Task

SALES PREDICTION USING PYTHON

Sales prediction involves forecasting the amount of a product that customers will purchase, taking into account various factors such as advertising expenditure, target audience segmentation, and advertising platform selection. In businesses that offer products or services, the role of a Data Scientist is crucial for predicting future sales. They utilize machine learning techniques in Python to analyze and interpret data, allowing them to make informed decisions regarding advertising costs. By leveraging these predictions, businesses can optimize their advertising strategies and maximize sales potential. Let's embark on the journey of sales prediction using machine learning in Python.

Use the advertising dataset given in ISLR and analyse the relationship between 'TV advertising' and 'sales' using a simple linear regression model.

Imports Library

```
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_absolute_error, mean_squared_error,
r2_score
```

Step 1: Load and prepare data

```
data = pd.read_csv('advertising.csv')  # replace with your data file

data.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   12.0
```

Step 2 :Clean and preprocess the data

```
# Step 2: Check for missing values and handle them
# dataset does not contain null values
# If there were null values, we could fill or drop them like this:
```

```
# data.fillna(method='ffill', inplace=True)  # forward fill missing values

data.isnull().sum()
```

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

Step 3: Split the data into features and target

```
X = data[['TV', 'Radio', 'Newspaper']]  # Features
y = data['Sales']  # Target variable
```

Step 4: 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)
```

Step 5: Train a linear regression model

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

Step 6: Make predictions

```
y_pred = model.predict(X_test)
```

Step 7: Evaluate the model

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

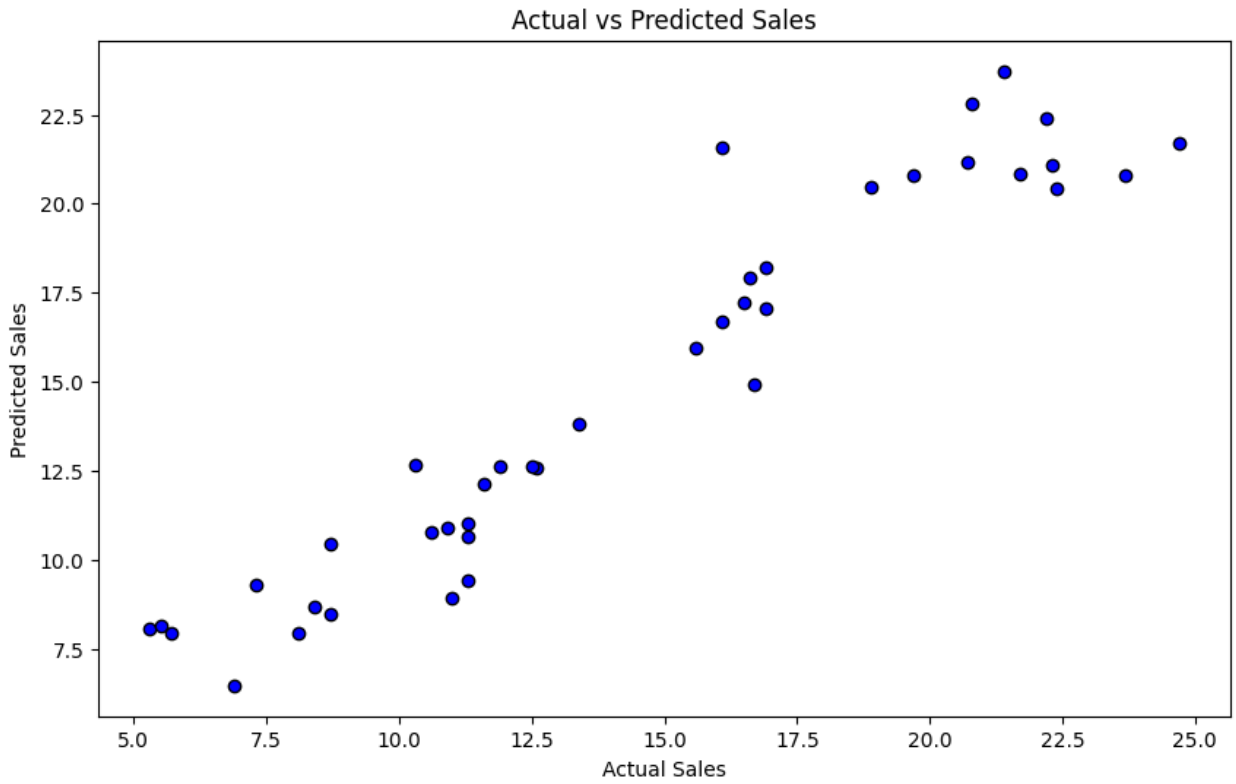
```
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² (Coefficient of Determination): {r2}")
```

```
Mean Absolute Error (MAE): 1.2748262109549338
Mean Squared Error (MSE): 2.9077569102710896
Root Mean Squared Error (RMSE): 1.7052146229349223
R² (Coefficient of Determination): 0.9059011844150826
```

Step 8: Visualize the results

1. Scatter plot of Actual vs Predicted Sales

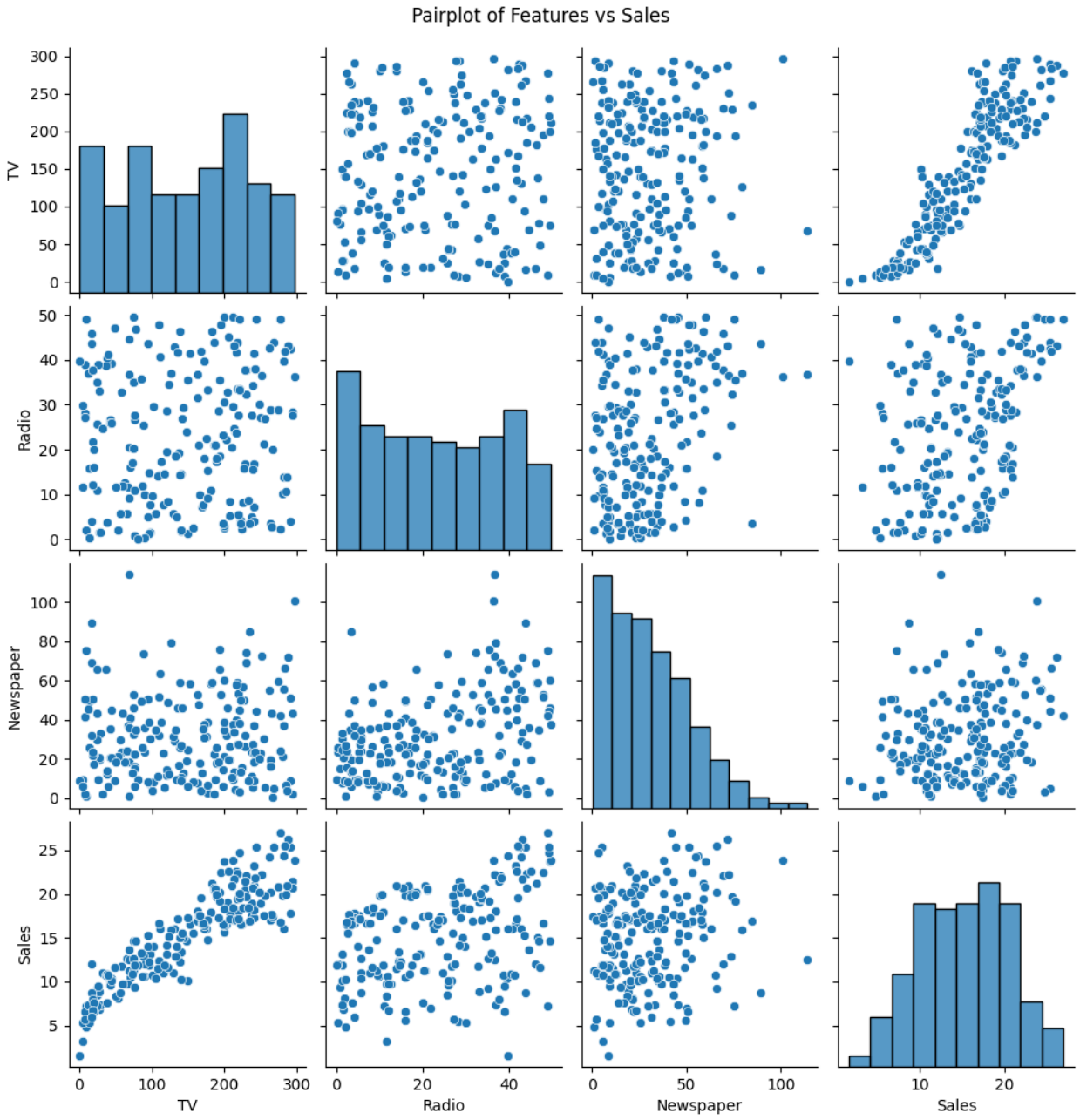
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', edgecolor='black')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.show()
```



2. Visualize the relationship between TV, Radio, Newspaper with Sales

Create pairplot or individual scatter plots for each feature

```
sns.pairplot(data[['TV', 'Radio', 'Newspaper', 'Sales']])
plt.suptitle('Pairplot of Features vs Sales', y=1.02)
plt.show()
```



3. Correlation heatmap to understand the relationships between features and sales

```
corr_matrix = data.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
```

```
plt.title('Correlation Matrix')
plt.show()
```