r2 score corresponding to random state 92  is  0.9844843263326095
r2 score corresponding to random state 93  is  0.9630497452010177
r2 score corresponding to random state 94  is  0.9707806800562109
r2 score corresponding to random state 95  is  0.9657652247596049
r2 score corresponding to random state 96  is  0.9791591824604176
r2 score corresponding to random state 97  is  0.9849139680201991
r2 score corresponding to random state 98  is  0.9611355266293411
r2 score corresponding to random state 99  is  0.9863354096106337
r2 score corresponding to random state 100  is  0.9616711545282447
max r2 score corresponding to  74  is  0.9897173850597649
```

In [94]:
```
# Use adaboost
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
ada_reg=AdaBoostRegressor()
parameters={"learning_rate":[0.001,0.01,0.1,1],"n_estimators":[10,100,500,100
0],'base_estimator':[lr,lsreg,DecisionTreeRegressor()]}
clf=GridSearchCV(ada_reg,parameters,cv=5)
clf.fit(x,y)
clf.best_params_
```

Out[94]:
```
{'base_estimator': DecisionTreeRegressor(criterion='mse', max_depth=None, max_
features=None,
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       presort=False, random_state=None, splitter='best'),
 'learning_rate': 1,
 'n_estimators': 500}
```

In [95]:
```
Dt=DecisionTreeRegressor()
ada_reg=AdaBoostRegressor(base_estimator=Dt,learning_rate=1,n_estimators=500)
r_state=maxr2_score(ada_reg,x,y)
```

```
r2 score corresponding to random state 42  is  0.9743484336230004
r2 score corresponding to random state 43  is  0.9730131075577894
r2 score corresponding to random state 44  is  0.95133356642875
r2 score corresponding to random state 45  is  0.9637628990450402
r2 score corresponding to random state 46  is  0.9752981252090719
r2 score corresponding to random state 47  is  0.9633169287994574
r2 score corresponding to random state 48  is  0.9693445248446012
r2 score corresponding to random state 49  is  0.9771479524830656
r2 score corresponding to random state 50  is  0.9803075874826715
r2 score corresponding to random state 51  is  0.960678610764933
r2 score corresponding to random state 52  is  0.9837192852766237
r2 score corresponding to random state 53  is  0.9580720370345449
r2 score corresponding to random state 54  is  0.9839331348038762
r2 score corresponding to random state 55  is  0.979892724282739
r2 score corresponding to random state 56  is  0.9744008959686411
r2 score corresponding to random state 57  is  0.9577556226182002
r2 score corresponding to random state 58  is  0.9781956686391033
r2 score corresponding to random state 59  is  0.9789541641219786
r2 score corresponding to random state 60  is  0.9840088080463658
r2 score corresponding to random state 61  is  0.9636021540898023
r2 score corresponding to random state 62  is  0.9549076058462642
r2 score corresponding to random state 63  is  0.9805046054576465
r2 score corresponding to random state 64  is  0.9811972385305863
r2 score corresponding to random state 65  is  0.9752416389929053
r2 score corresponding to random state 66  is  0.9733973790372301
r2 score corresponding to random state 67  is  0.9734465969589634
r2 score corresponding to random state 68  is  0.9816570518752105
r2 score corresponding to random state 69  is  0.984510899399335
r2 score corresponding to random state 70  is  0.9828213401174482
r2 score corresponding to random state 71  is  0.9672695664309781
r2 score corresponding to random state 72  is  0.9774512429029667
r2 score corresponding to random state 73  is  0.9666850992291891
r2 score corresponding to random state 74  is  0.9826897792472378
r2 score corresponding to random state 75  is  0.9618252018408736
r2 score corresponding to random state 76  is  0.9813989423394675
r2 score corresponding to random state 77  is  0.979416328232338
r2 score corresponding to random state 78  is  0.9416093516254971
r2 score corresponding to random state 79  is  0.9809436789894772
r2 score corresponding to random state 80  is  0.9749528997677849
r2 score corresponding to random state 81  is  0.9854496461289628
r2 score corresponding to random state 82  is  0.9590876710827477
r2 score corresponding to random state 83  is  0.9669946801631846
r2 score corresponding to random state 84  is  0.9765859161823552
r2 score corresponding to random state 85  is  0.9642079372223813
r2 score corresponding to random state 86  is  0.9712284453379579
r2 score corresponding to random state 87  is  0.9627473434898063
r2 score corresponding to random state 88  is  0.9785224290273667
r2 score corresponding to random state 89  is  0.9743791306373192
r2 score corresponding to random state 90  is  0.9769211121652193
r2 score corresponding to random state 91  is  0.9819569050911371
r2 score corresponding to random state 92  is  0.9750507659013019
r2 score corresponding to random state 93  is  0.9678225127234248
r2 score corresponding to random state 94  is  0.9718384439031518
r2 score corresponding to random state 95  is  0.9685530342226885
r2 score corresponding to random state 96  is  0.9575223791988835
r2 score corresponding to random state 97  is  0.9820696744696449
r2 score corresponding to random state 98  is  0.9566593724107162
r2 score corresponding to random state 99  is  0.9800059982005398
r2 score corresponding to random state 100  is  0.9537452684775216
max r2 score corresponding to  81  is  0.9854496461289628
```

In [ ]:
```
# GradientBoostingRegressor and Decision Tree Regressor are the best model
```

In [30]:
```
# lets check cross val score
from sklearn.model_selection import cross_val_score
print("Mean r2 score",cross_val_score(gbr,x,y,cv=5,scoring="r2").mean())
```

```
Mean r2 score 0.978310028684407
```

In [31]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=7
4)
```

In [32]:
```
gbr=GradientBoostingRegressor(learning_rate=0.1,n_estimators=500)
gbr.fit(x_train,y_train)
pred=gbr.predict(x_test)
```

In [33]:
```
print("r2 score",r2_score(y_test,pred))
```

```
r2 score 0.9897174944137418
```

In [34]:
```
print("RMSE is : ",np.sqrt(mean_squared_error(y_test,pred)))
```

```
RMSE is :  0.6267307686179309
```

In [35]:
```
from sklearn.externals import joblib
```

In [36]:
```
joblib.dump(gbr,"sales_predict.pkl")
```

Out[36]:
```
['sales_predict.pkl']
```

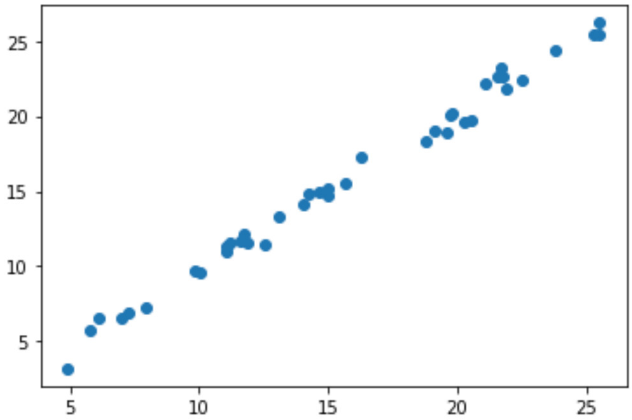
In [37]:
```
a=[pred,y_test]
```

In [38]:
```
a
```

Out[38]:
```
[array([19.79772981, 10.03566519, 21.71039905, 20.50709658, 14.96110592,
         6.95951679, 16.30369411, 11.07917717, 14.2538927 , 21.78133791,
        15.65944726, 12.52303264, 18.80791168, 25.46697941, 21.90049323,
         5.74094276, 14.95459163, 19.73111586, 25.2869774 , 19.62092212,
        13.10599864, 11.16202124, 11.89870135,  4.90861671, 11.72310042,
        14.66372998, 14.0072315 , 23.77217114, 21.581095  , 25.5111832 ,
        11.06899534, 20.27445297,  6.08211652, 22.51958607,  9.86131243,
        11.61397096,  7.93172636, 21.05055526,  7.2338646 , 19.1445069 ]),
 177    20.2
 139     9.6
 48     23.2
 125    19.7
 104    14.7
 120     6.6
 195    17.3
 74     11.0
 103    14.8
 186    22.6
 24     15.5
 95     11.5
 71     18.3
 99     25.4
 112    21.8
 133     5.7
 86     15.2
 143    20.1
 148    25.4
 69     18.9
 162    13.3
 123    11.6
 136    11.6
 156     3.2
 168    12.2
 163    14.9
 113    14.1
 18     24.4
 53     22.6
 184    26.2
 19     11.3
 194    19.6
 127     6.6
 16     22.4
 130     9.7
 174    11.7
 107     7.2
 94     22.2
 77      6.9
 15     19.0
 Name: sales, dtype: float64]
```

In [39]:
```
plt.scatter(pred, y_test)
```

Out[39]: <matplotlib.collections.PathCollection at 0x21667f0d198>




In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [150]:

In [ ]:

In [ ]: