sso-regression-on-advertising-data

February 3, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv('Advertising.csv')
```

[3]:
```
df.head()
```

[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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

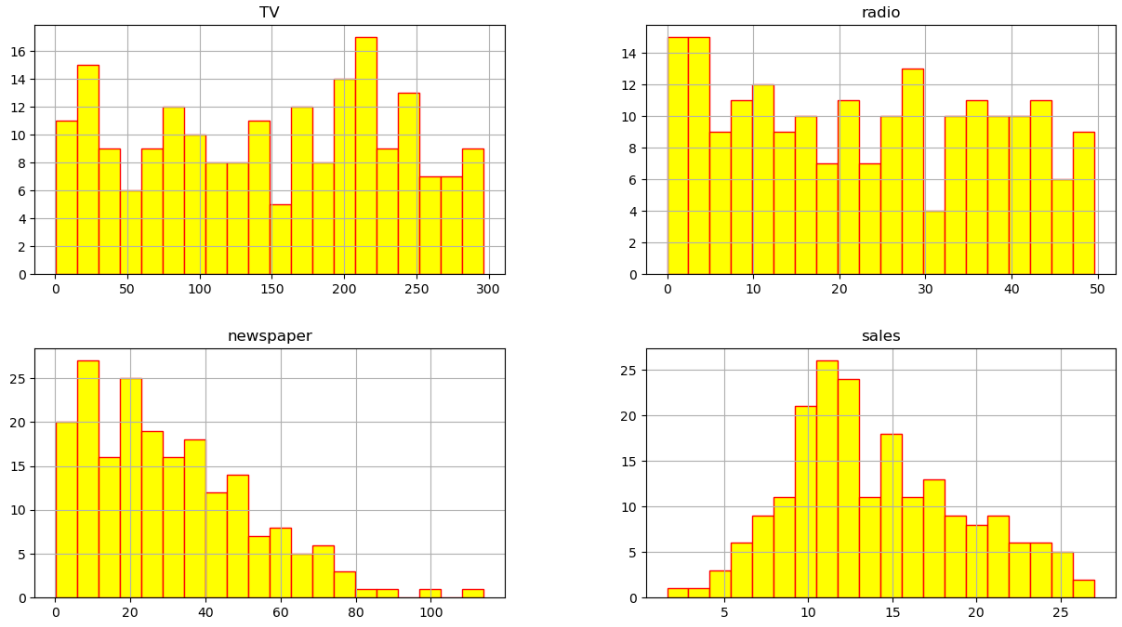
[4]:
```
df.info()
```

```
<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
```

[5]:
```
df.isnull().sum()
```

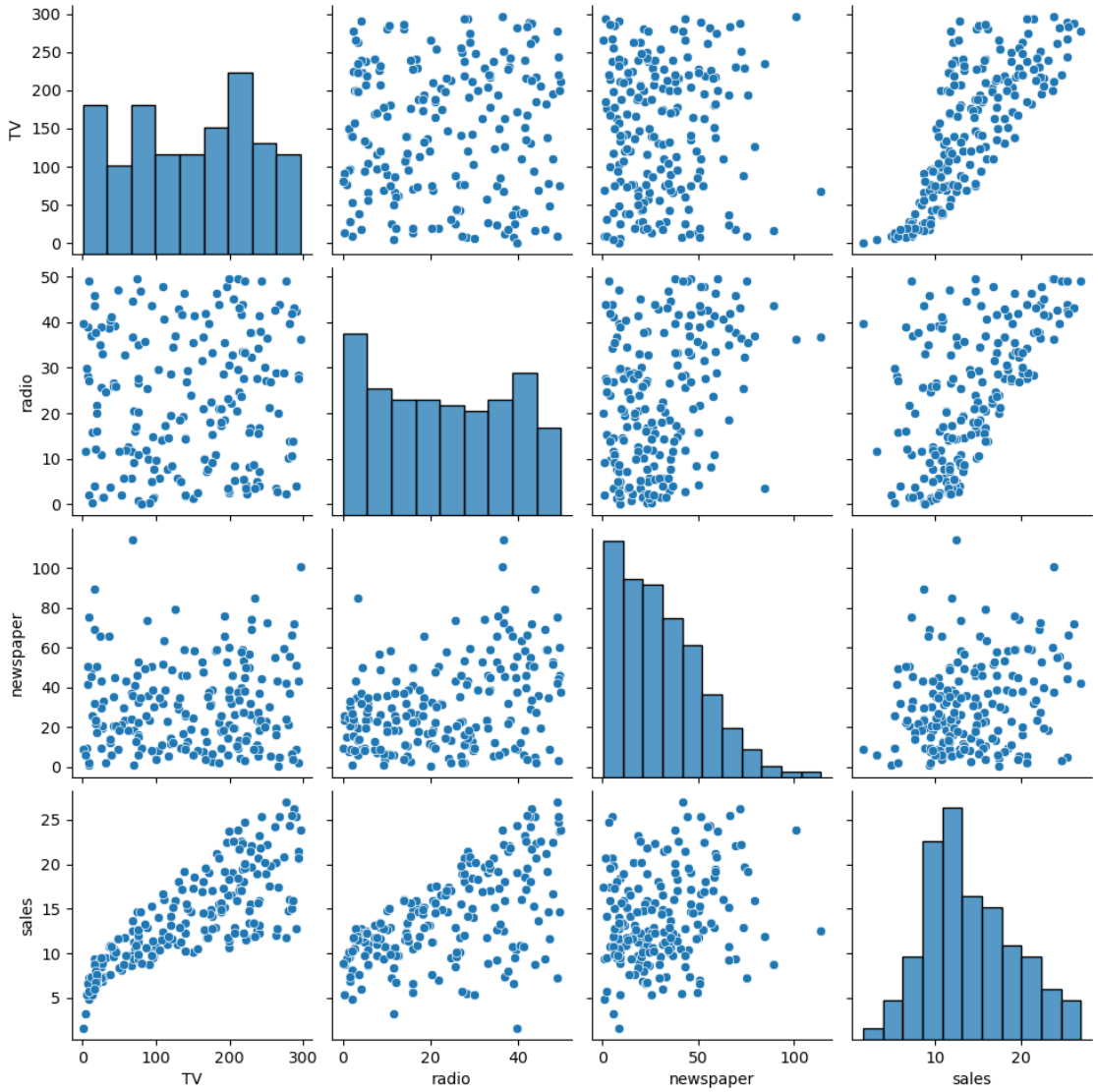
[5]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

[6]:
```
df.hist(bins=20,figsize=(15,8), color='yellow', edgecolor= 'red')
plt.show()
```

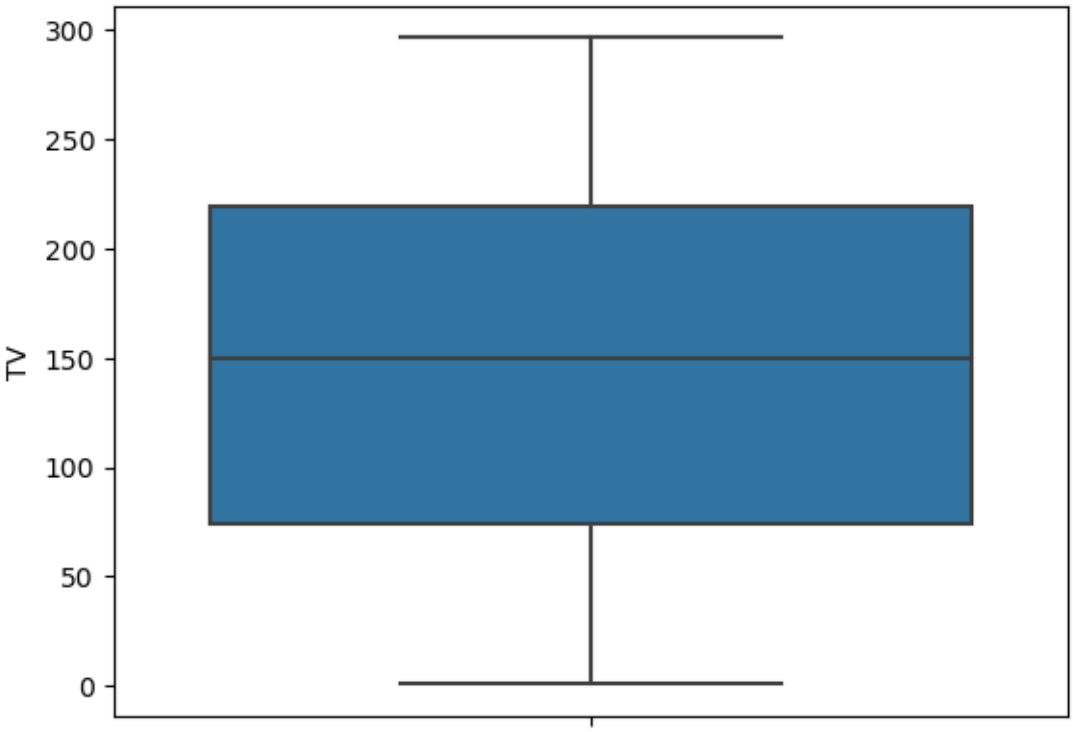


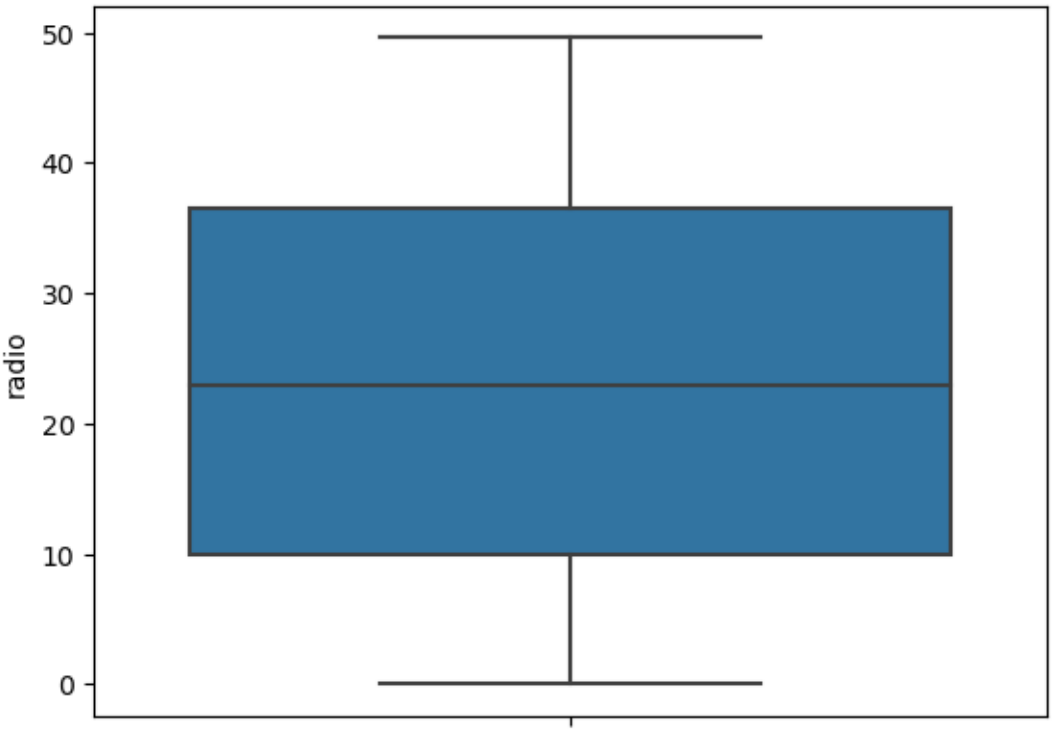
[7]:
```
sns.pairplot(df)
```

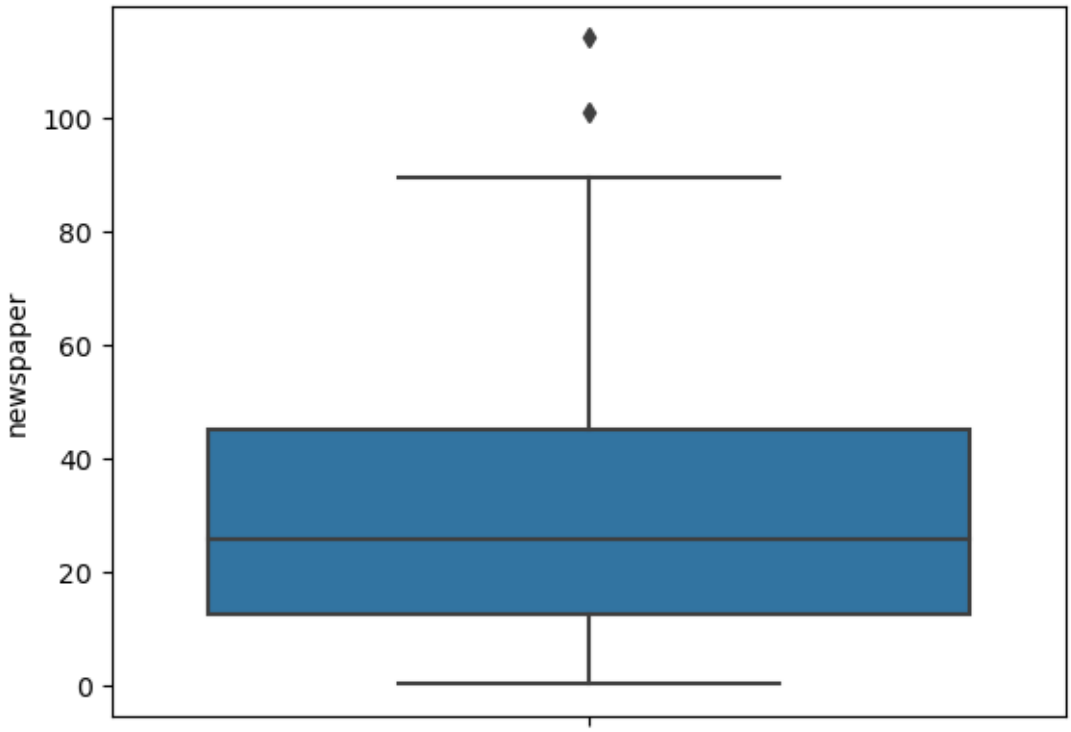
[7]: <seaborn.axisgrid.PairGrid at 0x13ab8526c10>

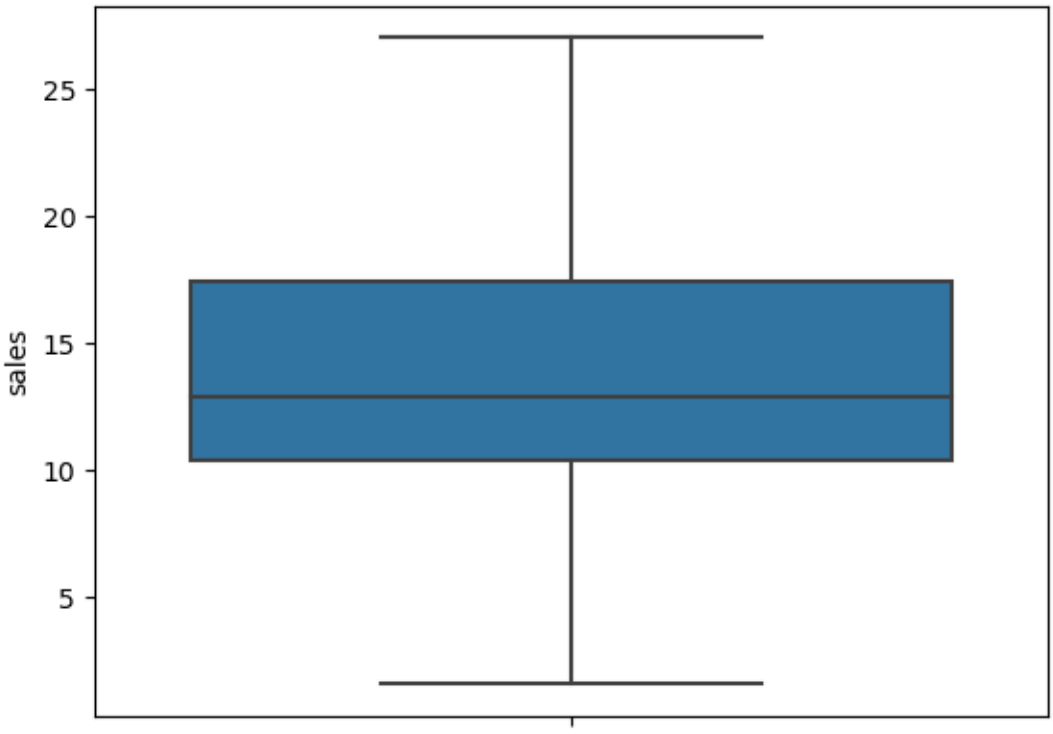


[8]:
```
num = df
for z in num.columns:
    sns.boxplot(data=df, y=z)
    plt.show()
```


300
250
200
150
100
50
0
TV


50
40
30
20
10
0
radio

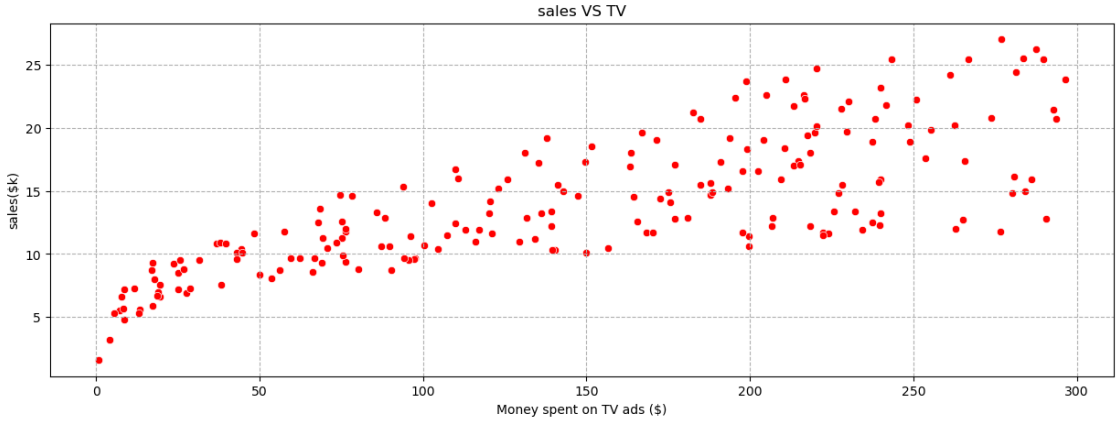

newspaper
100
80
60
40
20
0

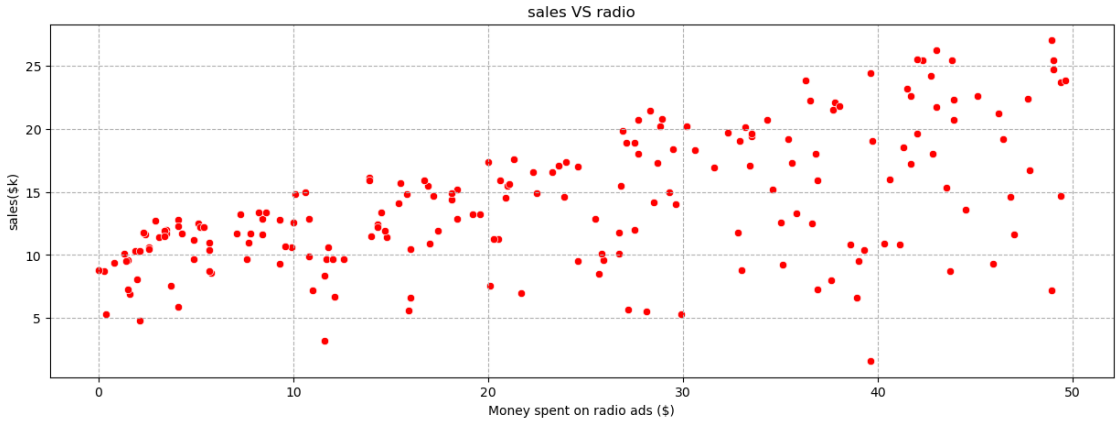

sales
25
20
15
10
5

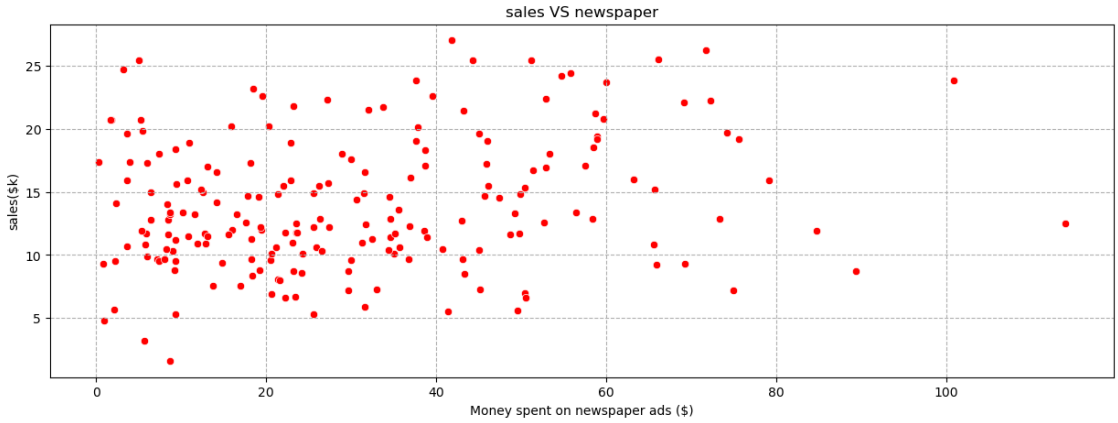
[9]:
```
df.columns
```

[9]:
```
Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

[10]:
```
target_clm = 'sales'
sub_columns = [z for z in df.columns if z != target_clm  ]
for sub_column1 in sub_columns:
    plt.figure(figsize=(15,5))
    sns.scatterplot(data=df, x=sub_column1, y=target_clm, color='red')
    plt.title(f'{target_clm} VS {sub_column1}')
    plt.xlabel(f'Money spent on {sub_column1} ads ($)')
    plt.ylabel(f'{target_clm}($k)')
    plt.grid(linestyle='--')
    plt.show()
```







[11]:
```
df.columns
```

[11]:
```
Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

[12]:
```
from sklearn.model_selection import train_test_split
X = df.drop(['sales'], axis=1)
y = df['sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

[13]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[14]:
```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
```

[15]:
```
alpha = 1.0
ridge_model = Ridge(alpha=alpha)
ridge_model.fit(X_train_scaled,y_train)
```

[15]:
```
Ridge()
```

[16]:
```
y_pred = ridge_model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 3.1941558922079616
```

[17]:
```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error
```

[18]:
```
alpha = 0.1
lasso_model = Lasso(alpha=alpha)
lasso_model.fit(X_train_scaled, y_train)
```

[18]:
```
Lasso(alpha=0.1)
```

[19]:
```
y_pred = lasso_model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 3.208876825052843
```