```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns


# Importing Data from the CSV file
data = pd.read_csv('Sales.csv')
print(data.head())
```

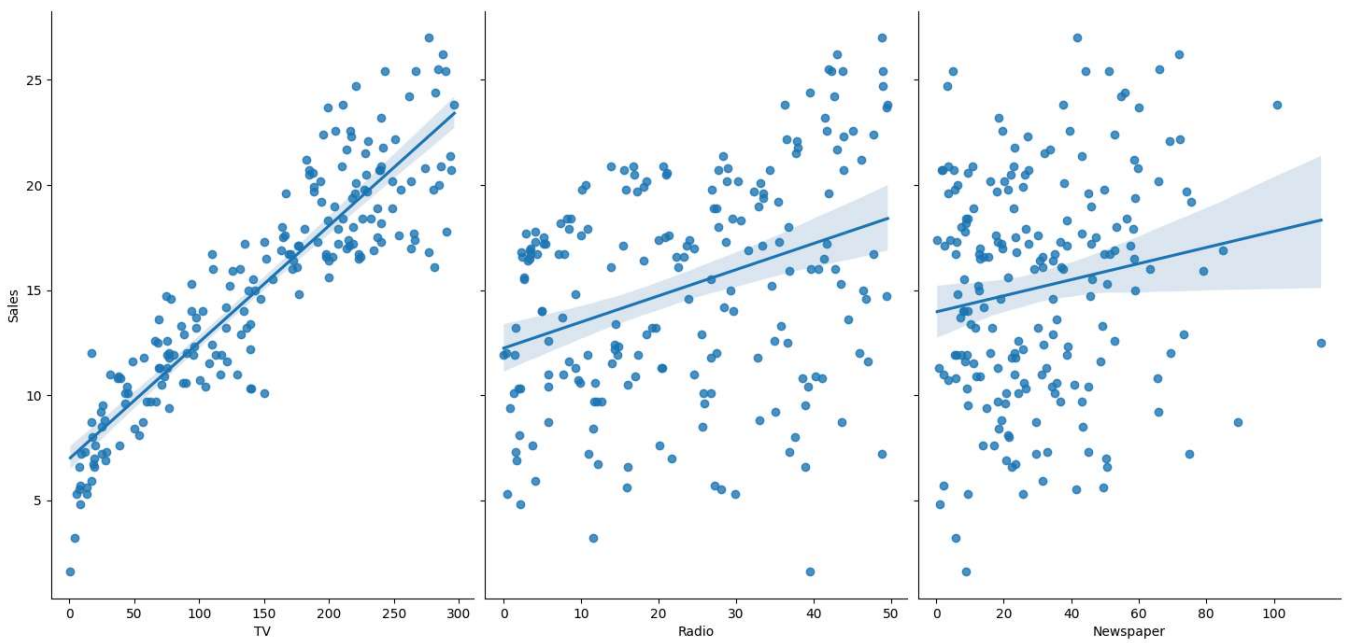
```
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
```

```
# Checking for null values that could require cleaning
print(data.isnull().sum())

# Getting a summary of data
print(data.describe())
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
                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
```

```
# Plotting the data
sns.pairplot(data, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=7, aspect=0.7, kind='reg')
plt.show()
```



```
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Using the Linear Regression Model

```
# Create a linear regression model
model = LinearRegression()

# Train the model using the training data
model.fit(X_train, y_train)

# Print the coefficients
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Coefficients: [0.05450927 0.10094536 0.00433665]
Intercept: 4.714126402214127
```

```
# Predict the sales using the testing data
y_pred = model.predict(X_test)

# Display the first few predictions
print(y_pred[:5])
```

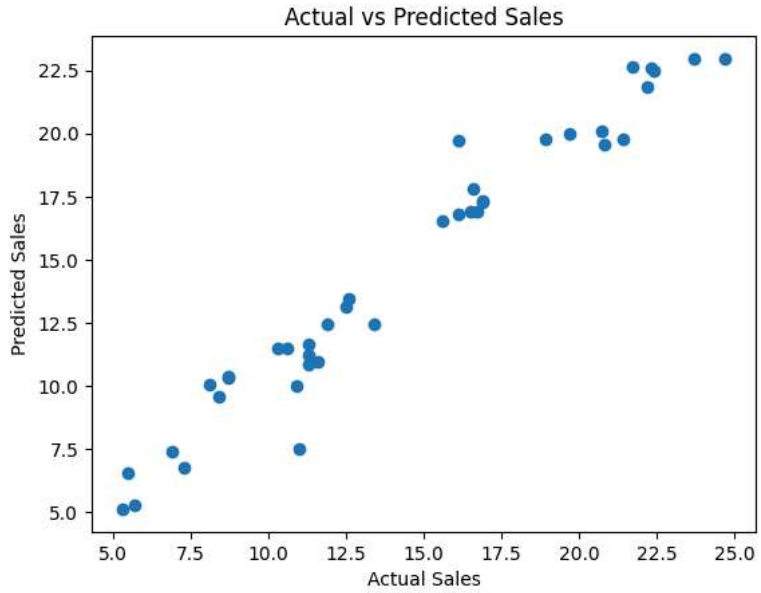
```
[17.0347724  20.40974033 23.72398873  9.27278518 21.68271879]
```

```
# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate the coefficient of determination (R^2)
r2 = r2_score(y_test, y_pred)
print("R^2 Score:", r2)

# Plot the predicted vs actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.title("Actual vs Predicted Sales")
plt.show()
```

```
Mean Squared Error: 1.4374328500000009
R^2 Score: 0.9534827934927883
```



Using the Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
```

```
# Create a Random Forest Regressor model
model = RandomForestRegressor(n_estimators=100, random_state=24)

# Train the model using the training data
model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=24)
```

```
# Predict the sales using the testing data
y_pred = model.predict(X_test)

# Display the first few predictions
print(y_pred[:5])
```

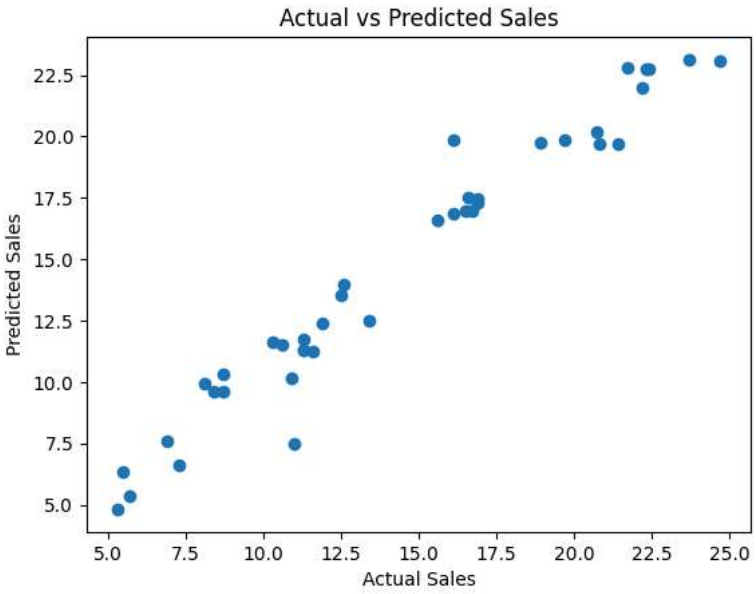
```
[17.479 22.761 19.698  6.614 23.098]
```

```
# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate the coefficient of determination (R^2)
r2 = r2_score(y_test, y_pred)
print("R^2 Score:", r2)

# Plot the predicted vs actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.title("Actual vs Predicted Sales")
plt.show()
```

```
Mean Squared Error: 1.4358602000000027
R^2 Score: 0.9535336865030694
```



Start coding or generate with AI.