SALES PREDICTION USING PYTHON

Objective

The goal is to predict product sales based on advertising expenditure across three platforms: TV, Radio, and Newspaper. By understanding the relationship between advertising efforts and sales, businesses can make informed decisions on how to allocate their advertising budgets effectively.

Dataset Description

The dataset contains the following columns:

- **TV**: Advertising expenditure on TV campaigns (in thousands of dollars).
- **Radio**: Advertising expenditure on Radio campaigns (in thousands of dollars).
- **Newspaper**: Advertising expenditure on Newspaper campaigns
- **Sales**: Total product sales (in thousands of units).

Key Steps

- **Import Libraries**: libraries for data handling, visualization, and machine learning.
- **Load Data**: Load the dataset, check for missing values, and summarize statistics.
- **EDA**: Visualize relationships (scatter plots, heatmaps) and analyze correlations.
- **Feature and Target Setup**: Define independent variables (X) and the target (y).
- **Split Data**: Divide the dataset into training and testing sets using train_test_split().
- **Train Model**: Fit a Linear Regression model to the training data.
- **Model Parameters**: Output the model's intercept and coefficients.
- **Make Predictions**: Predict sales on the test dataset.
- **Evaluate Model**: Calculate R-squared and RMSE to assess model performance.
- **Visualize Results**: Compare actual vs predicted sales using a scatter plot.
- **Conclude**: Summarize findings and recommend strategies based on insights.

Packages

- **pandas**: Used to load and manipulate data (e.g., reading CSV files, inspecting data).
- **numpy**: Used for numerical operations (e.g., calculating RMSE).
- **matplotlib.pyplot**: Used for plotting graphs (e.g., scatter plots, line plots).
- **seaborn**: Built on matplotlib, used for creating more advanced and aesthetically pleasing visualizations (e.g., pairplots, heatmaps).
- **sklearn.model_selection.train_test_split**:Splits the dataset into training and testing set
- **sklearn.linear_model.LinearRegression**: Used to create and train the linear regression model.
- **sklearn.metrics.mean_squared_error & r2_score**: Used to evaluate the model's performance (e.g., RMSE, R^2 score).

Step 1: Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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, r2_score
```

Step 2: Loading the Dataset

```
data = pd.read_csv('sales_pred.csv')
# Data Preprocessing and Exploration
print(data.head())  # Display the first 5 rows of the dataset
print(data.isnull().sum())  # Check for missing values in each column
print(data.describe())  # Get a summary of the numerical columns (mean, min, max,
etc.)
```

Output:

First 5 rows of the dataset:

```
        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
```

Missing values in each column:

```
TV            0
Radio         0
Newspaper     0
Sales         0
```

dtype: int64

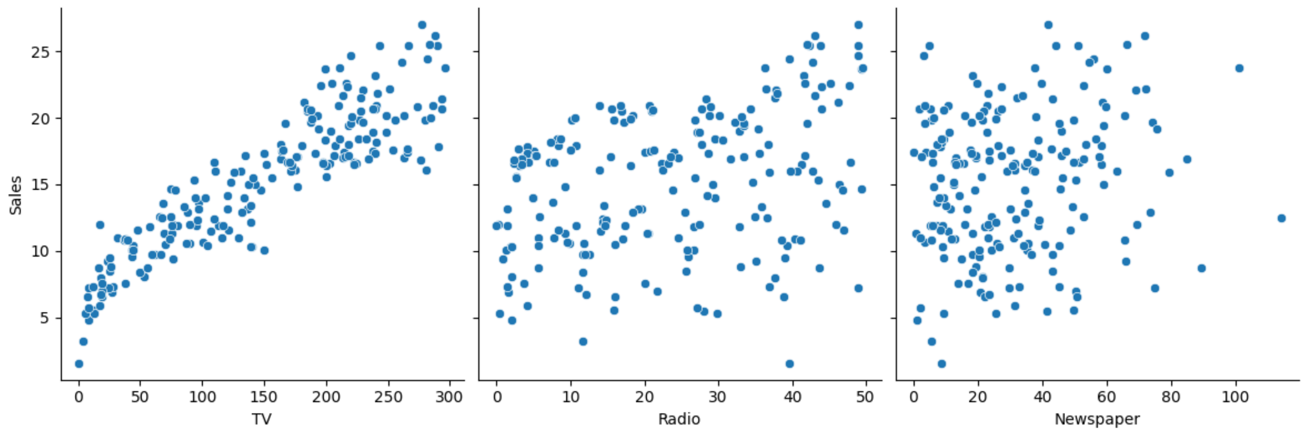
Statistical description of the dataset:

```
             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
```

Step 3: Visualize Relationships

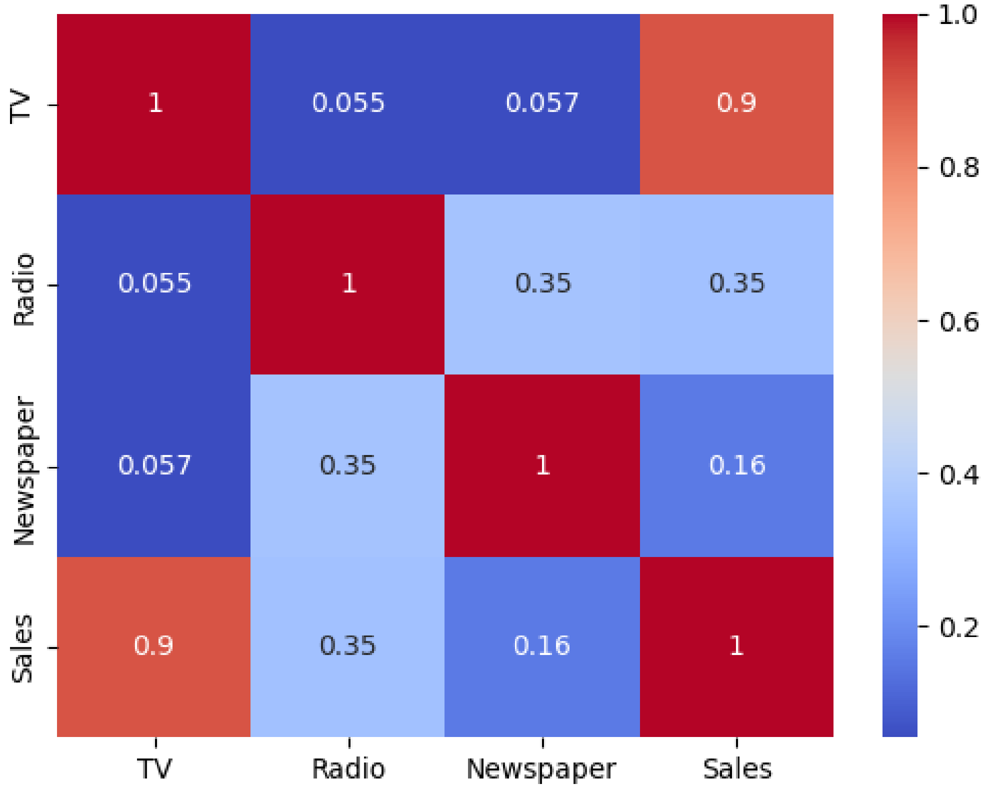
# Pairplot to see relationships between features and target

```
Visualize relationships between features (TV, Radio, Newspaper) and Sales using a
pairplot
sns.pairplot(data, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=4,
aspect=1, kind='scatter')
plt.show()  # Display the pairplot
```



# Correlation Heatmap

```
Visualize the correlation matrix using a heatmap
sns.heatmap(data.corr(), annot=True, cmap="coolwarm")  # Display correlation
matrix with annotations
plt.show()
```



Step 4: Splitting Data into Features and Target Variable

```
Split the dataset into features (X) and target variable (y)
X = data[['TV', 'Radio', 'Newspaper']]  # Independent variables (features)
y = data['Sales']  # Dependent variable (target: Sales)

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Step 5: Training the Linear Regression Model

```
Initialize and train the Linear Regression model
model = LinearRegression()  # Create a linear regression model
model.fit(X_train, y_train)  # Fit the model to the training data

# Output the intercept and coefficients of the trained model
print(f"Intercept: {model.intercept_}")  # Display the intercept (constant term)
print(f"Coefficients: {model.coef_}")
```

Output

Intercept: 4.714126402214127

Coefficients: [0.05450927 0.10094536 0.00433665]

Step 6: Making Predictions

```
y_pred = model.predict(X_test)  # Predict the sales on the test set
```

Step 7: Evaluating the Model

```
Evaluate the model using R-squared and RMSE (Root Mean Squared Error)
r2 = r2_score(y_test, y_pred)  # Calculate R-squared score (goodness of fit)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))  # Calculate RMSE (lower is
better)

print(f"R-squared: {r2:.2f}")  # Display the R-squared value
print(f"RMSE: {rmse:.2f}")  # Display the RMSE value
```

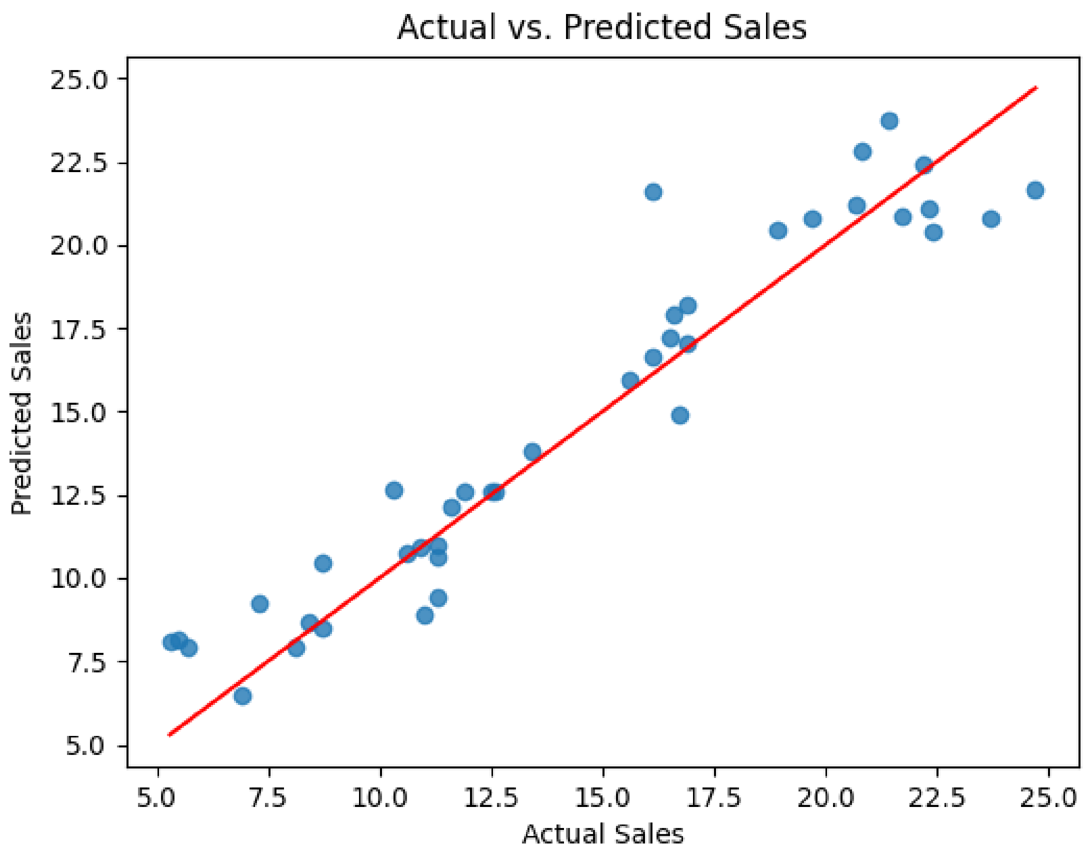
Output

R-squared: 0.91

RMSE: 1.71

Step 8: Visualizing Predictions

```
 Visualize the comparison of actual vs predicted sales using a scatter plot
plt.scatter(y_test, y_pred, alpha=0.8)  # Scatter plot for actual vs predicted
plt.xlabel("Actual Sales")  # Label for the x-axis
plt.ylabel("Predicted Sales")  # Label for the y-axis
plt.title("Actual vs. Predicted Sales")  # Title of the plot
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color="red")  # 
Line of perfect prediction
plt.show()
```



Conclusion

The analysis reveals that advertising on TV and Radio had the most significant impact on sales, as indicated by both the coefficients and the correlation heatmap. Newspaper advertising, however, showed a relatively lesser influence, suggesting an opportunity for optimizing budget allocation. The model’s performance is promising, with a high R-squared value, indicating that the model explains a significant portion of the variance in sales. Additionally, the RMSE highlights the average deviation between the predicted and actual sales. By leveraging this linear regression model, businesses can predict sales based on their advertising investments, enabling data-driven decisions for more efficient budget allocation and maximizing return on investment (ROI).