Lab_Task06

October 16, 2024

[9]:
```
import pandas as pd
df=pd.read_csv( C:/5th_semester/machine_learning/ml/data.csv )
```

[3]:
```
df
```

[3]:

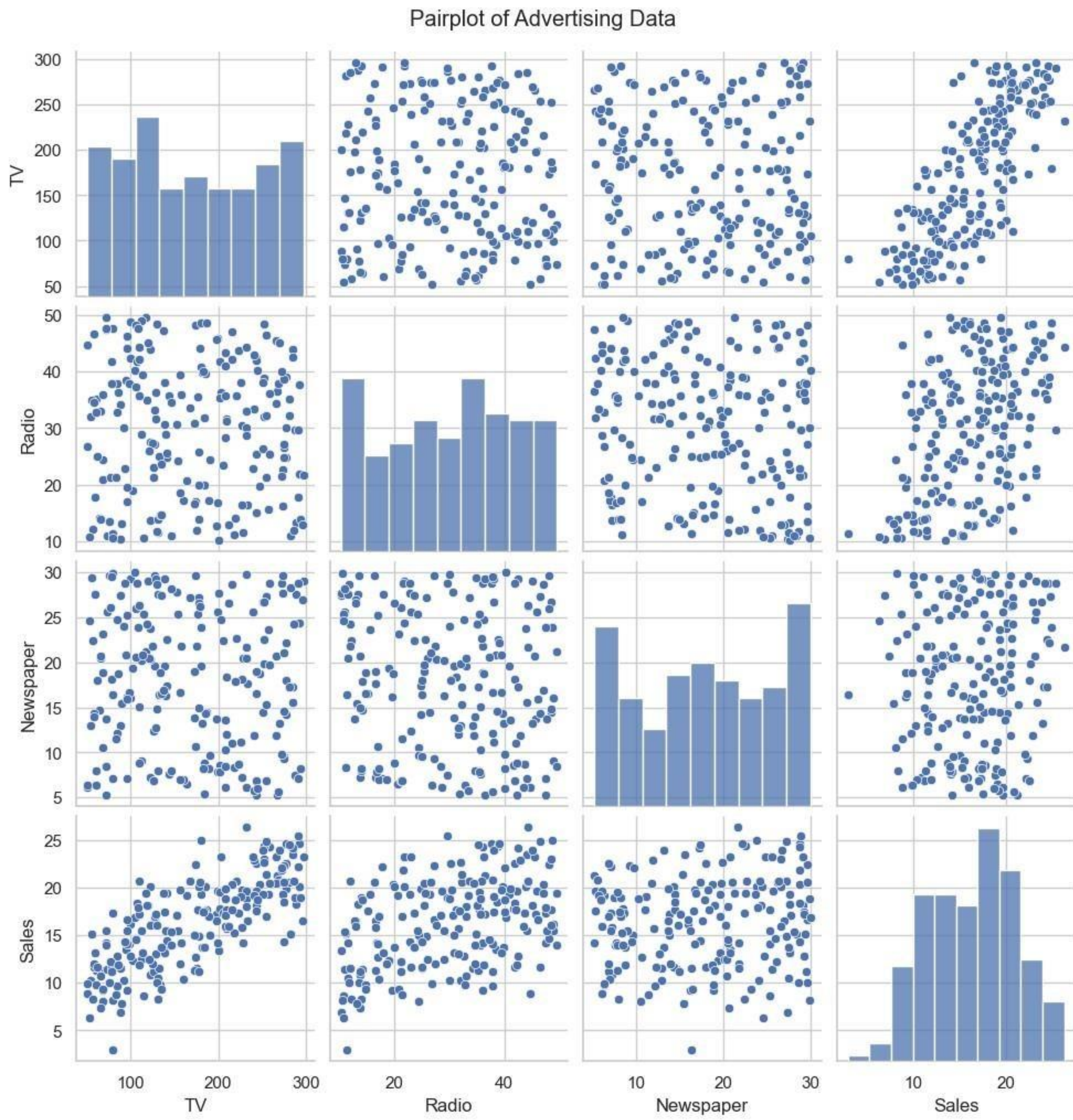
	TV	Radio	Newspaper	Sales
0	143.635030	35.681266	7.578097	14.015299
1	287.678577	13.365599	27.563823	18.227685
2	232.998485	16.465149	17.631309	16.492024
3	199.664621	45.942168	25.661437	19.667324
4	89.004660	34.257162	13.001240	11.494491
..	…	…	…	…
195	137.302394	47.230293	16.849041	19.388495
196	231.488920	44.336510	21.688943	26.308555
197	274.277565	27.159761	9.307997	22.329760
198	271.771606	40.034843	9.807225	22.038494
199	244.968886	40.181715	6.021715	18.673907

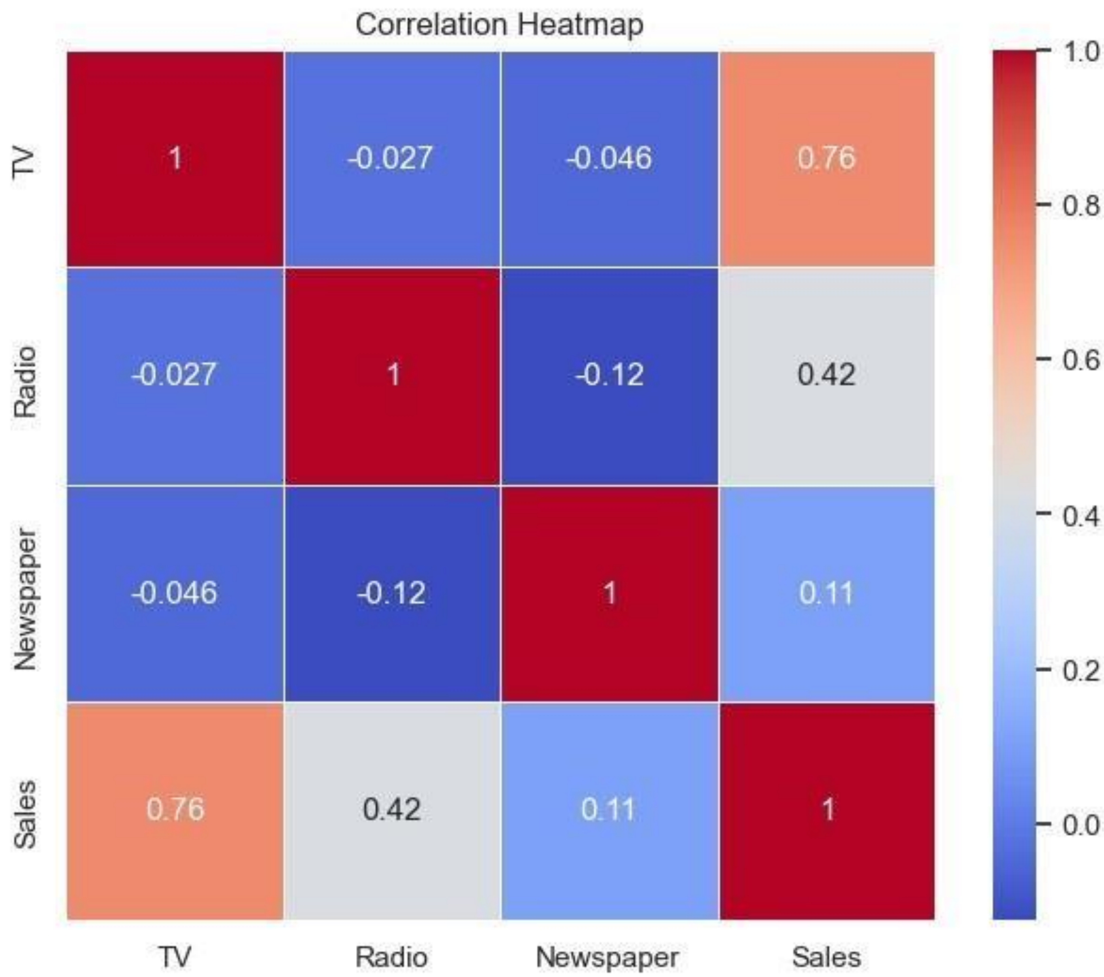
[200 rows x 4 columns]

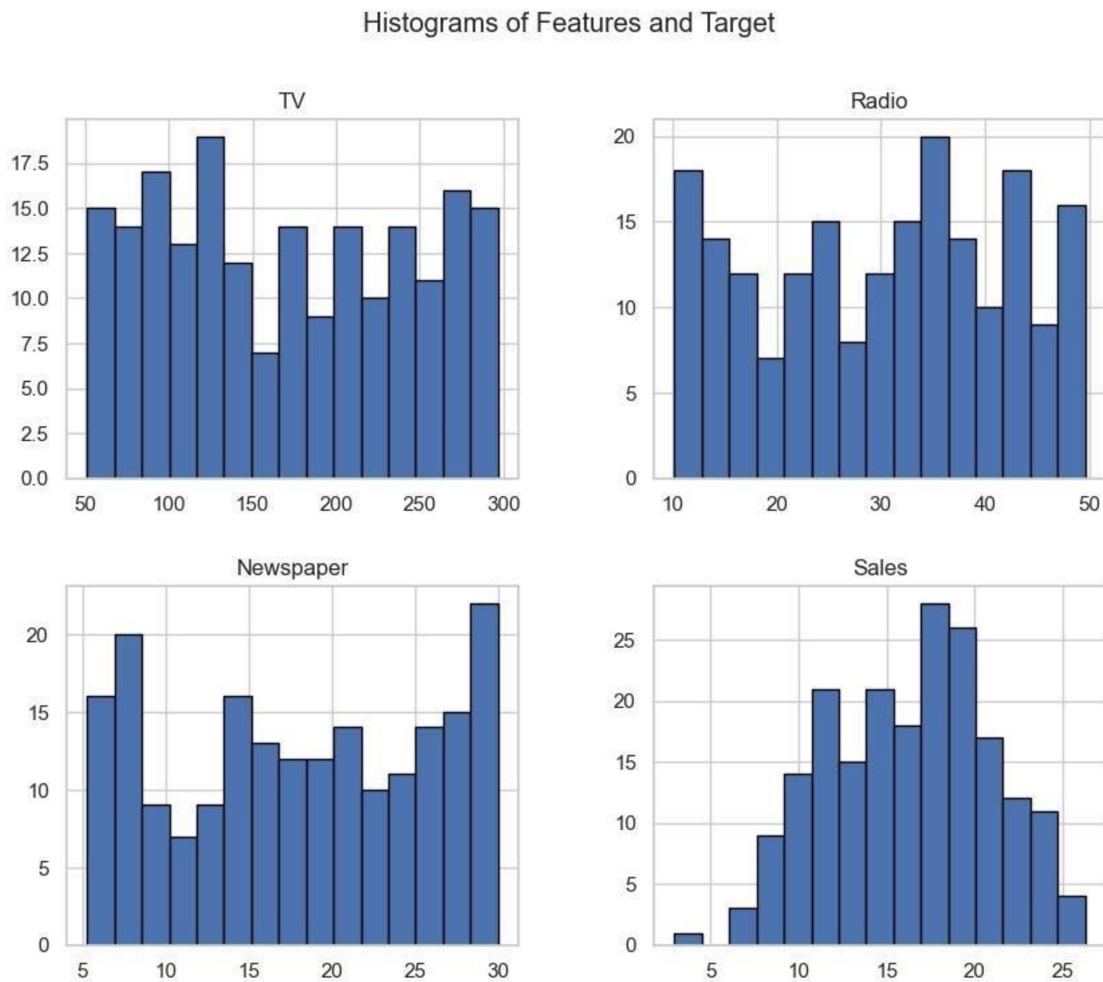
[4]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the aesthetics for the plots
sns.set(style="whitegrid")

# 1. Pairplot - To visualize pairwise relationships between features and target
 variable
plt.figure(figsize=(10   6))
sns.pairplot(data)
plt.suptitle("Pairplot of Advertising Data     =1.02)
plt.show()

# 2. Correlation Heatmap - To visualize the correlations between features and
 target
plt.figure(figsize=(8   6))
corr_matrix   data.corr()
```

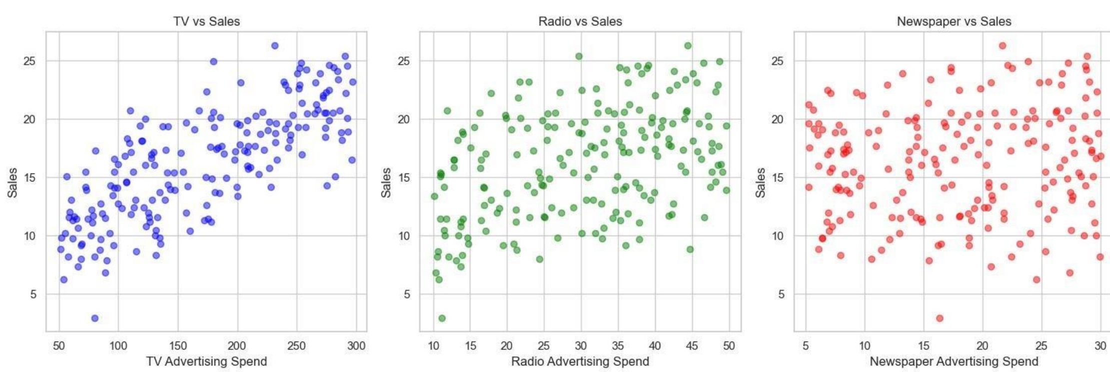
```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5,␣
↪square=True)
plt.title("Correlation Heatmap")
plt.show()

# 3. Histograms - To visualize the distribution of individual features
data.hist(bins=15, figsize=(10, 8), edgecolor='black')
plt.suptitle("Histograms of Features and Target")
plt.show()

# 4. Scatter Plots - To visualize relationships between features and the target␣
↪(Sales)
plt.figure(figsize=(15, 5))

# TV vs Sales
plt.subplot(1, 3, 1)
plt.scatter(data['TV'], data['Sales'], color='blue', alpha=0.5)
plt.title("TV vs Sales")
plt.xlabel("TV Advertising Spend")
plt.ylabel("Sales")

# Radio vs Sales
plt.subplot(1, 3, 2)
plt.scatter(data['Radio'], data['Sales'], color='green', alpha=0.5)
plt.title("Radio vs Sales")
plt.xlabel("Radio Advertising Spend")
plt.ylabel("Sales")

# Newspaper vs Sales
plt.subplot(1, 3, 3)
plt.scatter(data['Newspaper'], data['Sales'], color='red', alpha=0.5)
plt.title("Newspaper vs Sales")
plt.xlabel("Newspaper Advertising Spend")
plt.ylabel("Sales")

plt.tight_layout()
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```


Pairplot of Advertising Data
TV
Radio
Newspaper
Sales
TV
Radio
Newspaper
Sales


Correlation Heatmap
TV
Radio
Newspaper
Sales
1
-0.027
-0.046
0.76
-0.027
1
-0.12
0.42
-0.046
-0.12
1
0.11
0.76
0.42
0.11
1
1.0
0.8
0.6
0.4
0.2
0.0





[5]:
```
# Step 2: Implement Linear Regression from Scratch

class LinearRegressionScratch
```

```python
    def __init__(self):
        self.beta   None

    def fit(self, X, y):
        # Add intercept (bias term) to X
        X_b   np.c_[np.ones((X.shape[0], 1))    ]  # Add a column of ones to X
        # Compute the weights using the normal equation
        self.beta   np.linalg.inv(X_b.T.dot(X_b)) dot(X_b.T).dot(y)

    def predict(self, X):
        # Add intercept (bias term) to X
        X_b   np.c_[np.ones((X.shape[0], 1))    ]
        # Make predictions
        return X_b.dot(self.beta)

# Splitting the data into training and testing sets
X   data[['TV', 'Radio'  'Newspaper' ]].values
y   data['Sales'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2 □
 □random_state=42)

# Initialize and fit the model
model_scratch   LinearRegressionScratch()
model_scratch.fit(X_train, y_train)

# Predict on test data
y_pred_scratch   model_scratch.predict(X_test)

# Evaluate the model using Mean Squared Error (MSE)
mse_scratch   mean_squared_error(y_test, y_pred_scratch)
print(f"MSE from Scratch: {mse_scratch}")
```

```
MSE from Scratch: 2.7616556127332315
```

[6]:

```python
from sklearn.linear_model import LinearRegression

# Initialize the LinearRegression model from scikit-learn
model_sklearn   LinearRegression()

# Fit the model on the training data
model_sklearn.fit(X_train, y_train)

# Predict on test data
y_pred_sklearn   model_sklearn.predict(X_test)

# Evaluate the model using Mean Squared Error (MSE)
```

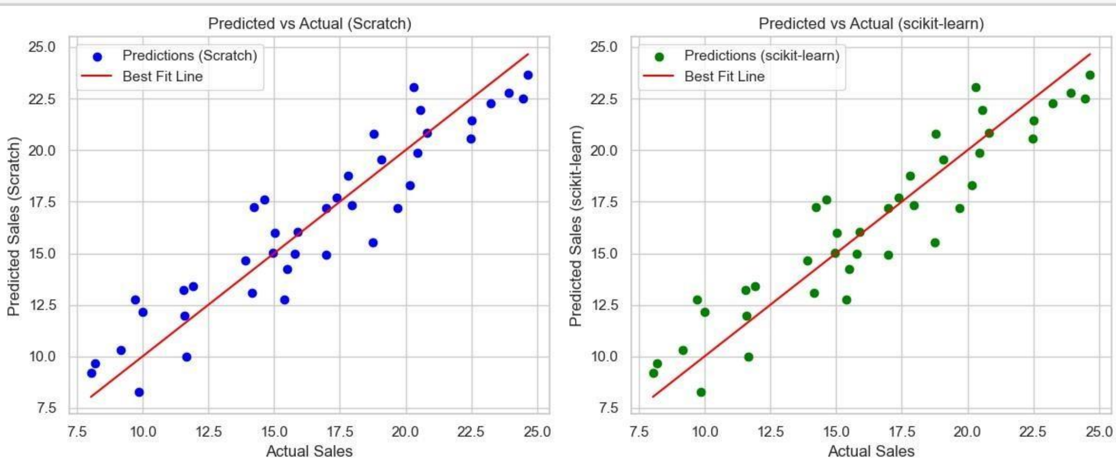
```python
mse_sklearn = mean_squared_error(y_test, y_pred_sklearn)
print(f"MSE with scikit-learn:
{mse_sklearn}")
```

```
MSE with scikit-learn: 2.761655612733244
```

[7]:
```python
# Step 2: Calculate R-squared (accuracy) using
scikit-learn

r2_sklearn = model_sklearn.score(X_test, y_test)
print(f"R-squared with scikit-learn: {r2_sklearn}")
```

```
R-squared with scikit-learn: 0.8716759282807236
```

[8]:
```
# Import required libraries
import matplotlib.pyplot as plt
import numpy as np

# Step 1: Plot predictions vs actual values
plt.figure(figsize=(12  5))

# Predictions from scratch model
plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred_scratch, color='blue'  label='Predictions (Scratch)')
# Plot best-fitted line
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red  label='Best Fit Line  )
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales (Scratch)")
plt.title("Predicted vs Actual (Scratch)")
plt.legend()

# Predictions from scikit-learn model
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_sklearn, color='green'  label='Predictions (scikit-learn)')
# Plot best-fitted line
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red  label='Best Fit Line  )
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales (scikit-learn)")
plt.title("Predicted vs Actual (scikit-learn)")
plt.legend()

plt.tight_layout()
plt.show()
```



[ ]:

```

```