NAME: RADHIKA GUPTA

REG. NO: 22MCA1119

ITA-6016 Machine Learning

Digital Assignment –Lab -1

SUBMITTED TO: Dr Dominic Savio M

LINEAR REGRESSION:

CODE OF THE PROGRAM AND OUTPUT:

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3 (ipyke

Run Code

In [1]:
```
import warnings
warnings.filterwarnings('ignore')

# Import the numpy and pandas package

import numpy as np
import pandas as pd

# Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
advertising = pd.DataFrame(pd.read_csv("D:\\vit notes\\MCA Second Semester\\MachineLearning\\advertising.csv"))
advertising.head()
```

Out[3]:

	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

In [4]:
```
advertising.shape
```

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

In [5]:
```
advertising.info()
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Pyth

Run Code

In [5]:
```
advertising.info()
```

Out[5]:
```
<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 [6]:
```
advertising.describe()
```

Out[6]:

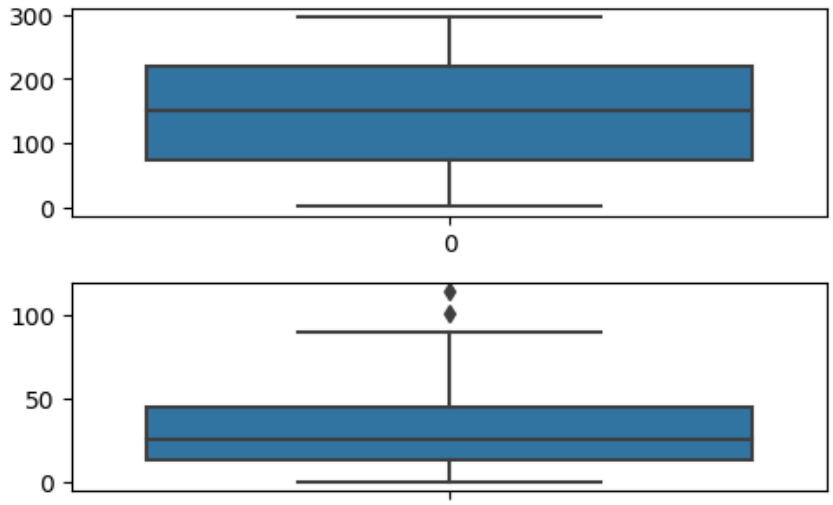
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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

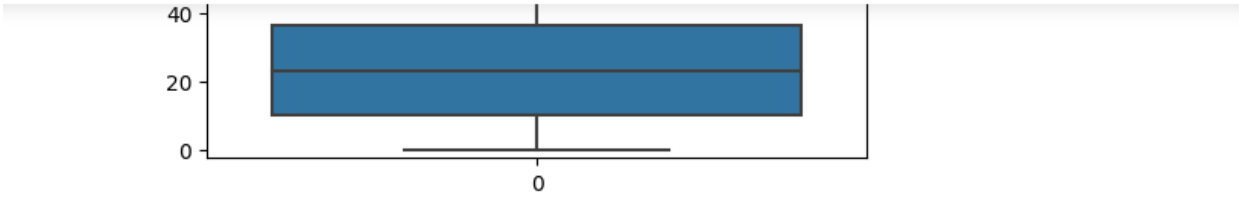
In [7]:
```
# Checking Null values
advertising.isnull().sum()*100/advertising.shape[0]
```

```
advertising.isnull().sum()*100/advertising.shape[0]
# There are no NULL values in the dataset, hence it is clean.
```

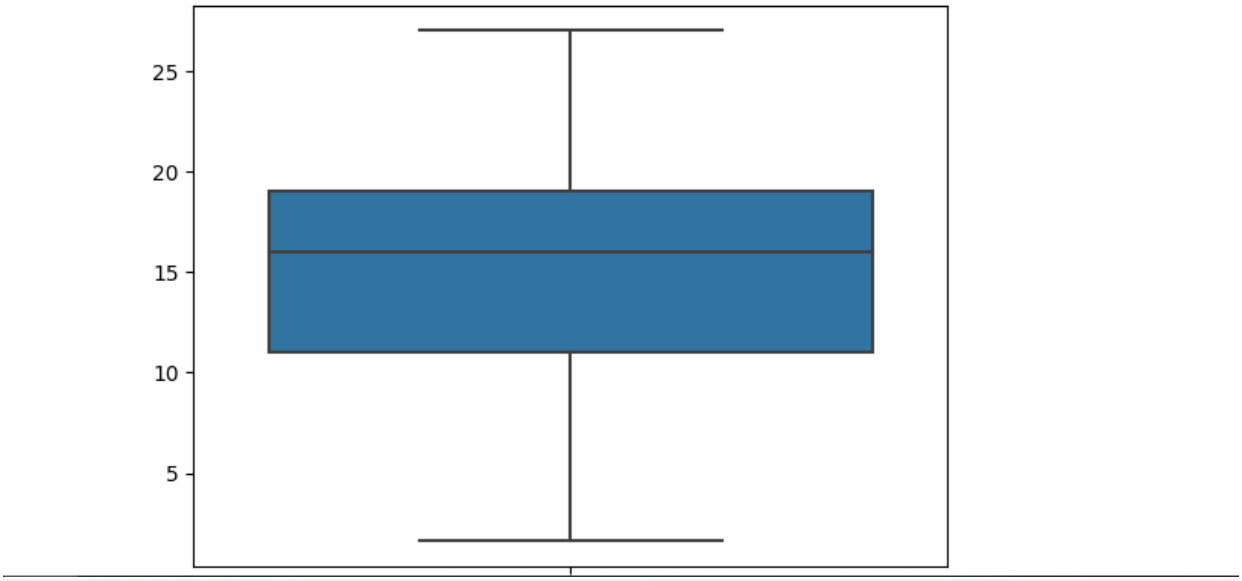
Out[7]:
```
TV           0.0
Radio        0.0
Newspaper    0.0
Sales        0.0
dtype: float64
```

In [8]:
```
# Outlier Analysis
fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(advertising['TV'], ax = axs[0])
plt2 = sns.boxplot(advertising['Newspaper'], ax = axs[1])
plt3 = sns.boxplot(advertising['Radio'], ax = axs[2])
plt.tight_layout()
```

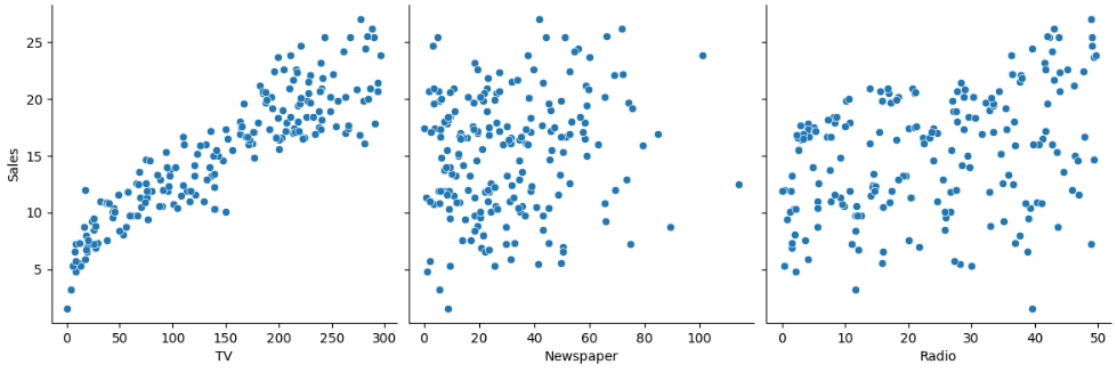




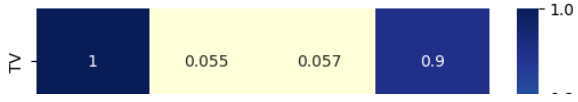
In [9]:
```
sns.boxplot(advertising['Sales'])
plt.show()
```

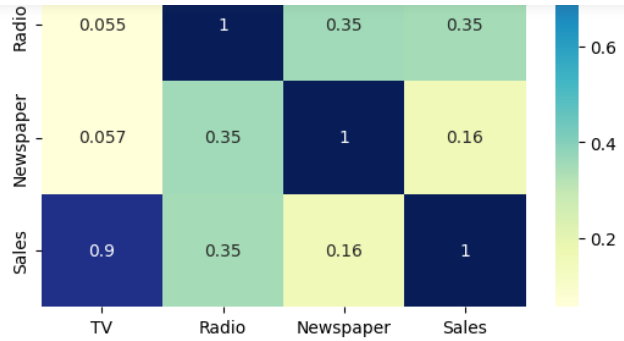


In [10]:
```
# Let's see how Sales are related with other variables using scatter plot.
sns.pairplot(advertising, x_vars=['TV', 'Newspaper', 'Radio'], y_vars='Sales', height=4, aspect=1, kind='scatter')
plt.show()
```



In [11]:
```
# Let's see the correlation between different variables.
sns.heatmap(advertising.corr(), cmap="YlGnBu", annot = True)
plt.show()
```





In [12]:
```
X = advertising['TV']
y = advertising['Sales']
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.3, random_state = 100)
```

In [14]:
```
X_train.head()
```

Out[14]:
```
74     213.4
3      151.5
185    205.0
26     142.9
90     134.3
Name: TV, dtype: float64
```


```
185    205.0
26     142.9
90     134.3
Name: TV, dtype: float64
```


In [15]:
```
y_train.head()
```

Out[15]:
```
74     17.0
3      16.5
185    22.6
26     15.0
90     14.0
Name: Sales, dtype: float64
```

In [16]:
```
import statsmodels.api as sm
```

In [17]:
```
# Add a constant to get an intercept
X_train_sm = sm.add_constant(X_train)

# Fit the resgression line using 'OLS'
lr = sm.OLS(y_train, X_train_sm).fit()
```

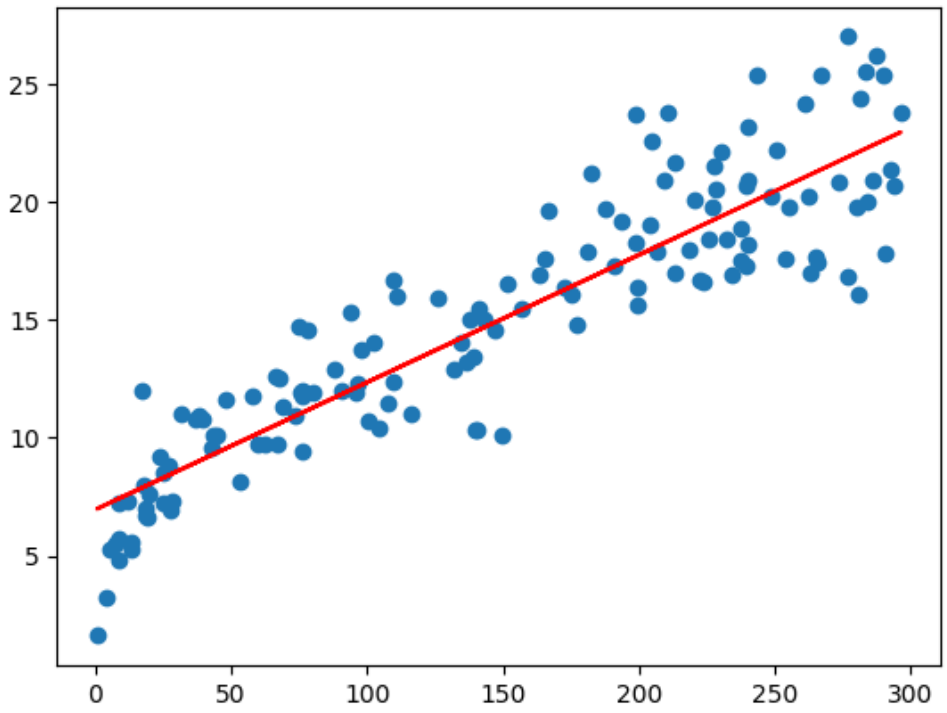
In [18]:
```
# Print the parameters, i.e. the intercept and the slope of the regression line fitted
lr.params
```

Out[18]:
```
const    6.948683
TV       0.054546
dtype: float64
```

In [19]:
```
# Performing a summary operation lists out all the different parameters of the regression line fitted
print(lr.summary())
```

```
                            OLS Regression Results
```

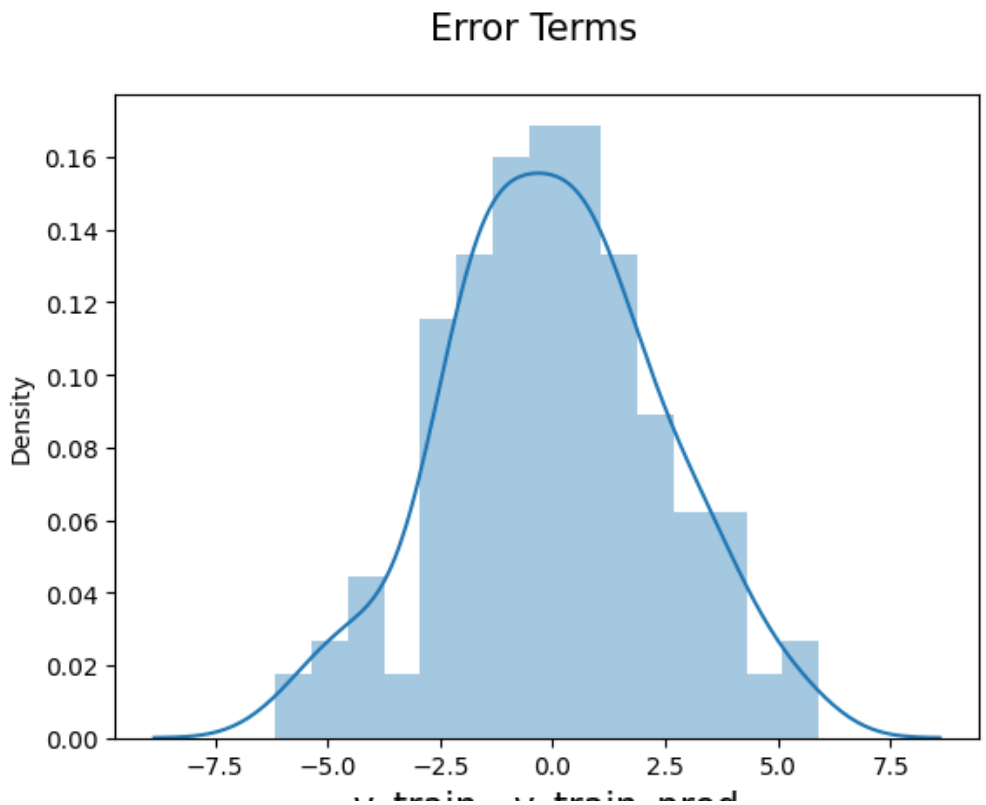
In [20]:
```
plt.scatter(X_train, y_train)
plt.plot(X_train, 6.948 + 0.054*X_train, 'r')
plt.show()
```



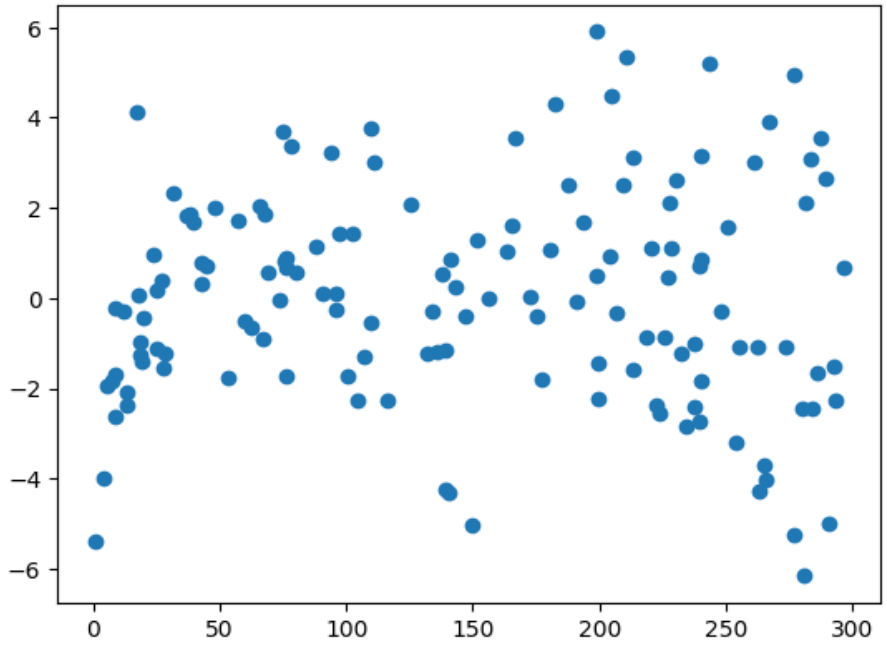
In [21]:
```
y_train_pred = lr.predict(X_train_sm)
res = (y_train - y_train_pred)
```

In [22]:
```
fig = plt.figure()
```

In [22]:
```
fig = plt.figure()
sns.distplot(res, bins = 15)
fig.suptitle('Error Terms', fontsize = 15)                  # Plot heading
plt.xlabel('y_train - y_train_pred', fontsize = 15)         # X-label
plt.show()
```



In [23]:
```
plt.scatter(X_train,res)
plt.show()
```



In [24]:
```
# Add a constant to X_test
X_test_sm = sm.add_constant(X_test)
```

In [24]:
```
# Add a constant to X_test
X_test_sm = sm.add_constant(X_test)

# Predict the y values corresponding to X_test_sm
y_pred = lr.predict(X_test_sm)
y_pred.head()
```

Out[24]:
```
126     7.374140
104    19.941482
99     14.323269
92     18.823294
111    20.132392
dtype: float64
```

In [25]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
#Returns the mean squared error; we'll take a square root
np.sqrt(mean_squared_error(y_test, y_pred))
2.019296008966232
```

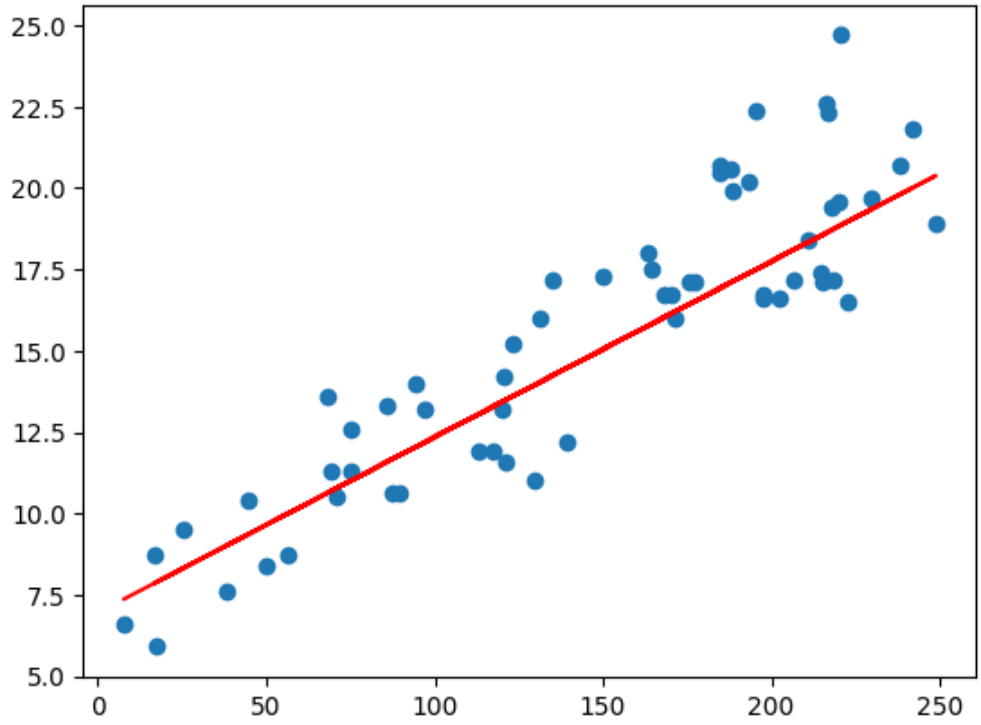
Out[25]:
```
2.019296008966232
```

In [26]:
```
r_squared = r2_score(y_test, y_pred)
r_squared
```

Out[26]:
```
0.792103160124566
```

In [27]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
plt.show()
```

```
In [27]: plt.scatter(X_test, y_test)
         plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
         plt.show()
```



MULTIPLE REGRESSION:

CODE OF THE PROGRAM AND OUTPUT:

Multiple Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
dataset=pd.read_excel("D:\\vit notes\\MCA Second Semester\\MachineLearning\\Date-Wise-Prices-all-Commodity.xlsx")
```

In [3]:
```
dataset
```

Out[3]:

	diffgr:id	msdata:rowOrder	State	District	Market	Commodity	Variety	Grade	Arrival_Date	Min_x0020_Price	Max_x0020_Price	Modal_x00
0	Table1	0	Andhra Pradesh	Chittor	Chittoor	Gur(Jaggery)	NO 1	FAQ	24/02/2023	4000	4100.0	
1	Table2	1	Andhra Pradesh	Chittor	Chittoor	Gur(Jaggery)	NO 2	FAQ	24/02/2023	3000	3500.0	
2	Table3	2	Andhra Pradesh	Chittor	Chittoor	Gur(Jaggery)	NO 3	FAQ	24/02/2023	2300	2300.0	
3	Table4	3	Andhra Pradesh	Chittor	Punganur	Tomato	Hybrid	FAQ	24/02/2023	1340	2000.0	
4	Table5	4	Andhra Pradesh	Chittor	Vayalapadu	Tomato	Local	FAQ	24/02/2023	640	2160.0	
...	...	...	...	...	...	...	...	...	...	...	...	
6430	Table6431	6430	West Bengal	South 24 Parganas	Diamond Harbour(South 24-pgs)	Onion	Red	FAQ	24/02/2023	1400	1500.0	
6431	Table6432	6431	West Bengal	South 24 Parganas	Diamond Harbour(South 24-pgs)	Potato	Jyoti	FAQ	24/02/2023	760	800.0	
6431	Table6432	6431	West Bengal	South 24 Parganas	Diamond Harbour(South 24-pgs)	Potato	Jyoti	FAQ	24/02/2023	760	800.0	
6432	Table6433	6432	West Bengal	South 24 Parganas	Diamond Harbour(South 24-pgs)	Rice	Common	FAQ	24/02/2023	2700	2900.0	
6433	Table6434	6433	West Bengal	South 24 Parganas	Diamond Harbour(South 24-pgs)	Rice	Super Fine	FAQ	24/02/2023	4700	4900.0	
6434	Table6435	6434	West Bengal	South 24 Parganas	Diamond Harbour(South 24-pgs)	Tomato	Tomato	FAQ	24/02/2023	1400	1600.0	

6435 rows × 12 columns

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   diffgr:id          6435 non-null   object
 1   msdata:rowOrder    6435 non-null   int64
 2   State              6435 non-null   object
 3   District           6435 non-null   object
 4   Market             6435 non-null   object
 5   Commodity          6435 non-null   object
 6   Variety            6435 non-null   object
 7   Grade              6435 non-null   object
 8   Arrival_Date       6435 non-null   object
 9   Min_x0020_Price    6435 non-null   int64
 10  Max_x0020_Price    6435 non-null   float64
```

```
 10  Max_x0020_Price    6435 non-null   float64
 11  Modal_x0020_Price  6435 non-null   float64
dtypes: float64(2), int64(2), object(8)
memory usage: 603.4+ KB
```

dataset having null or not

In [5]:
```
dataset.isnull().sum()
```

Out[5]:
```
diffgr:id            0
msdata:rowOrder      0
State                0
District             0
Market               0
Commodity            0
Variety              0
Grade                0
Arrival_Date         0
Min_x0020_Price      0
Max_x0020_Price      0
Modal_x0020_Price    0
dtype: int64
```

In [6]:
```
# work on Data Preproessing
dataset["Market"].unique()
```

Out[6]:
```
array(['Chittoor', 'Punganur', 'Vayalapadu', 'Alur', 'Atmakur',
       'Banaganapalli', 'Nandikotkur', 'Gopalavaram', 'Bihiya', 'Kaimur',
       'Bahadurganj', 'Munghair', 'Nawada',
       'Parsoniya Mandi, Mahua block', 'Chandigarh(Grain/Fruit)',
       'Bhatgaon', 'Kasdol', 'Sarsiwan', 'Kusmee', 'Bilaspur',
       'Pendraroad', 'Sakri', 'Takhatpur', 'Tiphra', 'Gidam',
```

In [7]:
```
dataset.head()
```

Out[7]:

	diffgr:id	msdata:rowOrder	State	District	Market	Commodity	Variety	Grade	Arrival_Date	Min_x0020_Price	Max_x0020_Price	Modal_x0020_Price
0	Table1	0	Andhra Pradesh	Chittor	Chittoor	Gur(Jaggery)	NO 1	FAQ	24/02/2023	4000	4100.0	4000.0
1	Table2	1	Andhra Pradesh	Chittor	Chittoor	Gur(Jaggery)	NO 2	FAQ	24/02/2023	3000	3500.0	3500.0
2	Table3	2	Andhra Pradesh	Chittor	Chittoor	Gur(Jaggery)	NO 3	FAQ	24/02/2023	2300	2300.0	2300.0
3	Table4	3	Andhra Pradesh	Chittor	Punganur	Tomato	Hybrid	FAQ	24/02/2023	1340	2000.0	1670.0
4	Table5	4	Andhra Pradesh	Chittor	Vayalapadu	Tomato	Local	FAQ	24/02/2023	640	2160.0	1400.0

In [8]:
```
from sklearn.preprocessing import LabelEncoder
lbl=LabelEncoder()
dataset["Market"]=lbl.fit_transform(dataset["Market"])
```

In [9]:
```
dataset["Commodity"]=lbl.fit_transform(dataset["Commodity"])
```

In [10]:
```
dataset["Grade"]=lbl.fit_transform(dataset["Grade"])
```

In [11]:
```
dataset["Variety"]=lbl.fit_transform(dataset["Variety"])
```

In [12]:
```
dataset
```

Out[12]:

	diffgr:id	msdata:rowOrder	State	District	Market	Commodity	Variety	Grade	Arrival_Date	Min_x0020_Price	Max_x0020_Price	Modal_x0020_Price
0	Table1	0	Andhra Pradesh	Chittor	124	85	222	0	24/02/2023	4000	4100.0	4000.0

Out[12]:

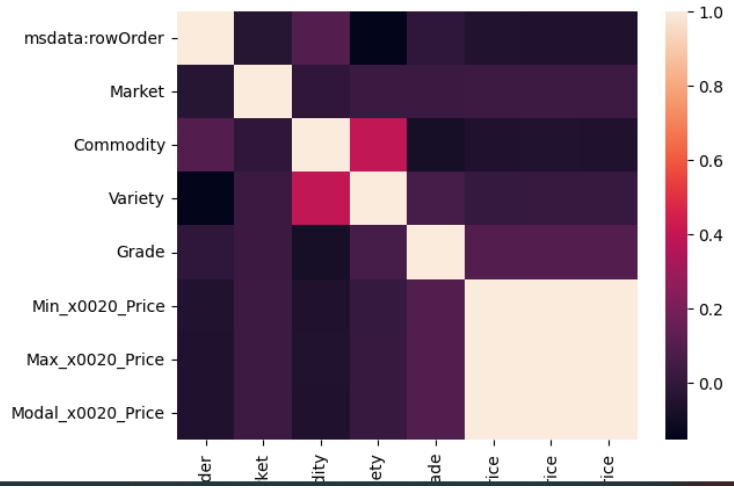
	diffgr:id	msdata:rowOrder	State	District	Market	Commodity	Variety	Grade	Arrival_Date	Min_x0020_Price	Max_x0020_Price	Modal_x0020_Price
0	Table1	0	Andhra Pradesh	Chittor	124	85	222	0	24/02/2023	4000	4100.0	4000.0
1	Table2	1	Andhra Pradesh	Chittor	124	85	223	0	24/02/2023	3000	3500.0	3500.0
2	Table3	2	Andhra Pradesh	Chittor	124	85	224	0	24/02/2023	2300	2300.0	2300.0
3	Table4	3	Andhra Pradesh	Chittor	453	166	155	0	24/02/2023	1340	2000.0	1670.0
4	Table5	4	Andhra Pradesh	Chittor	570	166	192	0	24/02/2023	640	2160.0	1400.0
...	...	...	...	...	...	...	...	...	...	...	...	...
6430	Table6431	6430	West Bengal	South 24 Parganas	155	118	269	0	24/02/2023	1400	1500.0	1450.0
6431	Table6432	6431	West Bengal	South 24 Parganas	155	133	169	0	24/02/2023	760	800.0	780.0
6432	Table6433	6432	West Bengal	South 24 Parganas	155	139	87	0	24/02/2023	2700	2900.0	2800.0
6433	Table6434	6433	West Bengal	South 24 Parganas	155	139	304	0	24/02/2023	4700	4900.0	4800.0
6434	Table6435	6434	West Bengal	South 24 Parganas	155	166	319	0	24/02/2023	1400	1600.0	1500.0

6435 rows × 12 columns

In [13]:
```
import seaborn as sns
sns.heatmap(data=dataset.corr())
```

```
C:\Users\ayush\AppData\Local\Temp\ipykernel_120\131510761.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(data=dataset.corr())
```

Out[13]: <Axes: >



We are predicting product price

In [14]:
```
# We have select features
x=dataset.iloc[:,[4,5,6,7,9,10]].values
y=dataset.iloc[:,-1].values
```

Work on Standardization Tech to Rescale the Dataset

In [15]:
```
from sklearn.preprocessing import StandardScaler
s1=StandardScaler()
x=s1.fit_transform(x)
```

In [16]:
```
e1=x.mean()
e1=round(e1)
e1
```

Out[16]: 0

In [17]:
```
e1=x.var()
e1=round(e1)
e1
```

Out[17]: 1

In [18]:
```
# Will check Data is Normal Distribusted or not

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=11)
```

apply your linear model

In [19]:
```
from sklearn.linear_model import LinearRegression
l1=LinearRegression()
l1.fit(x_train,y_train)
```

Out[19]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:
```
y_reg_score=l1.predict(x_test)
y_reg_score
```

Out[20]:
```
array([ 859.43279458, 2355.86215266, 3748.27723854, ..., 1527.22452285,
       3800.48137786,  453.63848044])
```

In [21]:
```
from sklearn.metrics import mean_squared_error, r2_score
r2_score(y_test,y_reg_score)*100
```

Out[21]: 99.98679475505587

Lets Check Model is Under Fitting or Overfitting

In [22]:
```
from sklearn.linear_model import LinearRegression
l1=LinearRegression()
l1.fit(x_train,y_train)


y_pred_test=l1.predict(x_train)
# y_reg1_score
```

```
y_pred_test=l1.predict(x_train)
# y_reg1_score

from sklearn.metrics import mean_squared_error, r2_score
r2_score(y_train,y_pred_test)*100
```

Out[22]: 99.98590105748974

In [23]:
```
dataset.head()
```

Out[23]:

	diffgr:id	msdata:rowOrder	State	District	Market	Commodity	Variety	Grade	Arrival_Date	Min_x0020_Price	Max_x0020_Price	Modal_x0020_Price
0	Table1	0	Andhra Pradesh	Chittor	124	85	222	0	24/02/2023	4000	4100.0	4000.0
1	Table2	1	Andhra Pradesh	Chittor	124	85	223	0	24/02/2023	3000	3500.0	3500.0
2	Table3	2	Andhra Pradesh	Chittor	124	85	224	0	24/02/2023	2300	2300.0	2300.0
3	Table4	3	Andhra Pradesh	Chittor	453	166	155	0	24/02/2023	1340	2000.0	1670.0
4	Table5	4	Andhra Pradesh	Chittor	570	166	192	0	24/02/2023	640	2160.0	1400.0

In [24]:
```
av=dataset["Modal_x0020_Price"].var()
```

In [25]:
```
sse=np.mean(np.mean(y_pred_test-y_train)**2)
bais=sse-av
bais=round(bais)
bais
```

Out[25]: -491045747

Test

In [26]:
```
from sklearn.linear_model import LinearRegression
l1=LinearRegression()
l1.fit(x_train,y_train)


y_pred_test1=l1.predict(x_test)
# y_reg1_score

from sklearn.metrics import mean_squared_error, r2_score
r2_score(y_test,y_pred_test1)*100
```

Out[26]: 99.98679475505587

In [27]:
```
sse=np.mean(np.mean(y_pred_test1-y_test)**2)
bais=sse-av
bais=round(bais)
bais
```

Out[27]: -491045743

Lets work on the Ridge Regression

In [28]:
```
x_train
```

Out[28]:
```
array([[-0.54122614, -0.92653143,  0.706977  , -0.36849895, -0.05451918,
        -0.04900792],
       [-0.23460616, -0.86947005, -1.20689185, -0.36849895, -0.15047167,
        -0.14694035]
```

Out[28]:
```
array([[-0.54122614, -0.92653143,  0.706977  , -0.36849895, -0.05451918,
        -0.04900792],
       [-0.23460616, -0.86947005, -1.20689185, -0.36849895, -0.15047167,
        -0.14694035],
       [-1.0778111 , -1.0406542 ,  0.706977  , -0.36849895, -0.12832879,
        -0.13235467],
       ...,
       [ 0.40221995, -0.92653143, -1.242554  , -0.36849895, -0.13054308,
        -0.1258953 ],
       [-1.18394879, -1.0406542 ,  1.18247237, -0.36849895, -0.10372559,
        -0.0906813 ],
       [ 0.23711688, -1.49714526,  0.706977  , -0.36849895, -0.09388431,
        -0.09901597]])
```

In [29]:
```
# Ridge
from sklearn.linear_model import Ridge
rg=Ridge(alpha=0.001)
rg.fit(x_train,y_train)

y_rg=rg.predict(x_test)
print(y_rg)


from sklearn.metrics import mean_squared_error, r2_score
r2_score(y_test,y_rg)*100
```

```
[ 859.4331714  2355.86218108 3748.277833   ... 1527.22482289 3800.48162692
  453.63873214]
```

Out[29]:
```
99.98679484429026
```

In [30]:
```
# Lasso
from sklearn.linear_model import Lasso
ls=Lasso(alpha=1.144)
```


```
print(y_rg)


from sklearn.metrics import mean_squared_error, r2_score
r2_score(y_test,y_rg)*100
```

```
[ 859.4331714  2355.86218108 3748.277833   ... 1527.22482289 3800.48162692
  453.63873214]
```

Out[29]:
```
99.98679484429026
```

In [30]:
```
# Lasso
from sklearn.linear_model import Lasso
ls=Lasso(alpha=1.144)
ls.fit(x_train,y_train)

y_ls=ls.predict(x_test)
print(y_ls)

from sklearn.metrics import mean_squared_error, r2_score
r2_score(y_test,y_ls)*100
```


```
[ 858.46957451 2357.07901754 3752.20740715 ... 1526.64551371 3794.20565908
  453.60840273]
```

Out[30]:
```
99.98624305650924
```

POLYNOMIAL REGRESSION:

CODE OF THE PROGRAM AND OUTPUT:

Polynomial Linear Regression

Polynomial regression is one of the machine learning algorithms used for making predictions. For example, it is widely applied to predict the spread rate of COVID-19 and other infectious diseases.

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

In [2]:
```
dataset = pd.read_csv('D:\\vit notes\\MCA Second Semester\\MachineLearning\\rank_salary.csv')
```

In [3]:
```
dataset.head()
```

Out[3]:

	Position	Level	Salary
0	Teaching Assistants	1	45000
1	Guest Faculty	2	50000
2	Contractual Faculty	3	60000
3	Assistant Professor	4	80000
4	Associate Professor	5	110000

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Position  10 non-null     object
 1   Level     10 non-null     int64
 2   Salary    10 non-null     int64
dtypes: int64(2), object(1)
memory usage: 368.0+ bytes
```

In [5]:
```
dataset.describe()
```

Out[5]:

	Level	Salary
count	10.00000	10.000000
mean	5.50000	249500.000000
std	3.02765	299373.883668
min	1.00000	45000.000000
25%	3.25000	65000.000000
50%	5.50000	130000.000000
75%	7.75000	275000.000000
max	10.00000	1000000.000000

In [6]:
```
X = dataset.iloc[:,1:-1]
```

In [6]:
```
X = dataset.iloc[:,1:-1]
Y = dataset.iloc[:,-1]
```

In [7]:
```
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
linear_reg.fit(X,Y)
```

Out[7]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

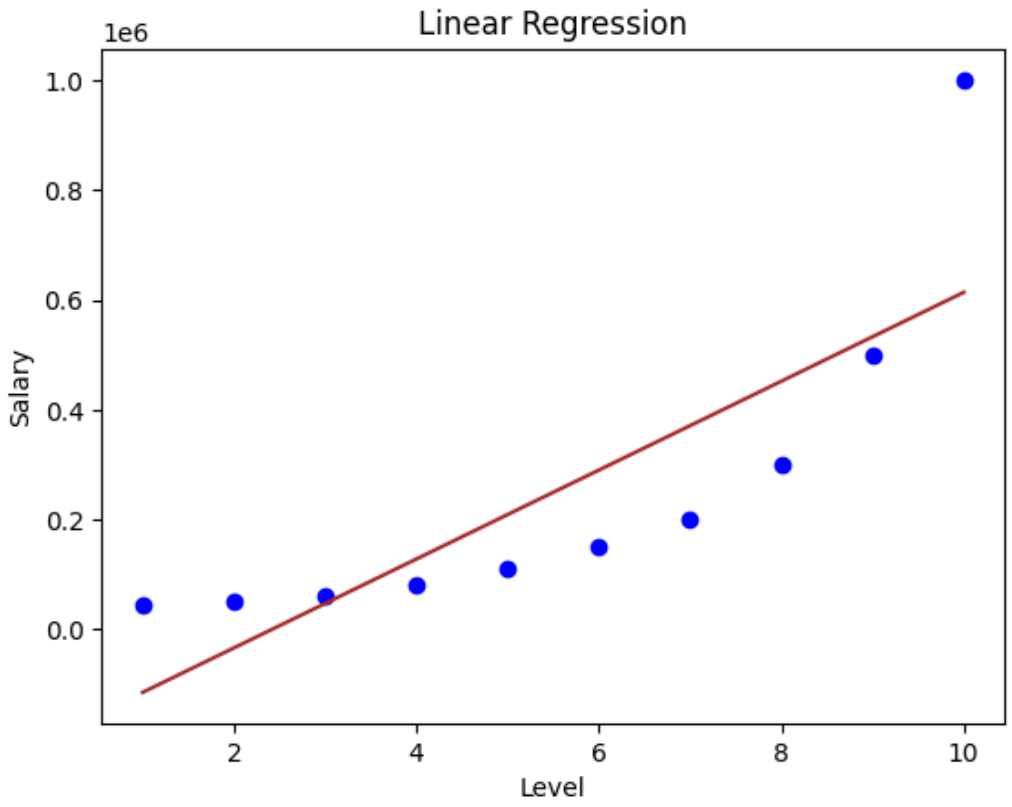
In [8]:
```
linear_reg.coef_
```

Out[8]:
```
array([80878.78787879])
```

In [9]:
```
linear_reg.intercept_
```

Out[9]:
```
-195333.33333333337
```

In [10]:
```
plt.scatter(X,Y,color = 'blue')
plt.plot(X, linear_reg.predict(X),color = 'brown')
plt.title('Linear Regression')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()
```



ı [11]:
```
dataset
```

ıt[11]:

	Position	Level	Salary

In [11]:
```
dataset
```

Out[11]:

	Position	Level	Salary
0	Teaching Assistants	1	45000
1	Guest Faculty	2	50000
2	Contractual Faculty	3	60000
3	Assistant Professor	4	80000
4	Associate Professor	5	110000
5	Professor	6	150000
6	Associate Dean	7	200000
7	Dean	8	300000
8	Vice Chancellor	9	500000
9	Chancellor	10	1000000

In [12]:
```
linear_reg.predict([[6.5]])
```

```
C:\Users\ayush\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have vali
d feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[12]:
```
array([330378.78787879])
```

In [13]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
lin_reg = LinearRegression()
```

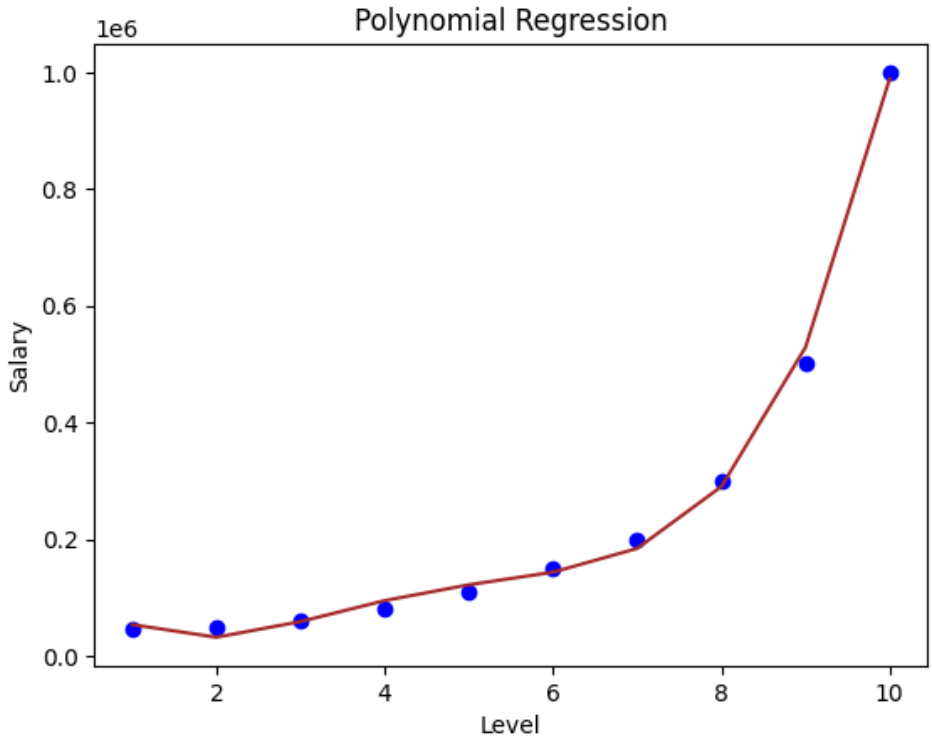
Out[12]:
```
array([330378.78787879])
```

In [13]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
lin_reg = LinearRegression()
lin_reg.fit(X_poly,Y)
```

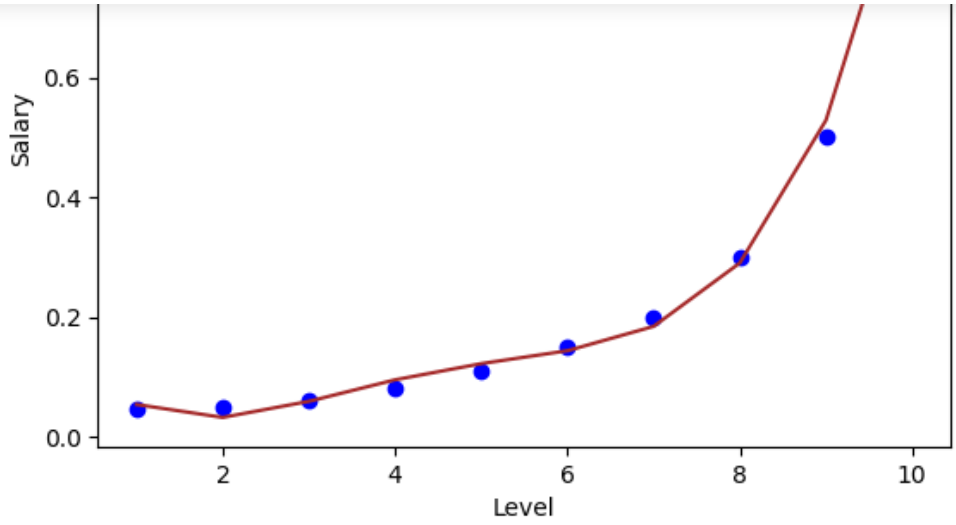
Out[13]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [14]:
```
plt.scatter(X,Y,color = 'blue')
plt.plot(X, lin_reg.predict(poly_reg.fit_transform(X)), color = 'brown')
plt.title('Polynomial Regression')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()
```



In [15]:
```
lin_reg.predict(poly_reg.fit_transform([[6.5]]))
```



In [15]:
```
lin_reg.predict(poly_reg.fit_transform([[6.5]]))
```

Out[15]:
```
array([158862.45265155])
```

In [ ]:
```

```