CB.EN.U4CSE21450 LAB EVAL 3

ALL LIBRARIES NEEDED

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix

# Load the dataset
```

QUESTION 1

In [ ]:
```
df1=pd.read_csv('data1.csv')
df1=df1[['TV','Radio','Newspaper','Sales']]
df1
```

Out[ ]:

	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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

In [ ]:
```
print(df1.isnull().sum())
df1_object_rows = df1.select_dtypes(include='object')
print(df1_object_rows)
```

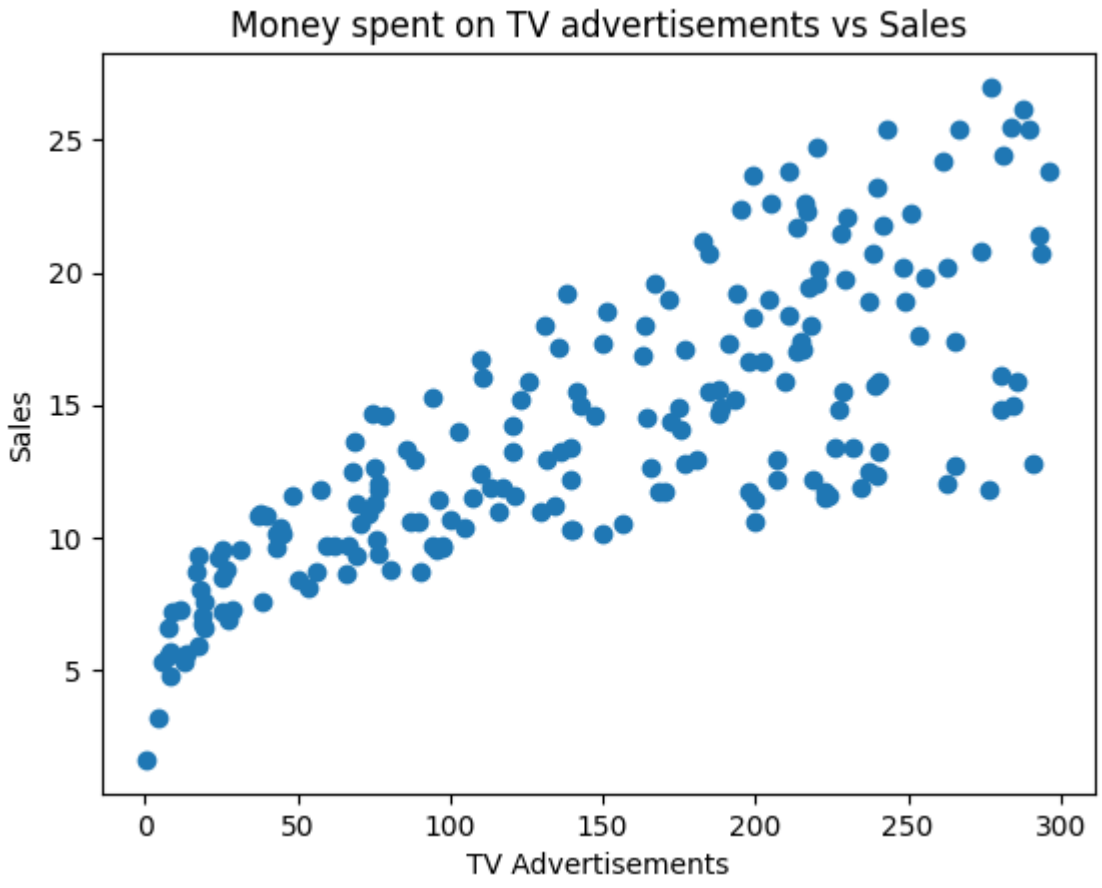
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 4
1, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 6
1, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 8
1, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, ...]

[200 rows x 0 columns]
```

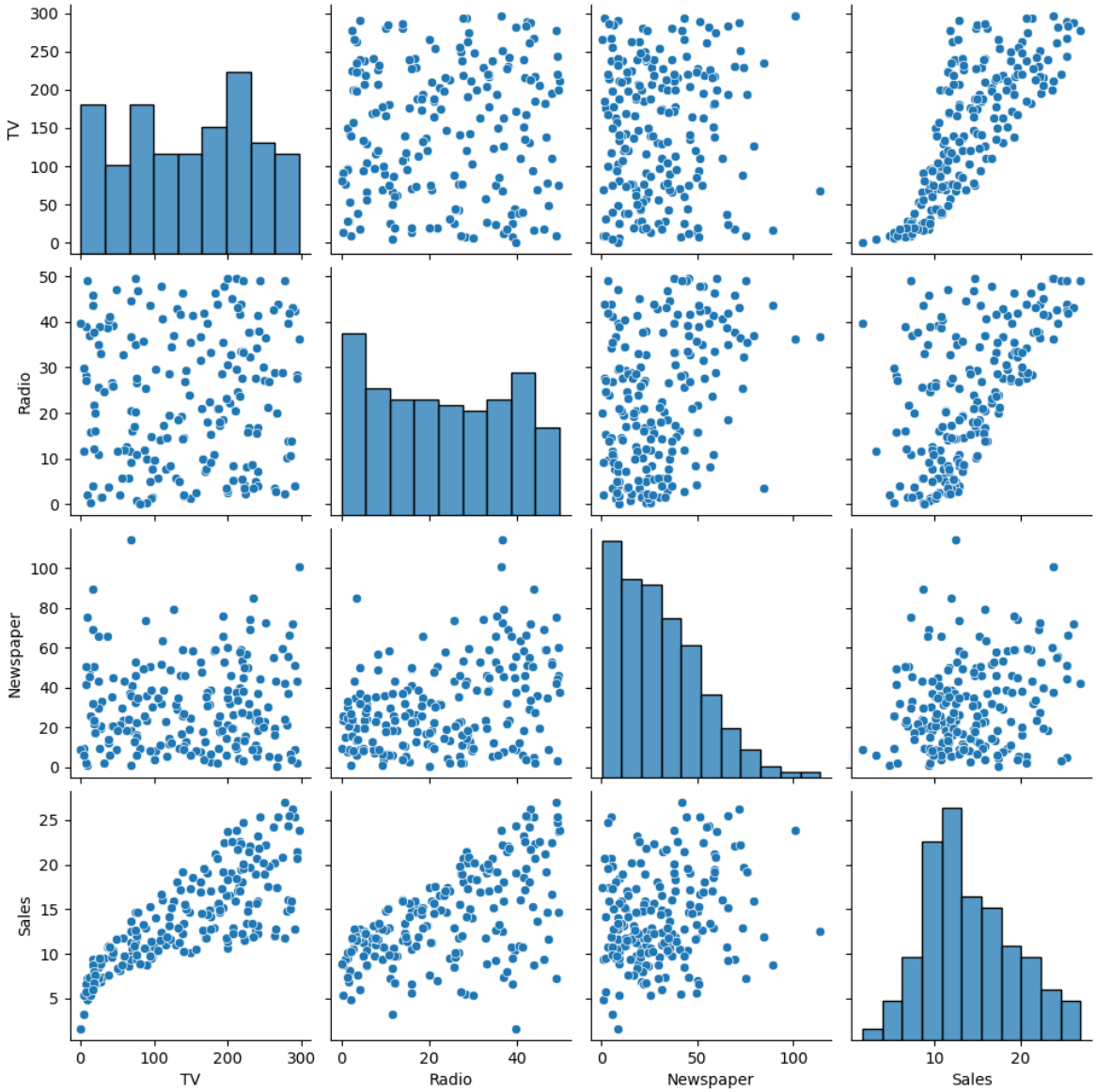
SO, NO NULL VALUES AND NO DATA WITH DATA TYPES = OBJECTS, SO NO NEED FOR PREPROCESSING

1.1 Scatter plot of money spent on TV advertisements versus Sales

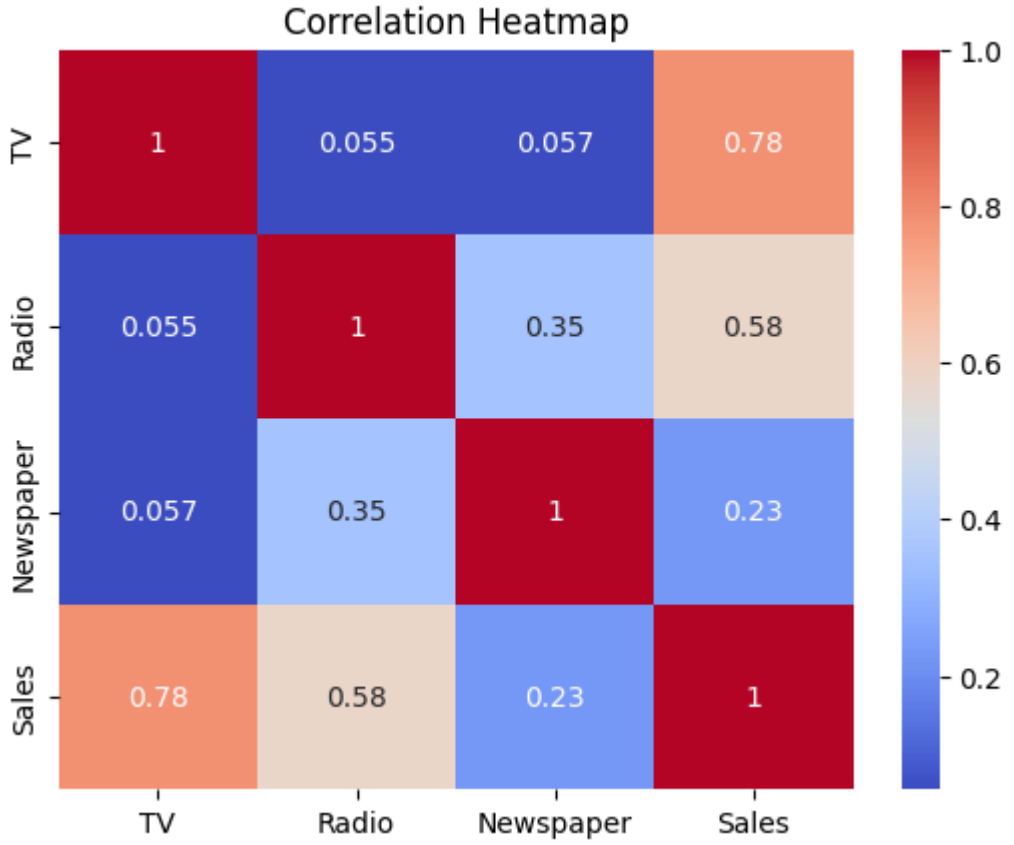
In [ ]:
```
plt.scatter(df1['TV'], df1['Sales'])
plt.xlabel('TV Advertisements')
plt.ylabel('Sales')
plt.title('Money spent on TV advertisements vs Sales')
plt.show()
```



In [ ]:
```
sns.pairplot(df1)
plt.show()
```



In [ ]:
```
sns.heatmap(df1.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



we can see that correlation for the same columns is one as it is supposed to be and the correlation between the columns sales and tv is the highest compared to radio and tv meaning that the correlation between those two is very very less.

1.2 Linear Regression model based on money spent on TV advertisements versus Sales

In [ ]:
```
X = df1[['TV']]
y = df1['Sales']
model = LinearRegression()
model.fit(X, y)
```

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

1.3 Predict sales based on money spent on TV advertisements

In [ ]:
```
new_TV_advertisements = np.array([100, 200, 300]).reshape(-1, 1)
predicted_sales = model.predict(new_TV_advertisements)
print("Predicted Sales:", predicted_sales)
```

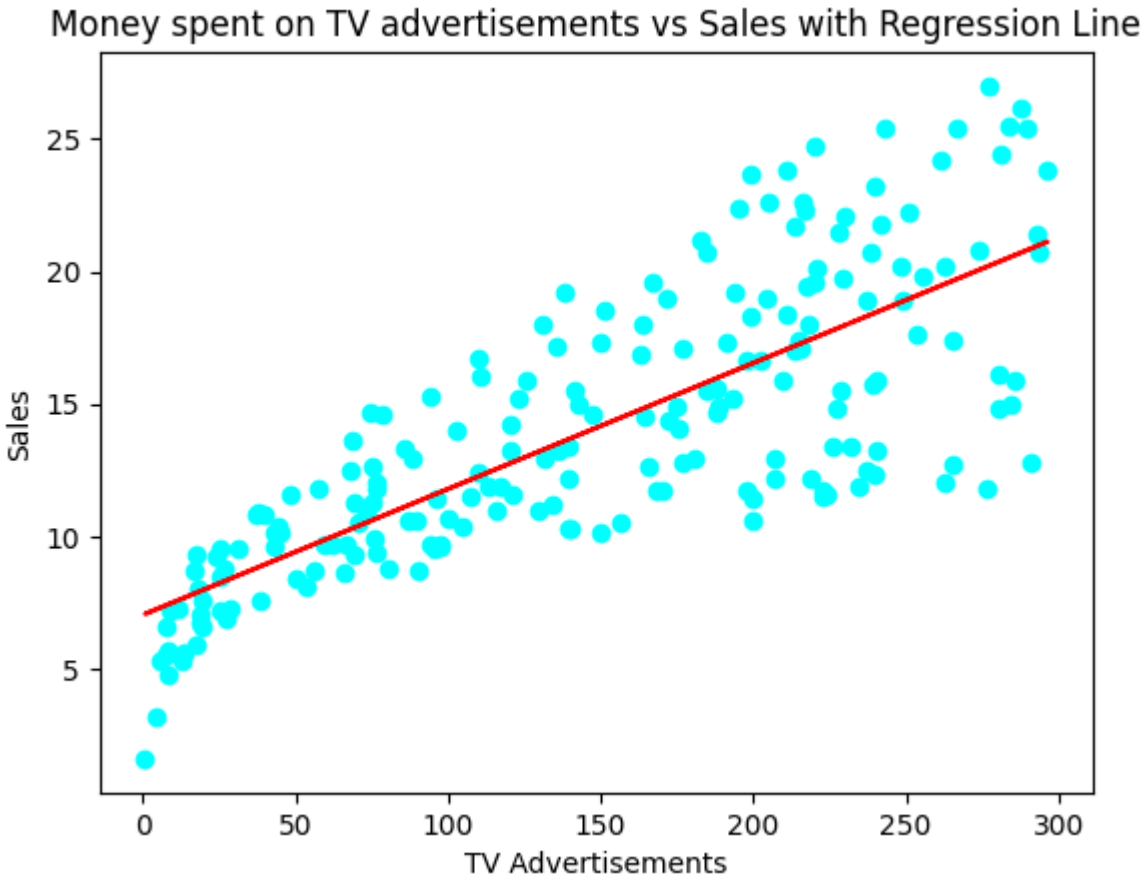
```
Predicted Sales: [11.78625759 16.53992164 21.29358568]
C:\Users\DELL\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

The code above predicts the sales based on the new TV advertisements spending values. The predicted sales for the new TV advertisements spending values of 100, 200, and 300 are printed.

1.4 Regression Line superimposed on the data

In [ ]:
```
plt.scatter(df1['TV'], df1['Sales'],color='cyan')
plt.plot(df1['TV'], model.predict(df1[['TV']]), color='red')
plt.xlabel('TV Advertisements')

plt.ylabel('Sales')
plt.title('Money spent on TV advertisements vs Sales with Regression Line')
plt.show()
```



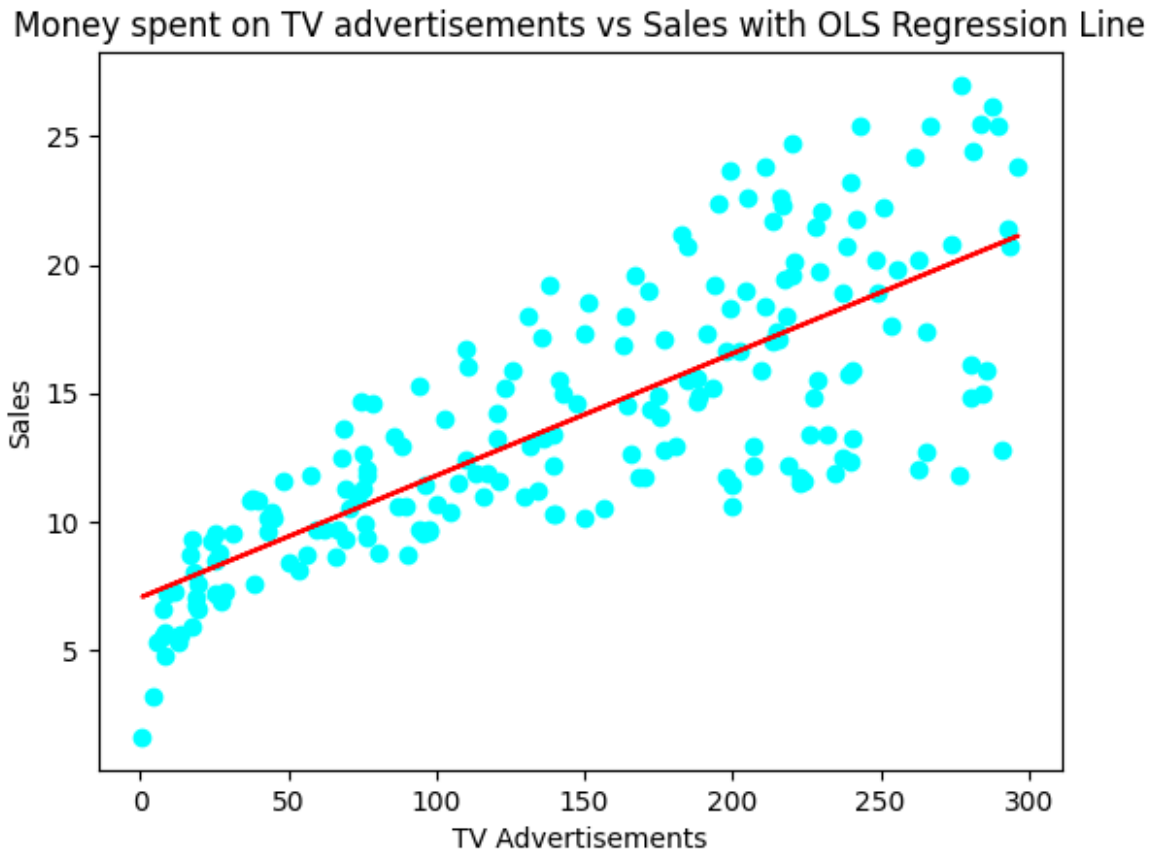
1.5 OLS regressor and plot line of regression and residuals

In [ ]:
```
import statsmodels.api as sm
X = sm.add_constant(X)
model_ols = sm.OLS(y, X).fit()
print(model_ols.summary())
```

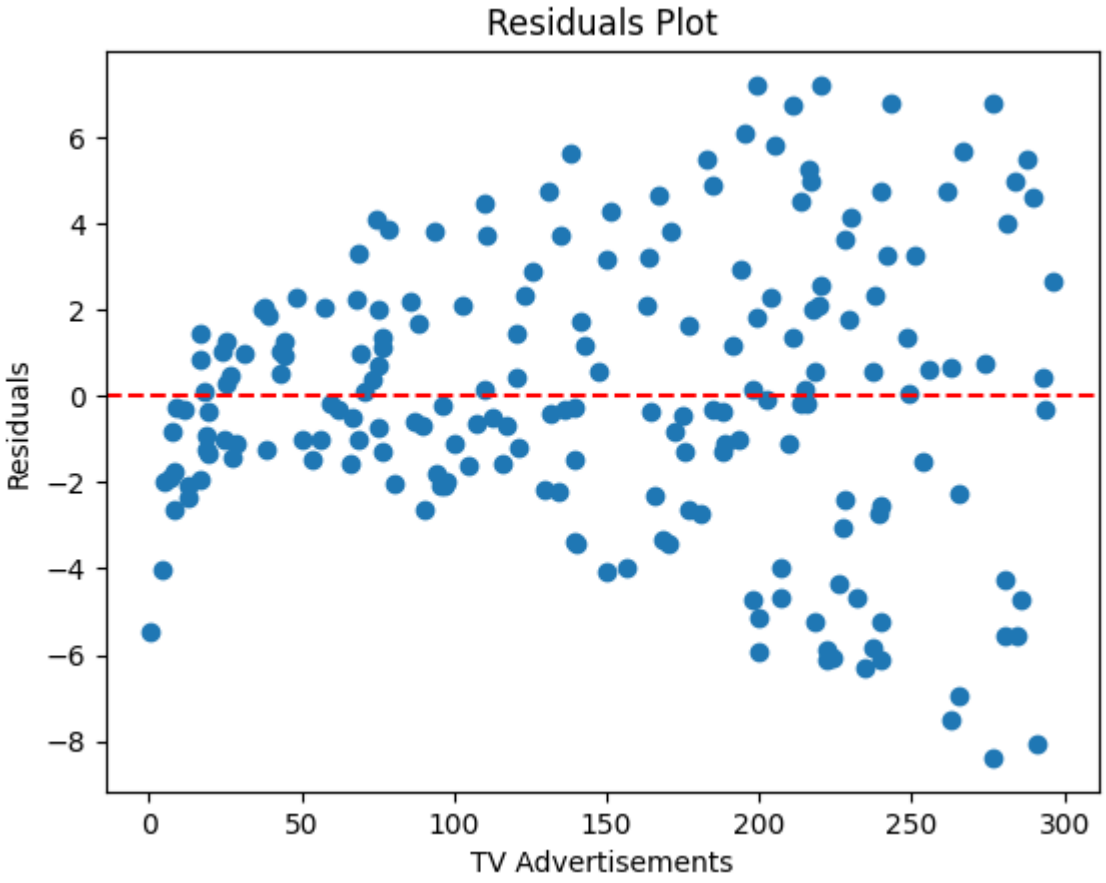
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Fri, 01 Dec 2023   Prob (F-statistic):           1.47e-42
Time:                        15:58:33   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

In [ ]:
```
plt.scatter(df1['TV'], df1['Sales'],color='cyan')
plt.plot(df1['TV'], model_ols.predict(X), color='red')
plt.xlabel('TV Advertisements')
plt.ylabel('Sales')
plt.title('Money spent on TV advertisements vs Sales with OLS Regression Line')
plt.show()
```



In [ ]:
```
plt.scatter(df1['TV'], model_ols.resid)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('TV Advertisements')
plt.ylabel('Residuals')
plt.title('Residuals Plot')
plt.show()
```



To comment on the heteroscedasticity from the above code, we need to analyze the residuals plot.

above we can see a scatter plot of TV advertisements versus residuals is plotted. If the spread of residuals is consistent across the range of TV advertisements, it indicates homoscedasticity. On the other hand, if the spread of residuals varies across the range of TV advertisements, it indicates heteroscedasticity.

From the residuals plot, it appears that the spread of residuals is not consistent. The residuals is seen like a cone in the above plot , indicating heteroscedasticity. This means that thechange in error is not constant or like consisnt across different levels of TV advertisements.

QUESTION 2

In [ ]:
```
df2=pd.read_csv('data2.csv')
df2
```

Out[ ]:

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0
...	...	...	...	...	...	...	...	...	...
762	10	101	76	48	180	32.9	0.171	63	0
763	2	122	70	27	0	36.8	0.340	27	0
764	5	121	72	23	112	26.2	0.245	30	0
765	1	126	60	0	0	30.1	0.349	47	1
766	1	93	70	31	0	30.4	0.315	23	0

767 rows × 9 columns

2.1 Split the data into training and testing sets

In [ ]:
```
X = df2.iloc[:, :-1]
y = df2.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
print('X_train\n', X_train.head(),end='\n')
print(end='\n')
print('X_test\n', X_test.head())
print(end='\n')

print('y_train\n', y_train.head())
print(end='\n')

print('y_test\n', y_test.head())
```

```
X_train
      6  148  72  35    0  33.6  0.627  50
60    8  133  72   0    0  32.9  0.270  39
554   7  124  70  33  215  25.5  0.161  37
346   3  116   0   0    0  23.5  0.187  23
294   6  151  62  31  120  35.5  0.692  28
231   1   79  80  25   37  25.4  0.583  22

X_test
      6  148  72  35    0  33.6  0.627  50
667   6   98  58  33  190  34.0  0.430  43
324   1  157  72  21  168  25.6  0.123  24
623   2  108  64   0    0  30.8  0.158  21
689   8  107  80   0    0  24.6  0.856  34
521   6  114   0   0    0   0.0  0.189  26

y_train
 60     1
554    0
346    0
294    0
231    0
Name: 1, dtype: int64

y_test
 667    0
324    0
623    0
689    0
521    0
Name: 1, dtype: int64
```

2.2 Rescale the distribution of values

for mean to be zero and standard dev=1 we have to use standard scaler and not tnormal scaler for us to acheive the desired result

In [ ]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


print('X_train_scaled\n', X_train_scaled[1:5])
print(end='\n')
print('X_test_scaled\n', X_test_scaled[1:5])
```

```
X_train_scaled
 [[ 0.9261063   0.12849508  0.01685836  0.78575146  1.16685692 -0.80599749
  -0.9358749   0.313639  ]
 [-0.24804708 -0.13018303 -3.77053549 -1.29229355 -0.70192629 -1.05319639
  -0.85744861 -0.87682976]
 [ 0.63256795  1.00153373 -0.41598665  0.65980934  0.3411155   0.42999706
   0.66583121 -0.45166235]
 [-0.83512378 -1.32656932  0.55791462  0.28198297 -0.38032174 -0.81835743
   0.33704408 -0.96186324]]

X_test_scaled
 [[-0.83512378  1.19554232  0.12506961  0.03009873  0.75833222 -0.79363754
  -1.05049793 -0.79179628]
 [-0.54158543 -0.38886115 -0.3077754  -1.29229355 -0.70192629 -0.15092038
  -0.94492408 -1.04689673]
 [ 1.21964464 -0.42119591  0.55791462 -1.29229355 -0.70192629 -0.91723699
   1.1605201   0.05853855]
 [ 0.63256795 -0.19485256 -3.77053549 -1.29229355 -0.70192629 -3.95778357
  -0.85141582 -0.62172931]]
```

2.3 Develop a KNN classifier model

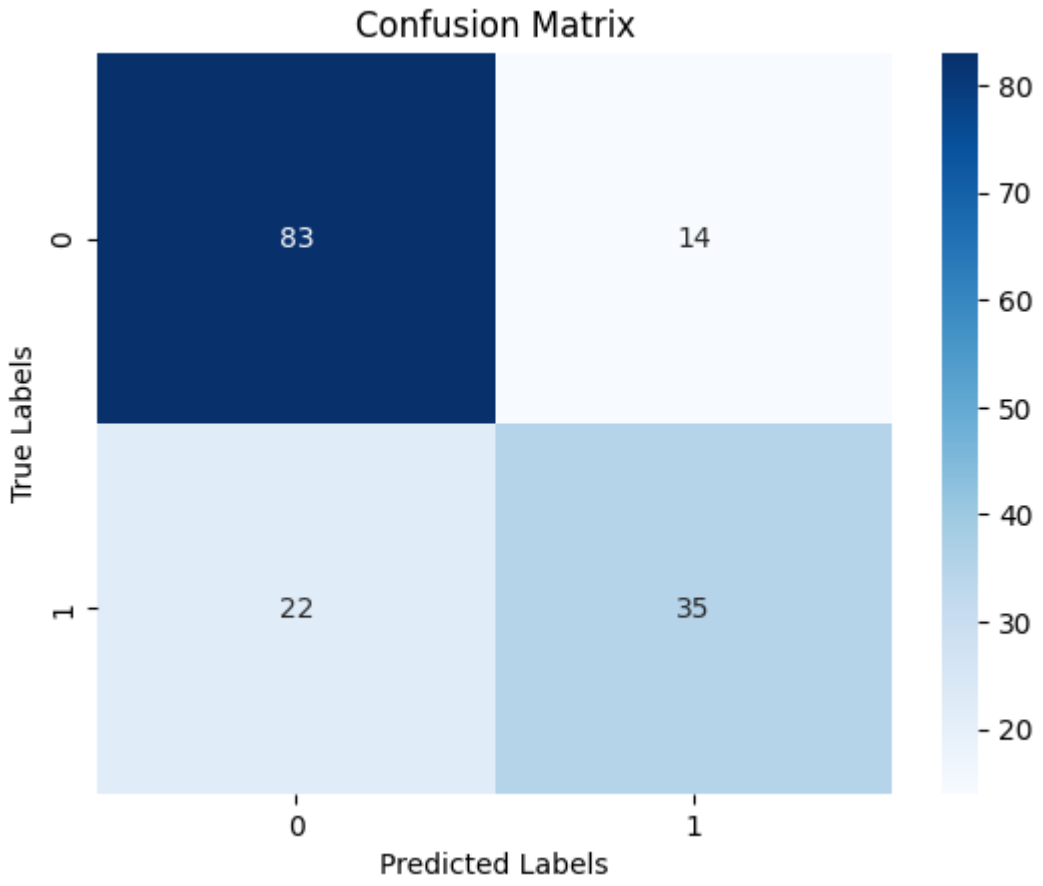
In [ ]:
```
k_values = range(1, 21)
mean_errors = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled)
    mean_errors.append((y_pred != y_test).mean())

optimal_k = k_values[mean_errors.index(min(mean_errors))]
```

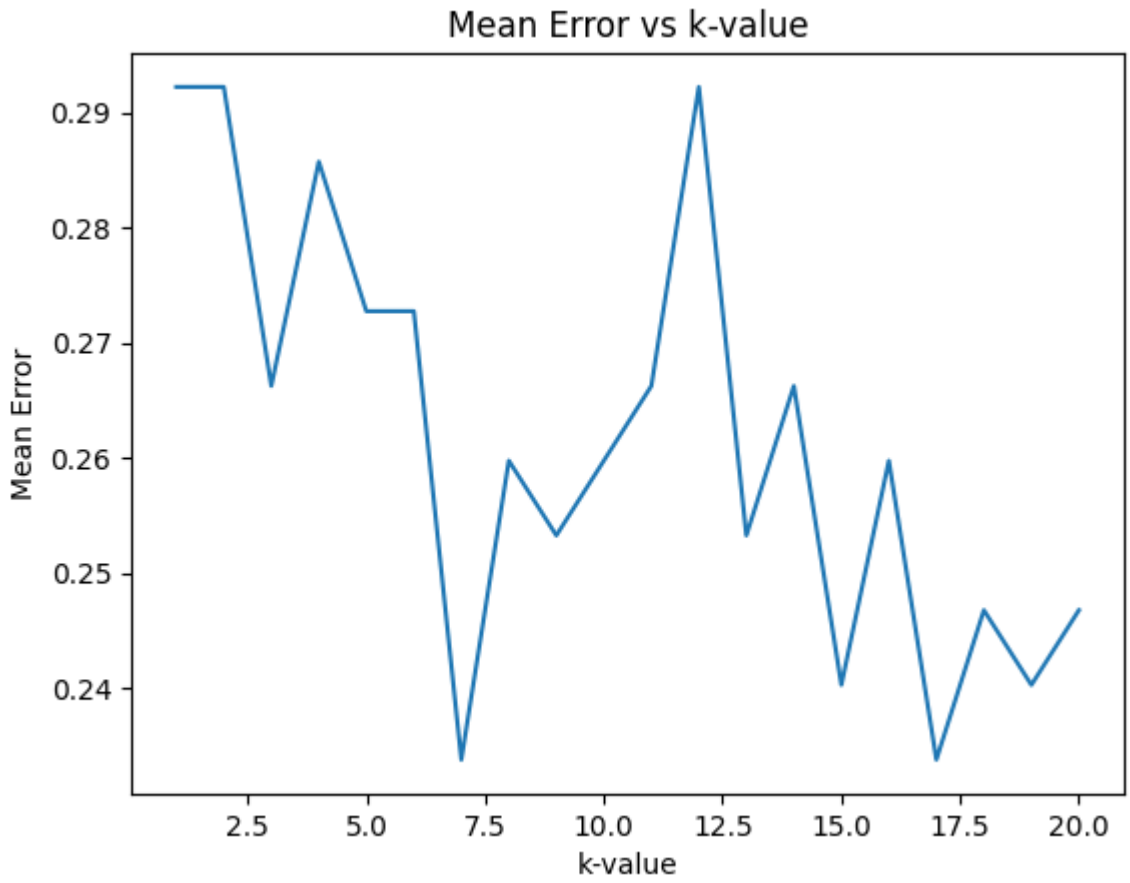
2.4 CONFUSION MATRIX

In [ ]:
```
sns.heatmap(confusion_mat, annot=True, cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



2.5 MEAN ERROR VS K-VALUE

In [ ]:
```
plt.plot(k_values, mean_errors)
plt.xlabel('k-value')
plt.ylabel('Mean Error')
plt.title('Mean Error vs k-value')
plt.show()
```



In [ ]:
```
knn_optimal = KNeighborsClassifier(n_neighbors=optimal_k)
knn_optimal.fit(X_train_scaled, y_train)
y_pred_optimal = knn_optimal.predict(X_test_scaled)
y_pred_optimal
```

Out[ ]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0],
      dtype=int64)
```

We can see that:

1. As the k value increases, the mean error generally decreases.
2. The lowest point on the graph represents the optimal k value with the minimum mean error.this help us in choosing the optimal k value , as it is is important to balance between overfitting (low k value) and underfitting (high k value) the data.
3. It is recommended to select the k value that corresponds to the lowest mean error for the most accurate predictions.

Based on the k value vs mean error graph, we can determine the optimal k value for our KNN classifier model which is 7.5