Problem Statement

As an owner of a startup, you wish to forecast the sales of your product to plan how much money should be spent on advertisements. This is because the sale of a product is usually proportional to the money spent on advertisements.

Predict the impact of TV advertising on your product sales by performing simple linear regression analysis.

List of Activities

Activity 1: Analysing the dataset

Activity 2: Train-Test split

Activity 3: Model training

Activity 4: Plotting the best fit line

Activity 5: Model prediction

Activity 1: Analysing the Dataset

Create a Pandas DataFrame for **Advertising-Sales** dataset using the below link. This dataset contains information about the money spent on the TV, radio and newspaper advertisement (in thousand dollars) and their generated sales (in thousand units). The dataset consists of examples that are divided by 1000.

Dataset Link: https://raw.githubusercontent.com/jiss-sngce/CO_3/main/advertising.csv

Also, print the first five rows of the dataset. Check for null values and treat them accordingly.

```
# Import modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Load the dataset
df=pd.read_csv('https://raw.githubusercontent.com/jiss-sngce/CO_3/main/advertising.csv'

# Print first five rows using head() function
```

```
df.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

```
# Check if there are any null values. If any column has null values, treat them accordi
df.isnull().sum()
```

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

Activity 2: Train-Test Split

For simple linear regression, consider only the effect of **TV ads** on sales. Thus, `TV` is the feature variable and `Sales` is the target variable.

Split the dataset into training set and test set such that the training set contains 67% of the instances and the remaining instances will become the test set.

```
# Split the DataFrame into the training and test sets.
from sklearn.model_selection import train_test_split
feature=df["TV"]
target=df['Sales']
feature_train,feature_test,target_train,target_test=train_test_split(feature,target,tes
feature_train
```

```
42     293.6
189     18.7
90     134.3
136     25.6
51     100.4
       ...
106     25.0
14     204.1
92     217.7
179    165.6
102    280.2
Name: TV, Length: 134, dtype: float64
```

Activity 3: Model Training

Train the simple regression model using **training data** to obtain the best fit line $y = mx + c$. For this, perform the following tasks:

1. Create following two functions:
   - A function `errors_product()` that calculates the errors for the feature and target variables i.e. $(x_i - \bar{x})(y_i - \bar{y})$
   - A function `squared_errors()` that calculates the squared errors for the feature variable only i.e. $(x_i - \bar{x})^2$
2. Calculate the **slope** and **intercept** values for the best fit line by applying the following formulae:

$$slope \Rightarrow m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{errors\_product().sum()}{squared\_errors().sum()}$$

$$intercept \Rightarrow c = \bar{y} - m\bar{x}$$

```
# Create the 'errors_product()' and 'squared_errors()' function.
def errors_product():
  prod=(feature_train-feature_train.mean())*(target_train-target_train.mean())
  return prod

def squared_errors():
  sq=(feature_train-feature_train.mean())**2
  return sq

```

```
# Calculate the slope and intercept values for the best fit line.
slope=errors_product().sum()/squared_errors().sum()
intercept=target_train.mean()-slope*feature_train.mean()
print('slope',round(slope,3))
print('intercept',round(intercept,3))

```

```
slope 0.056
intercept 7.127
```

Q: What is the equation obtained for the best fit line of this model?

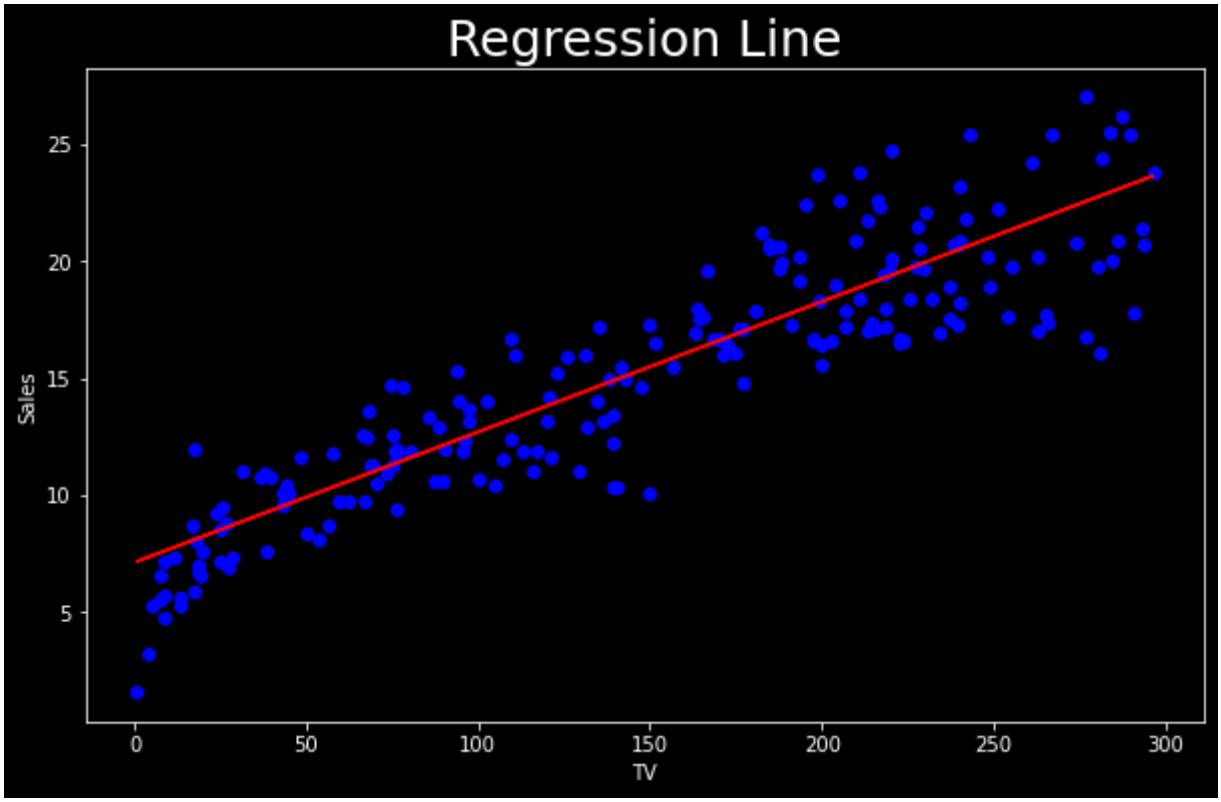
A: Sales=0.056*TV+7.12

Activity 4: Plotting the Best Fit Line

After obtaining the slope and intercept values for the best fit line, plot this line along with the scatter plot to see how well it fits the points.

```
#  Plot the regression line in the scatter plot between Sales and TV advertisment value
plt.style.use('dark_background')
plt.figure(figsize=(10,6))
plt.title('Regression Line',fontsize=25)
plt.scatter(df['TV'],df['Sales'],color='blue')
plt.plot(df['TV'],slope*df['TV']+intercept,color='r')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.show()

```



Activity 5: Model Prediction

For the TV advertising of $50,000, what is prediction for Sales? In order to predict this value, perform the following task:

- Based on the regression line, create a function `sales_predicted()` which takes a budget to be used for TV advertising as an input and returns the corresponding units of Sales.
- Call the function `sales_predicted()` and pass the amount spent on TV advertising.

Note: To predict the sales for TV advertising of $50,000, pass 50 as parameter to `sales_predicted()` function as the original data of this dataset consists of examples that are

divided by 1000. Also, the value obtained after calling `sales_predicted(50)` must be multiplied by 1000 to obtain the predicted units of sales.

```
#Create a function which takes TV advertisement value as an input and returns the sales
def sales_predicted(adv_tv):
  return 0.056*adv_tv+7.127


# Calculating sales value against $50,000 spent in TV ads
mysales=round(sales_predicted(50)*1000,0)
mysales

```

```
9927.0
```

Q: If you are planning to invest $50,000 dollars in TV advertising, how many unit of sales can be predicted according to this simple linear regression model?

A: 9927

```
#Deploy linear regression model using sklearn.linear_model
#1.Import the 'LinearRegression' class from the sklearn.linear_model

from sklearn.linear_model import LinearRegression
feature_train_res=feature_train.values.reshape(-1,1)
feature_train_res.shape
target_train_res=target_train.values.reshape(-1,1)
target_train_res.shape

#2.Create an object for LinearRegression class
model=LinearRegression()

#3.Call the fit() function.Fit function is used for training the model.Input to fit() s
model.fit(feature_train_res,target_train_res)

#4.Print the slope and intercept values
print(model.intercept_)
print(model.coef_)
```

```
[7.12701266]
[[0.05569894]]
```

```
feature_train.shape
target_train.shape
type(feature_train)
```

```
pandas.core.series.Series
```

✓ 0s completed at 12:15 PM