7_FEB_Regression_analysis_hw

February 9, 2025

1 Experiment IV

2 Tufan Kundu

3 Reg no: 24MDT0184

3.1 Importing the libraries

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
```

3.2 Reading the dataset

[7]:
```
df = pd.read_excel(r"D:\study material\VIT_Data_Science\Winter_Sem\Regression␣
 ↪Analysis and Predictive Models Lab\7_feb\advertising.xlsx")
df
```

[7]:
```
        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 x 4 columns]
```

[8]:
```
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
```

3.3 Perform Linear Regression using sklearn:

- Define TV as the independent variable (X) and Sales as the dependent variable (y).

[10]:
```
x_tv = df['TV'].values.reshape(-1,1)
y = df['Sales'].values

model_tv = LinearRegression()
model_tv.fit(x_tv,y)
y_pred_tv = model_tv.predict(x_tv)
residuals_tv = y - y_pred_tv
```

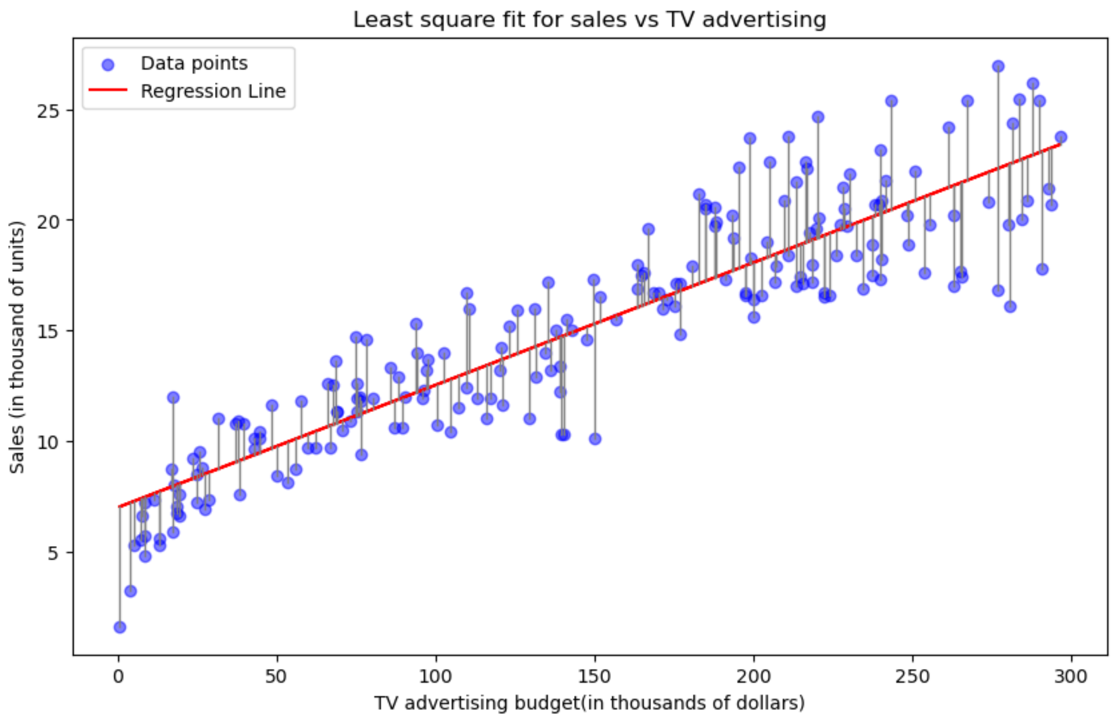
3.3.1 Scatter plot of TV vs Sales along with the regression line and the residuals

[12]:
```
## creating plot to visualize the regression line

plt.figure(figsize = (10,6))
plt.scatter(x = x_tv, y = y, color = 'blue', label = 'Data points', alpha = 0.5)
plt.plot(x_tv,y_pred_tv, color = 'red', label = "Regression Line")

## Adding lines showing the residuals(the vertical distance between actual and␣
 ↪predicted value)
for i in range(len(x_tv)):
    plt.plot([x_tv[i],x_tv[i]],[y[i],y_pred_tv[i]], color = 'grey', lw=1)

plt.title("Least square fit for sales vs TV advertising")
plt.xlabel("TV advertising budget(in thousands of dollars)")
plt.ylabel("Sales (in thousand of units)")
plt.legend()
plt.show()
```



3.4 Perform Linear Regression using statsmodels:

[14]:
```
import statsmodels.api as sm
x_tv = df['TV']
y = df['Sales']

# adds a constant to the independent variable
x_with_const = sm.add_constant(x_tv) # adds a column of ones to x for the␣
↪intercept form

# Fit the OLS regression model using stats model
model_sm_tv = sm.OLS(y,x_with_const).fit()

print(model_sm_tv.summary())
# get the 95% confidence interval for the model coefficients (  and  )

confidence_intervals_tv = model_sm_tv.conf_int(alpha = 0.05)

# print the confidence intervals for the intercept  and coefficient

print("95% confidence interval for  :\n",confidence_intervals_tv.iloc[0])
print("95% confidence interval for  1:\n",confidence_intervals_tv.iloc[1])
```

```
standard_error_tv = model_sm_tv.bse

print("Standard error for  (Intercept):",standard_error_tv.iloc[0])
print("Standard error for  1:",standard_error_tv.iloc[1])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.812
Model:                            OLS   Adj. R-squared:                  0.811
Method:                 Least Squares   F-statistic:                     856.2
Date:                Sun, 09 Feb 2025   Prob (F-statistic):           7.93e-74
Time:                        11:02:44   Log-Likelihood:                -448.99
No. Observations:                 200   AIC:                             902.0
Df Residuals:                     198   BIC:                             908.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.9748      0.323     21.624      0.000       6.339       7.611
TV             0.0555      0.002     29.260      0.000       0.052       0.059
==============================================================================
Omnibus:                        0.013   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.993   Jarque-Bera (JB):                0.043
Skew:                          -0.018   Prob(JB):                        0.979
Kurtosis:                       2.938   Cond. No.                         338.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
95% confidence interval for  :
 0    6.338740
1    7.610903
Name: const, dtype: float64
95% confidence interval for  1:
 0    0.051727
1    0.059203
Name: TV, dtype: float64
Standard error for  (Intercept): 0.3225534848524013
Standard error for  1: 0.001895551178040241
```

3.5 Simple Linear Regression for Radio advertising

[16]:
```
x_radio = df['Radio'].values.reshape(-1,1)
y = df['Sales'].values
```

```
model_radio = LinearRegression()
model_radio.fit(x_radio,y)
y_pred_radio = model_radio.predict(x_radio)
residuals_radio = y - y_pred_radio
```

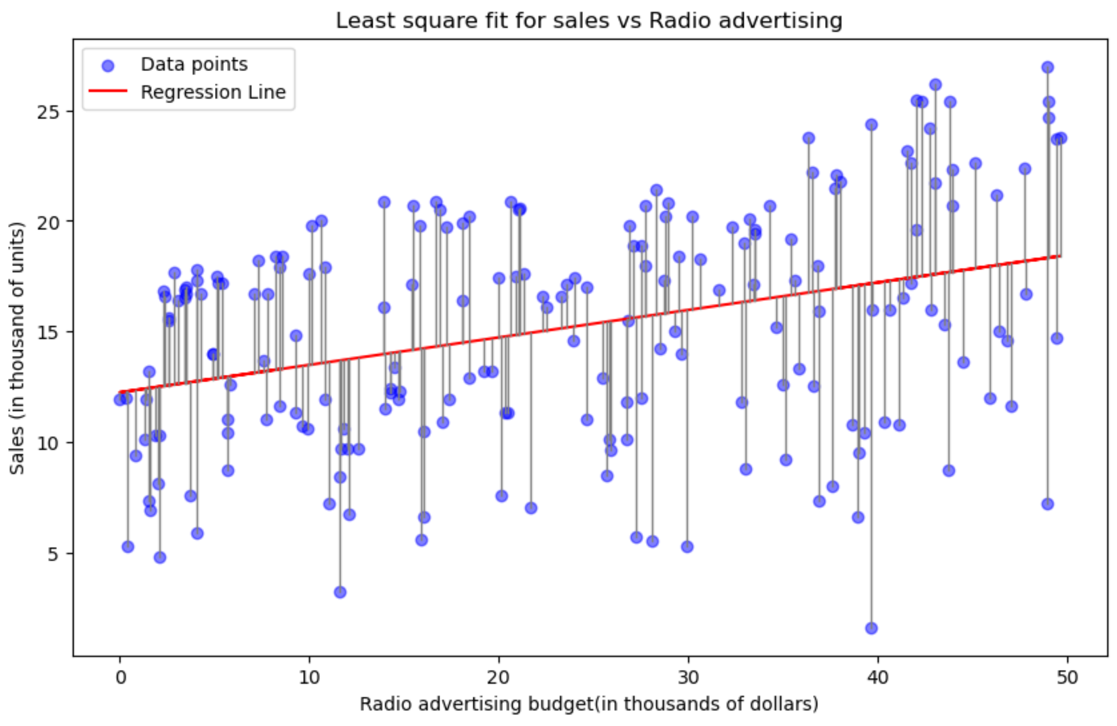
3.5.1 Scatter plot of Radio vs Sales along with the regression line and the residuals

[18]:
```
## creating plot to visualize the regression line

plt.figure(figsize = (10,6))
plt.scatter(x = x_radio, y = y, color = 'blue', label = 'Data points', alpha =␣
↪0.5)
plt.plot(x_radio,y_pred_radio, color = 'red', label = "Regression Line")

## Adding lines showing the residuals(the vertical distance between actual and␣
↪predicted value)
for i in range(len(x_radio)):
    plt.plot([x_radio[i],x_radio[i]],[y[i],y_pred_radio[i]], color = 'grey',␣
↪lw=1)

plt.title("Least square fit for sales vs Radio advertising")
plt.xlabel("Radio advertising budget(in thousands of dollars)")
plt.ylabel("Sales (in thousand of units)")
plt.legend()
plt.show()
```



3.5.2 Performing Linear regression using stats model

[20]:
```python
import statsmodels.api as sm
x_radio = df['Radio']
y = df['Sales']

# adds a constant to the independent variable
x_with_const = sm.add_constant(x_radio) # adds a column of ones to x for the␣
↪intercept form

# Fit the OLS regression model using stats model
model_sm_radio = sm.OLS(y,x_with_const).fit()

print(model_sm_radio.summary())

# get the 95% confidence interval for the model coefficients (   and   )

confidence_intervals_radio = model_sm_radio.conf_int(alpha = 0.05)

# print the confidence intervals for the intercept  and coefficient

print("95% confidence interval for  :\n",confidence_intervals_radio.iloc[0])
print("95% confidence interval for  1:\n",confidence_intervals_radio.iloc[1])

standard_error_radio = model_sm_radio.bse

print("Standard error for  (Intercept):",standard_error_radio.iloc[0])
print("Standard error for  1:",standard_error_radio.iloc[1])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.122
Model:                            OLS   Adj. R-squared:                  0.118
Method:                 Least Squares   F-statistic:                     27.57
Date:                Sun, 09 Feb 2025   Prob (F-statistic):           3.88e-07
Time:                        11:02:47   Log-Likelihood:                -603.18
No. Observations:                 200   AIC:                             1210.
Df Residuals:                     198   BIC:                             1217.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.2357      0.653     18.724      0.000      10.947      13.524
Radio          0.1244      0.024      5.251      0.000       0.078       0.171
```

```
==============================================================================
Omnibus:                       11.077   Durbin-Watson:                   2.018
Prob(Omnibus):                  0.004   Jarque-Bera (JB):                9.124
Skew:                          -0.433   Prob(JB):                       0.0104
Kurtosis:                       2.414   Cond. No.                         51.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
95% confidence interval for  :
 0    10.947036
1    13.524408
Name: const, dtype: float64
95% confidence interval for  1:
 0    0.077703
1    0.171161
Name: Radio, dtype: float64
Standard error for  (Intercept): 0.653486294381421
Standard error for  1: 0.023696033393257657
```

3.6 Simple Linear Regression for Newspaper advertising

[22]:
```
x_np = df['Newspaper'].values.reshape(-1,1)
y = df['Sales'].values

model_np = LinearRegression()
model_np.fit(x_np,y)
y_pred_np = model_np.predict(x_np)
residuals_np = y - y_pred_np
```

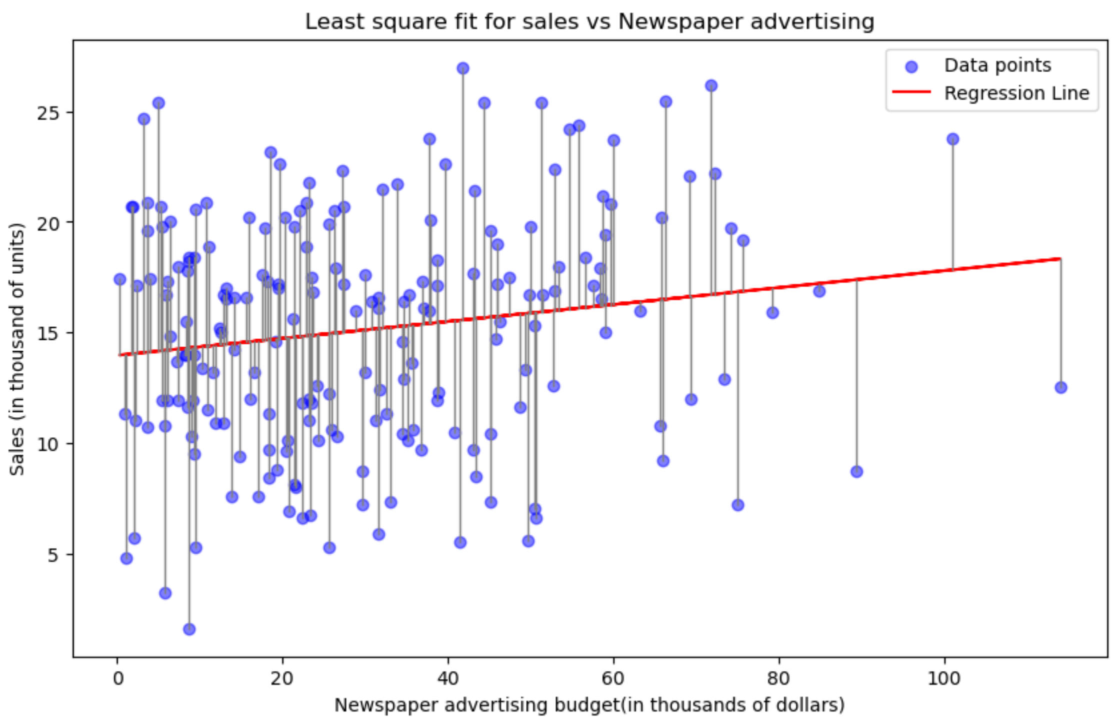
3.6.1 Scatter plot of Newspaper vs Sales along with the regression line and the residuals

[24]:
```
## creating plot to visualize the regression line

plt.figure(figsize = (10,6))
plt.scatter(x = x_np, y = y, color = 'blue', label = 'Data points', alpha = 0.5)
plt.plot(x_np,y_pred_np, color = 'red', label = "Regression Line")

## Adding lines showing the residuals(the vertical distance between actual and␣
 ↪predicted value)
for i in range(len(x_np)):
    plt.plot([x_np[i],x_np[i]],[y[i],y_pred_np[i]], color = 'grey', lw=1)

plt.title("Least square fit for sales vs Newspaper advertising")
plt.xlabel("Newspaper advertising budget(in thousands of dollars)")
plt.ylabel("Sales (in thousand of units)")
plt.legend()
plt.show()
```



3.6.2 Performing Linear regression using stats model

[26]:
```
import statsmodels.api as sm
x_np = df['Newspaper']
y = df['Sales']

# adds a constant to the independent variable
x_with_const = sm.add_constant(x_np) # adds a column of ones to x for the␣
↪intercept form

# Fit the OLS regression model using stats model
model_sm_np = sm.OLS(y,x_with_const).fit()

print(model_sm_np.summary())

# get the 95% confidence interval for the model coefficients (  and  )

confidence_intervals_np = model_sm_np.conf_int(alpha = 0.05)
```

```python
# print the confidence intervals for the intercept  and coefficient

print("95% confidence interval for  :\n",confidence_intervals_np.iloc[0])
print("95% confidence interval for  1:\n",confidence_intervals_np.iloc[1])

standard_error_np = model_sm_np.bse

print("Standard error for  (Intercept):",standard_error_np.iloc[0])
print("Standard error for  1:",standard_error_np.iloc[1])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.025
Model:                            OLS   Adj. R-squared:                  0.020
Method:                 Least Squares   F-statistic:                     5.067
Date:                Sun, 09 Feb 2025   Prob (F-statistic):             0.0255
Time:                        11:02:48   Log-Likelihood:                -613.69
No. Observations:                 200   AIC:                             1231.
Df Residuals:                     198   BIC:                             1238.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         13.9595      0.638     21.870      0.000      12.701      15.218
Newspaper      0.0383      0.017      2.251      0.025       0.005       0.072
==============================================================================
Omnibus:                       10.252   Durbin-Watson:                   2.017
Prob(Omnibus):                  0.006   Jarque-Bera (JB):                4.808
Skew:                          -0.111   Prob(JB):                       0.0903
Kurtosis:                       2.273   Cond. No.                         64.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
95% confidence interval for  :
 0    12.700833
1    15.218265
Name: const, dtype: float64
95% confidence interval for  1:
 0    0.004749
1    0.071899
Name: Newspaper, dtype: float64
Standard error for  (Intercept): 0.6382885131624155
Standard error for  1: 0.017025666500722073
```

3.6.3 Comparison of the three models

[28]:
```
print(" 0(Intercept) for TV:",model_tv.intercept_)
print(" 1(Coefficient) for TV:",model_tv.coef_[0])
print()
print(" 0(Intercept) for Radio:",model_radio.intercept_)
print(" 1(Coefficient) for Radio:",model_radio.coef_[0])
print()
print(" 0(Intercept) for Newspaper:",model_np.intercept_)
print(" 1(Coefficient) for Newspaper:",model_np.coef_[0])
```

```
 0(Intercept) for TV: 6.974821488229891
 1(Coefficient) for TV: 0.05546477046955874

 0(Intercept) for Radio: 12.235721966369233
 1(Coefficient) for Radio: 0.12443165550338577

 0(Intercept) for Newspaper: 13.959548653554414
 1(Coefficient) for Newspaper: 0.03832399510524274
```

[29]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("R2 score for TV model:",r2_score(y,y_pred_tv))
print("MSE for TV model:",mean_squared_error(y,y_pred_tv))
print()
print("R2 score for Radio model:",r2_score(y,y_pred_radio))
print("MSE for Radio model:",mean_squared_error(y,y_pred_radio))
print()
print("R2 score for Newspaper model:",r2_score(y,y_pred_np))
print("MSE for Newspaper model:",mean_squared_error(y,y_pred_np))
```

```
R2 score for TV model: 0.8121757029987414
MSE for TV model: 5.2177438977951285

R2 score for Radio model: 0.1222419039947863
MSE for Radio model: 24.384049466937633

R2 score for Newspaper model: 0.024951369862864836
MSE for Newspaper model: 27.086772697557045
```

3.7 Summarized Insights:

- **TV advertising has the highest impact on sales**, explaining **81.2% of the variation in sales** with the lowest prediction error (MSE = 5.22).
- **Radio advertising has a weak impact on sales**, explaining only **12.2% of the variation**, with higher prediction errors (MSE = 24.38).
- **Newspaper advertising has minimal influence on sales**, explaining just **2.5% of the**

variation, with the highest prediction error (MSE = 27.09).

- TV should be the primary focus for advertising investments as it provides the best return.
- Radio may be considered as a supplementary channel.
- Newspaper advertising is not a cost-effective medium for increasing sales and should be reconsidered.