Task 04 : SALES PREDICTION USING PYTHON

Description: I have used the sales prediction dataset to build a model for Sales prediction as it involves forcasting the amount of a product that customers will purchase, taking into account vairous factors such as advertising platform selection.

FLOW ANALYSIS:

- Importing Libraries
- Data loading
- Data Understanding
- Data Visualization
- Spliting training and test data
- Scaling
- Model training -Linear Regression
- Model Evaluation - Prediction

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Importing all the required libraries
import numpy as np
import pandas as pd
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.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
```

```
# Data Loading
sales_data = pd.read_csv('/content/drive/MyDrive/CodSoft/advertising.csv')
```

```
# Displaying first 5 rows of the dataset
sales_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

```
sales_data.shape
```

```
(200, 4)
```

```
# Displaying information regarding datatype, null values of every column
sales_data.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
```

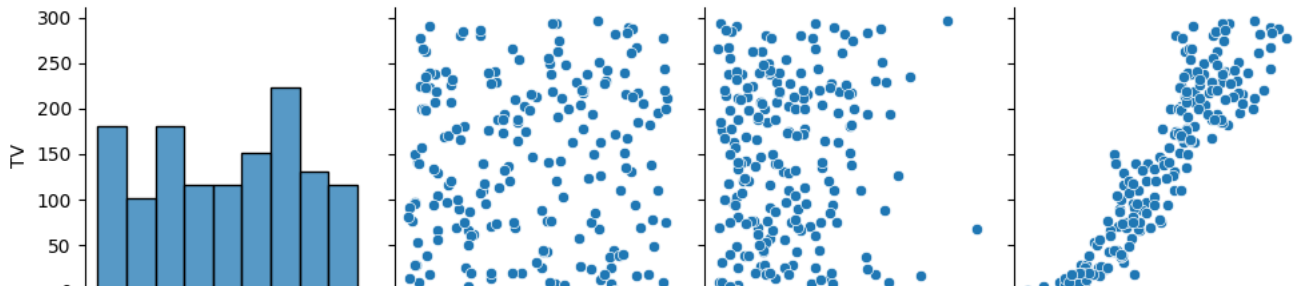
```
# It will calculate and display count, mean, std, min, max, 25%, 50% and 75% of numeric columns
sales_data.describe()
```

	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

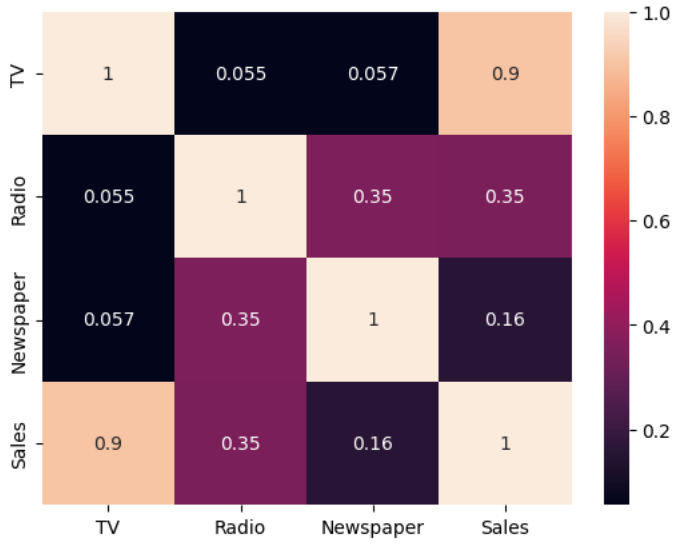
```
# Checking for null values
sales_data.isnull().sum()
```

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

```
# Pairplot to visualize relationships between features
sns.pairplot(sales_data)
plt.show()
```




```
# Heatmap
correlation_matrix = sales_data.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```




```
X=sales_data.drop(columns=['Sales'],axis=1)
Y=sales_data['Sales']
```

```
print(X)
```

```
        TV  Radio  Newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     ...    ...        ...
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7

[200 rows x 3 columns]
```

```
print(Y)
```

```
0      22.1
1      10.4
2      12.0
3      16.5
4      17.9
       ...
195     7.6
196    14.0
197    14.8
198    25.5
199    18.4
Name: Sales, Length: 200, dtype: float64
```

```
print(X.shape,Y.shape)
```

```
(200, 3) (200,)
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Data Splitting

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0.2, random_state=42)
```

```
# Model Training
model=LinearRegression()
model.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE) and R-squared (R2) score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 2.9077569102710914
R-squared: 0.9059011844150826
```

```
Prediction_model = [[100, 100, 100]]
predicted_sales = model.predict(Prediction_model)

print('Models Predicted Sales of TV, Radio, and Newspaper:', predicted_sales)
```

```
Models Predicted Sales of TV, Radio, and Newspaper: [640.94143574]
```