import warnings

warnings.filterwarnings('ignore')

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

# Correct File ID (Extracted from Your Link) file\_id = "1hPU9aVGKrNR-BDZfQ\_x3Rf6K-7HyxWDj"

url = f"https://drive.google.com/uc?id={file\_id}"

# Read the CSV File

df = pd.read\_csv(url)

# Display First 5 Rows print(df.head())

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| df.tail() | **TV** | **Radio** | **Newspaper** | **Sales** |
| **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 |
| df.info() |  |  |  |  |
| <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

df['TV'].unique()

 array([230.1, 44.5, 17.2, 151.5, 180.8, 8.7, 57.5, 120.2, 8.6,

199.8, 66.1, 214.7, 23.8, 97.5, 204.1, 195.4, 67.8, 281.4,

69.2, 147.3, 218.4, 237.4, 13.2, 228.3, 62.3, 262.9, 142.9,

240.1, 248.8, 70.6, 292.9, 112.9, 97.2, 265.6, 95.7, 290.7,

|  |  |
| --- | --- |
| 266.9, | 74.7, 43.1, 228. , 202.5, 177. , 293.6, 206.9, 25.1, |
| 175.1, | 89.7, 239.9, 227.2, 66.9, 100.4, 216.4, 182.6, 262.7, |
| 198.9, | 7.3, 136.2, 210.8, 210.7, 53.5, 261.3, 239.3, 102.7, |
| 131.1, | 69. , 31.5, 139.3, 216.8, 199.1, 109.8, 26.8, 129.4, |
| 213.4, | 16.9, 27.5, 120.5, 5.4, 116. , 76.4, 239.8, 75.3, |

68.4, 213.5, 193.2, 76.3, 110.7, 88.3, 134.3, 28.6, 217.7,

250.9, 107.4, 163.3, 197.6, 184.9, 289.7, 135.2, 222.4, 296.4,

|  |  |  |
| --- | --- | --- |
| 280.2, 187.9, 238.2, 137.9, | 25. , 90.4, 13.1, 255.4, | 225.8, |
| 241.7, 175.7, 209.6, 78.2, | 75.1, 139.2, 125.7, 19.4, | 141.3, |
| 18.8, 224. , 123.1, 229.5, | 87.2, 7.8, 80.2, 220.3, | 59.6, |
| 0.7, 265.2, 8.4, 219.8, | 36.9, 48.3, 25.6, 273.7, | 43. , |
| 73.4, 193.7, 220.5, 104.6, | 96.2, 140.3, 243.2, 38. , | 44.7, |
| 280.7, 121. , 171.3, 187.8, | 4.1, 93.9, 149.8, 11.7, | 131.7, |
| 172.5, 85.7, 188.4, 163.5, 117.2, 234.5, 17.9, 206.8, 215.4, | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 284.3, 50. , | 164.5, | 19.6, 168.4, 276.9, | 248.4, 170.2, 276.7, |
| 165.6, 156.6, | 218.5, | 56.2, 287.6, 253.8, | 205. , 139.5, 191.1, |
| 286. , 18.7, | 39.5, | 75.5, 166.8, 149.7, | 38.2, 94.2, 283.6, |
| 232.1]) |  |  |  |

df['Newspaper'].unique()



|  |  |  |  |
| --- | --- | --- | --- |
| array([ 69.2, 45.1, | 69.3, | 58.5, 58.4, 75. , | 23.5, 11.6, 1. , |
| 21.2, 24.2, | 4. , | 65.9, 7.2, 46. , | 52.9, 114. , 55.8, |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 18.3, | 19.1, | 53.4, | 49.6, | 26.2, | 19.5, | 12.6, | 22.9, | 40.8, |
| 43.2, | 38.6, | 30. , | 0.3, | 7.4, | 8.5, | 5. , | 45.7, | 35.1, |
| 32. , | 31.6, | 38.7, | 1.8, | 26.4, | 43.3, | 31.5, | 35.7, | 18.5, |
| 49.9, | 36.8, | 34.6, | 3.6, | 39.6, | 58.7, | 15.9, | 60. , | 41.4, |
| 16.6, | 37.7, | 9.3, | 21.4, | 54.7, | 27.3, | 8.4, | 28.9, | 0.9, |
| 2.2, | 10.2, | 11. , | 27.2, | 31.7, | 19.3, | 31.3, | 13.1, | 89.4, |
| 20.7, | 14.2, | 9.4, | 23.1, | 22.3, | 36.9, | 32.5, | 35.6, | 33.8, |
| 65.7, | 16. , | 63.2, | 73.4, | 51.4, | 33. , | 59. , | 72.3, | 10.9, |
| 5.9, | 22. , | 51.2, | 45.9, | 49.8, | 100.9, | 17.9, | 5.3, | 29.7, |
| 23.2, | 25.6, | 5.5, | 56.5, | 2.4, | 10.7, | 34.5, | 52.7, | 14.8, |
| 79.2, | 46.2, | 50.4, | 15.6, | 12.4, | 74.2, | 25.9, | 50.6, | 9.2, |
| 3.2, | 43.1, | 8.7, | 43. , | 2.1, | 65.6, | 59.7, | 20.5, | 1.7, |
| 12.9, | 75.6, | 37.9, | 34.4, | 38.9, | 9. , | 44.3, | 11.9, | 20.6, |
| 37. , | 48.7, | 9.5, | 5.7, | 50.5, | 24.3, | 45.2, | 30.7, | 49.3, |
| 5.4, | 84.8, | 21.6, | 19.4, | 57.6, | 6.4, | 18.4, | 47.4, | 17. , |
| 12.8, | 41.8, | 20.3, | 35.2, | 23.7, | 17.6, | 8.3, | 27.4, | 71.8, |
| 19.6, | 26.6, | 18.2, | 3.7, | 23.4, | 5.8, | 6. , | 13.8, | 8.1, |
| 66.2]) |  |  |  |  |  |  |  |  |

df.isnull().sum()

# 0

**TV** 0

**Radio** 0

# Newspaper 0

**Sales** 0

**dtype:** int64

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| df.describe() | **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 |

from scipy import stats x=df['TV'].mean()

y=df['TV'].median()

z=stats.mode(df['TV'])

print("Mean:",x) print("Median",y) print("Mode:",z)

 Mean: 147.0425

Median 149.75

Mode: ModeResult(mode=np.float64(17.2), count=np.int64(2))

from scipy import stats x=df['Radio'].mean()

y=df['Radio'].median()

z=stats.mode(df['Radio']) print("Mean:",x)

print("Median",y) print("Mode:",z)

 Mean: 23.264000000000006

Median 22.9

Mode: ModeResult(mode=np.float64(4.1), count=np.int64(3))

from scipy import stats x=df['Radio'].mean()

y=df['Radio'].median()

z=stats.mode(df['Radio']) print("Mean:",x)

print("Median",y) print("Mode:",z)

 Mean: 23.264000000000006

Median 22.9

Mode: ModeResult(mode=np.float64(4.1), count=np.int64(3))

from scipy import stats x=df['Newspaper'].mean()

y=df['Newspaper'].median()

z=stats.mode(df['Newspaper']) print("Mean:",x)

print("Median",y) print("Mode:",z)

 Mean: 30.553999999999995

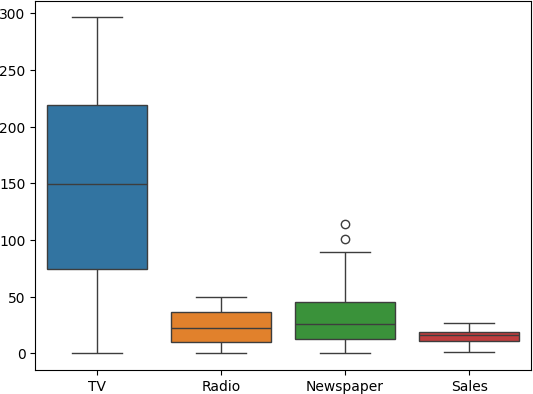
Median 25.75

Mode: ModeResult(mode=np.float64(8.7), count=np.int64(3))

import matplotlib.pyplot as plt import seaborn as sns

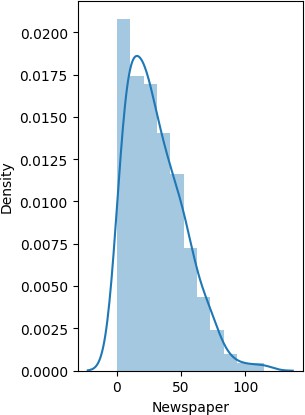
sns.boxplot(df)

 <Axes: >



plt.subplot(1,2,2)

plt2 = sns.distplot(df['Newspaper'])



Start coding or ge nerate with AI.

data=df['Newspaper']

lower\_limit = np.mean(data) - 2\* np.std(data) upper\_limit = np.mean(data) + 2\* np.std(data) outliers = []

def detect\_outliers(data):

for i in data:

if (i<lower\_limit or i>upper\_limit): outliers.append(i)

return outliers

outliers = detect\_outliers(data) outliers

 [75.0, 114.0, 89.4, 100.9, 79.2, 74.2, 75.6, 84.8]

q1,q3 = np.percentile(data,[25,75]) print('Quartile 1: ',q1)

print('Quartile 2: ',q3) iqr = q3 - q1

print('Inter Quartile Range: ',iqr) lower\_bound = q1 - (iqr\*1.5)

upper\_bound = q3 + (iqr\*1.5) outliers = []

def detect\_outliers(data1):

for i in data1:

if (i < lower\_bound or i > upper\_bound): outliers.append(i)

return outliers

outliers = detect\_outliers(data) outliers

 Quartile 1: 12.75

Quartile 2: 45.1

Inter Quartile Range: 32.35 [114.0, 100.9]

outliers = []

def detect\_outliers(data):

threshold = 2.0

mean = np.mean(data) std = np.std(data)

for x in data:

z\_score = np.abs((x - mean)/std) if np.ceil(z\_score) > threshold:

outliers.append(x) return outliers

outliers\_pts = detect\_outliers(data) outliers\_pts

 [75.0, 114.0, 89.4, 100.9, 79.2, 74.2, 75.6, 84.8]

plt.figure(figsize = (16, 10)) import seaborn as sns

sns.heatmap(df.corr(),annot=True)

<Axes: >



dfc=df.corr()

dfc["Sales"].sort\_values(ascending=False)



|  |  |
| --- | --- |
|  | **Sales** |
| **Sales** | 1.000000 |
| **TV** | 0.901208 |
| **Radio** | 0.349631 |
| **Newspaper** | 0.157960 |
| **dtype:** float64 |  |

encoded\_df = pd.get\_dummies(df) print(encoded\_df.shape)

print(encoded\_df.columns) print(encoded\_df.dtypes)

 (200, 4)

Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object') TV float64

Radio float64

Newspaper float64

Sales float64 dtype: object

encoded\_df.head()

# 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

Generate code with encoded\_df

 View recommended plots

New interactive sheet

Next steps:

x = encoded\_df.drop('Sales',axis=1)

from sklearn.preprocessing import MinMaxScaler norm = MinMaxScaler().fit(x)

x = norm.transform(x) print(x.min())

print(x.max())

 0.0

1.0

Start coding or ge nerate with AI.

from sklearn.model\_selection import train\_test\_split x = df['TV'].values.reshape(-1, 1)

y = df['Sales'].values.reshape(-1, 1)

train\_x,test\_x,train\_y,test\_y = train\_test\_split(x, y, test\_size=0.2,shuffle=True) print(train\_x.shape)

print(test\_x.shape)

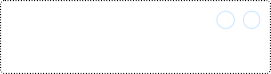
 (160, 1)

(40, 1)

from sklearn.linear\_model import LinearRegression model = LinearRegression()

model.fit(train\_x, train\_y)





▾ LinearRegression i [?](https://scikit-learn.org/1.6/modules/generated/sklearn.linear_model.LinearRegression.html)

LinearRegression()

x\_mean = np.mean(x) y\_mean = np.mean(y)

numerator = np.sum((x - x\_mean) \* (y - y\_mean)) denominator = np.sum((x - x\_mean) \*\* 2)

m = numerator / denominator c = y\_mean - m \* x\_mean

print("Slope:",m)

print("Intercept:",c)

 Slope: 0.055464770469558854

Intercept: 6.974821488229894

y\_pred = m \* x + c y\_pred=y\_pred[:160]

from sklearn.metrics import r2\_score from sklearn import metrics as mt

from sklearn.metrics import mean\_squared\_error score=r2\_score(train\_y,y\_pred)

rms = np.sqrt(mean\_squared\_error(train\_y,y\_pred)) print("R2 score is:",score)

print("RMS value is:",rms)

print("The Mean Absolute Error of model is:", np.round(mt.mean\_absolute\_error(train\_y,y\_pred ),2))

 R2 score is: -0.8301678240980328 RMS value is: 7.139707921157611

The Mean Absolute Error of model is: 5.82

x=x[0:160] y=y[0:160]

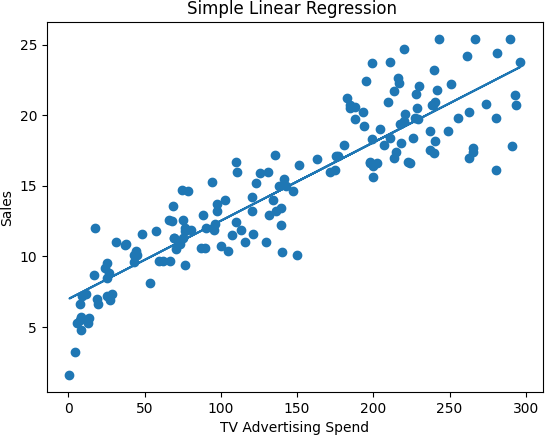
plt.scatter(x,y)

plt.plot(x, y\_pred)

plt.xlabel("TV Advertising Spend") plt.ylabel("Sales")

plt.title("Simple Linear Regression ")

plt.show()



from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error print("Mean absolute error:",mean\_absolute\_error(train\_y,train\_x))

 Mean absolute error: 134.699375

#POLYNOMIAL REGRESSION

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error from sklearn.preprocessing import PolynomialFeatures

trans = PolynomialFeatures(degree=4) x = trans.fit\_transform(x)

print(x.shape)

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,shuffle=True) x\_train.shape,y\_train.shape, x\_test.shape,y\_test.shape

from sklearn.linear\_model import LinearRegression lm=LinearRegression()

lm.fit(x\_train,y\_train)

y\_train\_pred=lm.predict(x\_train)

print('MAE of Polynomial regressions is .. ',mean\_absolute\_error(y\_train,y\_train\_pred))

y\_test\_pred=lm.predict(x\_test)

print('MAE of Polynomial regressions is .. ',mean\_absolute\_error(y\_test,y\_test\_pred))

 (200, 35)

MAE of Polynomial regressions is .. 0.8284885183241801 MAE of Polynomial regressions is .. 0.9608435565128588

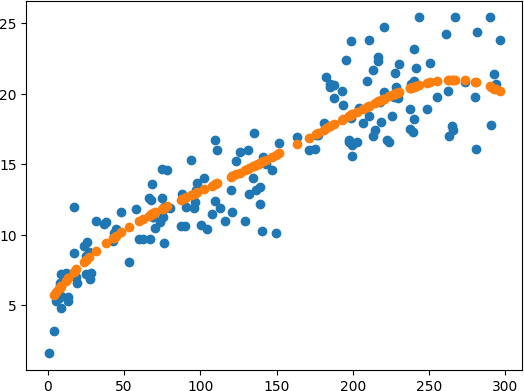
print(x\_train[:,1].shape)  (128,)

print(y\_train\_pred.flatten().shape)  (128,)

plt.scatter(x[:,1],y)

plt.scatter(x\_train[:,1].flatten(),y\_train\_pred.flatten())

 <matplotlib.collections.PathCollection at 0x7ab0e0958990>



x=df[['TV','Newspaper','Radio']] y=df["Sales"]

print(y)

|  |  |
| --- | --- |
| 0 | 22.1 |
| 1 | 10.4 |
| 2 | 12.0 |
| 3 | 16.5 |
| 4 | 17.9 |
| 195 | ...  7.6 |
| 196 | 14.0 |
| 197 | 14.8 |
| 198 | 25.5 |
| 199 | 18.4 |

Name: Sales, Length: 200, dtype: float64

from sklearn.model\_selection import train\_test\_split

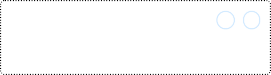
x\_train,x\_test, y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2) x\_train.shape,y\_train.shape, x\_test.shape,y\_test.shape

 ((160, 3), (160,), (40, 3), (40,))

from sklearn.linear\_model import LinearRegression lm=LinearRegression()

lm.fit(x\_train,y\_train)





▾ LinearRegression i [?](https://scikit-learn.org/1.6/modules/generated/sklearn.linear_model.LinearRegression.html)

LinearRegression()

#Prediction for train data

y\_train\_pred=lm.predict(x\_train)

mean\_absolute\_error(y\_train,y\_train\_pred)

1 182126671974186

y\_pred=y\_pred[:160]

from sklearn.metrics import r2\_score

from sklearn import metrics as mt

from sklearn.metrics import mean\_squared\_error

score=r2\_score(train\_y,y\_pred)

rms = np.sqrt(mean\_squared\_error(train\_y,y\_pred))

print("R2 score is:",score)

print("RMS value is:",rms)

print("The Mean Absolute Error of model is:", np.round(mt.mean\_absolute\_error(train\_y,y\_pred ),2))

R2 score is: -0.8301678240980328

RMS value is: 7.139707921157611

The Mean Absolute Error of model is: 5.82

**OBSERVATION:**

MAE FOR SIMPLE LINEAR REGRESSION:6.17

MAE FOR MULTIPLE LINEAR REGRESSION: 5.82

MAE FOR POLYNOMIAL REGRESSION:0.96

**CONCLUSION:** Out of all regression models, Polynomial regression is best because the errors are **less in polynomial regression**.

Hence, For the given dataset polynomial Regression provides the best fit line with minimum Mean Absolute Error.