PROJECT On Data science/Machine Learning

```
#pandas(pd)-->for data manipulation and analysis.
import pandas as pd
#matplotlib.pylot--> to create various types of visualizations, such as line plots, bar plots, scatter plots.
import matplotlib.pyplot as plt
#numpy(np)-->Fundamental library for numerical computing, supporting arrays and mathematical functions.
import numpy as np
#seaborn(sns)-->Data visualization library based on Matplotlib, focused on statistical graphics.
import seaborn as sns
```

```
df=pd.read_csv("Movies_Ratings.csv")
#-->This code reads the data from a CSV file named "Movies_Ratings.csv" and stores it in a Pandas DataFrame called df.
```

```
df #dataframe
```

```
df.head()  #Displays the first few rows
```

	adult	budget	id	imdb_id	original_title	tagline	revenue	runtime	vote_ave
0	False	30000000	862	tt0114709	Toy Story	I Love You... The Way You Are.	373554033	81	
1	False	65000000	8844	tt0113497	Jumanji	Roll the dice and unleash the excitement! Still Yelling.	262797249	104	

```
df.tail()  #Displays the last few rows
```

	adult	budget	id	imdb_id	original_title	tagline	revenue	runtime	vote_averag
24	False	3600000	451	tt0113627	Leaving Las Vegas	I Love You... The Way You Are.	49800000	112	7
25	False	0	16420	tt0114057	Othello	Envy, greed, jealousy and love.	0	123	7

```
df.sample(5) #any 5 random samples
```

	adult	budget	id	imdb_id	original_title	tagline	revenue	runtime	vote_av
28	False	18000000	902	tt0112682	La Cité des Enfants Perdus	Where happily ever after is just a dream.	1738611	108	
18	False	30000000	9273	tt0112281	Ace Ventura: When Nature Calls	New animals. New adventures. Same hair.	212385533	90	
10	False	62000000	9087	tt0112346	The American President	Why can't the most powerful man in the	107879496	106	

```
df.dtypes #Shows the data types of each column in the DataFrame df.
```

```
adult               bool
budget             int64
id                 int64
```

```
imdb_id            object
original_title     object
tagline            object
revenue             int64
runtime             int64
vote_average      float64
vote_count          int64
popularity        float64
popularity.1      float64
dtype: object
```

```
df.info() #Provides a concise summary of the DataFrame df, including information about the data types, non-null counts, and memory usage.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29 entries, 0 to 28
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   adult           29 non-null     bool   
 1   budget          29 non-null     int64  
 2   id              29 non-null     int64  
 3   imdb_id         29 non-null     object 
 4   original_title  29 non-null     object 
 5   tagline         29 non-null     object 
 6   revenue         29 non-null     int64  
 7   runtime         29 non-null     int64  
 8   vote_average    29 non-null     float64
 9   vote_count      29 non-null     int64  
 10  popularity      29 non-null     float64
 11  popularity.1    29 non-null     float64
dtypes: bool(1), float64(3), int64(5), object(3)
memory usage: 2.6+ KB
```

```
df.count() #Returns the number of non-null values in each column of the DataFrame df.
```

```
adult             29
budget            29
id                29
imdb_id           29
original_title    29
tagline           29
revenue           29
runtime           29
vote_average      29
vote_count        29
popularity        29
popularity.1      29
dtype: int64
```

```
df.shape
#It is used to determine the dimensions or the size of an array or DataFrame.
#29 --> Rows
#12--> Columns
```

```
(29, 11)
```

```
df[df.duplicated()] #This selects all the duplicate rows in the DataFrame df.
```

	adult	budget	id	imdb_id	original_title	popularity	revenue	runtime	vote_average	vot

```
duplicate_rows_df=df[df.duplicated()]
print("Number of duplicate rows:", duplicate_rows_df.shape)
# This creates a new DataFrame called duplicate_rows_df containing only the duplicate rows from the original DataFrame df.
```

```
Number of duplicate rows: (0, 12)
```

```
df.columns #Returns a list of column names in the DataFrame df.
```

```
Index(['adult', 'budget', 'id', 'imdb_id', 'original_title', 'tagline',
       'revenue', 'runtime', 'vote_average', 'vote_count', 'popularity',
       'popularity.1'],
      dtype='object')
```

```
df.revenue  #This assumes that 'revenue' is a column in the DataFrame df. It retrieves the entire 'revenue' column.
```

```
0     373554033
1     262797249
2             0
3      81452156
4      76578911
5     187436818
6             0
7             0
8      64350171
9     352194034
10    107879496
11            0
12     11348324
13     13681765
14     10017322
15    116112375
16    135000000
17      4300000
18    212385533
19     35431113
20    115101622
21            0
22     30303072
23            0
24     49800000
25            0
26     27400000
27            0
28      1738611
Name: revenue, dtype: int64
```

Mean

```
mean_value = df['revenue'].mean()
print(mean_value)
#Calculates the mean (average) value of the 'revenue' column and assigns it to the variable mean_value.
```

```
78236641.55172414
```

Median

```
median_value = df['revenue'].median()
print(median_value)
# Calculates the median value of the 'revenue' column and assigns it to the variable median_value.
```

```
30303072.0
```

Mode

```
mode_value = df['revenue'].mode()
print(mode_value)
#Calculates the mode (most frequent value) of the 'revenue' column and assigns it to the variable mode_value.
```

```
0    0
Name: revenue, dtype: int64
```

Standard Deviation

```
std_value = df['revenue'].std()
print(std_value)
#Calculates the standard deviation of the 'revenue' column and assigns it to the variable std_value.
```

```
105973899.643369
```

```
df.describe()
#Provides a summary of descriptive statistics for each numeric column in the DataFrame df
```

	budget	id	revenue	runtime	vote_average	vote_count	popula
count	2.900000e+01	29.000000	2.900000e+01	29.000000	29.000000	29.000000	29.00
mean	2.766724e+07	10103.068966	7.823664e+07	115.724138	6.562069	623.448276	9.34
std	2.771504e+07	9935.642829	1.059739e+08	26.744688	0.720324	1091.036322	4.70
min	0.000000e+00	5.000000	0.000000e+00	78.000000	5.400000	33.000000	1.84
25%	0.000000e+00	1408.000000	0.000000e+00	101.000000	6.100000	137.000000	6.31
50%	1.800000e+07	9263.000000	3.030307e+07	106.000000	6.500000	210.000000	9.02

```
df.sort_values(['adult', 'budget', 'id', 'imdb_id', 'original_title', 'popularity',
       'revenue', 'runtime', 'vote_average', 'vote_count', 'tagline'])

#it is used to sort values of the dataframe df.
```

	adult	budget	id	imdb_id	original_title	tagline	revenue	runtime	v
21	False	0	1710	tt0112722	Copycat	One man is copying the most notorious killers ...	0	124	

```
df["budget"].unique()
#Returns an array containing all unique values
```

```
array([30000000, 65000000,        0, 16000000, 60000000, 58000000,
       35000000, 62000000, 44000000, 98000000, 52000000, 16500000,
        4000000, 30250000, 50000000,  3600000, 12000000, 18000000])
```

```
df.isnull().sum()
#returns the sum of these missing values for each column
```

```
adult             0
budget            0
id                0
imdb_id           0
original_title    0
tagline           0
revenue           0
runtime           0
vote_average      0
vote_count        0
popularity        0
popularity.1      0
dtype: int64
```

```
df.drop(["popularity.1"],axis=1,inplace=True)
#Drops the column named "popularity.1" from the DataFrame df.
#The axis=1 argument indicates that the operation is performed along columns
#inplace=True modifies the DataFrame directly.
```

```
df.head()
```

	country	date	value	timestamp
0	Brazil	01/01/2019	0.096799	1546300800
1	China	01/01/2019	14.816100	1546300800
2	EU27 & UK	01/01/2019	1.886490	1546300800
3	France	01/01/2019	0.051217	1546300800
4	Germany	01/01/2019	0.315002	1546300800

```
df.dropna(inplace=True)
#Removes rows with any missing values from the DataFrame df.
```

```
df.isnull().sum()
```

```
adult             0
budget            0
id                0
imdb_id           0
original_title    0
tagline           0
revenue           0
runtime           0
vote_average      0
vote_count        0
popularity        0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29 entries, 0 to 28
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
```

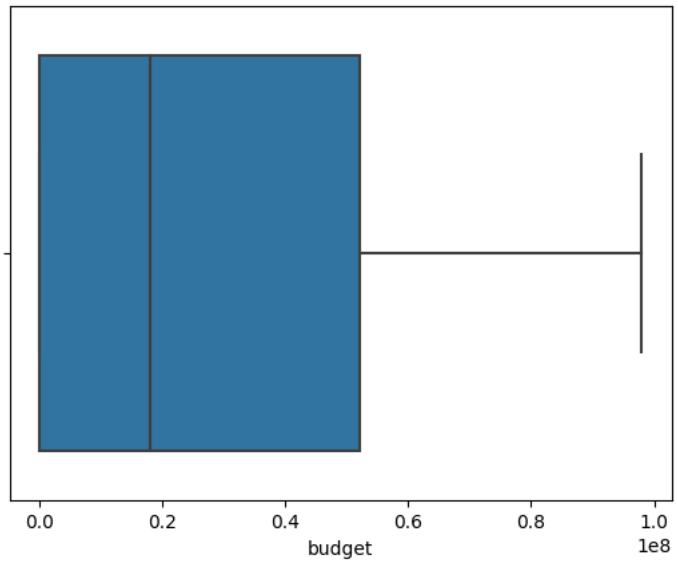
```
---  ------          --------------  -----
 0   adult           29 non-null     bool
 1   budget          29 non-null     int64
 2   id              29 non-null     int64
 3   imdb_id         29 non-null     object
 4   original_title  29 non-null     object
 5   tagline         29 non-null     object
 6   revenue         29 non-null     int64
 7   runtime         29 non-null     int64
 8   vote_average    29 non-null     float64
 9   vote_count      29 non-null     int64
 10  popularity      29 non-null     float64
dtypes: bool(1), float64(2), int64(5), object(3)
memory usage: 2.4+ KB
```

Detecting Outliers

Outliers--> Outliers are data points that deviate significantly from the majority of the dataset and can potentially impact data analysis and modeling.

```
sns.boxplot(x=df['budget'])
```

```
<Axes: xlabel='budget'>
```



Upon analyzing the data in the "budget" column, no outliers were found. The data appears to be uniformly distributed without any extreme values that could be considered outliers. And non of the coloumns have outliers.

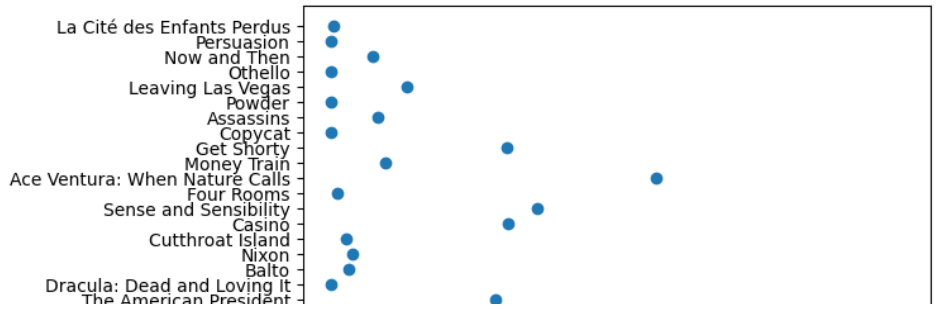
We can detect outliers by visual inspection of the box plot,scatter plot and Histograms

Exploratory Data Analysis(EDA)

It is the initial step in data analysis, where the main goal is to gain insights and understand the data before applying any formal modeling or statistical techniques.

```
#SCATTER PLOT
#Visualizes the relationship between two continuous variables by plotting individual data points as dots on a 2D plane.
plt.scatter(df['revenue'],df['original_title'])
```

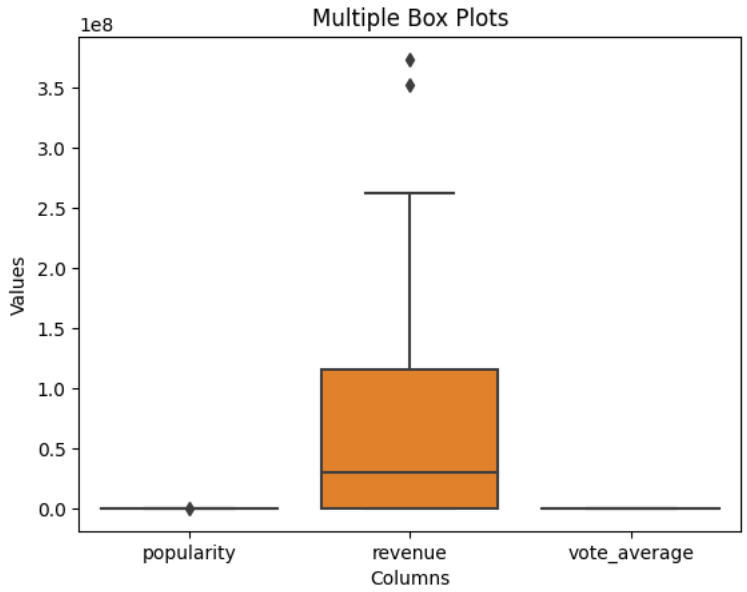
```
<matplotlib.collections.PathCollection at 0x7e69810b2e90>
```




```
#BOX PLOT
#Illustrates the distribution of data, showing the median, quartiles, and possible outliers.
sns.boxplot(data=df[['popularity', 'revenue', 'vote_average']])

# Set plot labels and title
plt.xlabel('Columns')
plt.ylabel('Values')
plt.title('Multiple Box Plots')
```

```
Text(0.5, 1.0, 'Multiple Box Plots')
```

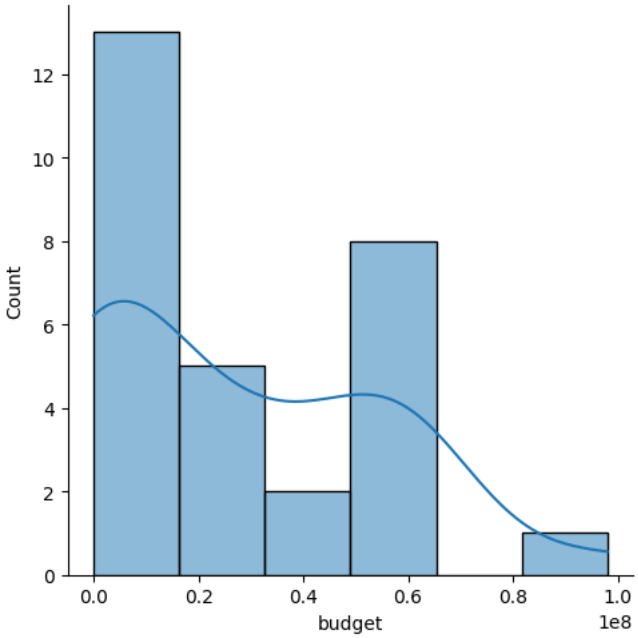



```
#PIE CHART
#Represents the proportion of each category relative to the whole using slices of a circle.
pie=df['original_title'].value_counts()
pie
pie.plot(kind="pie",autopct="%.2f%%")
```

```
<Axes: ylabel='original_title'>
```

```
#DIS PLOT
#Combines a histogram and a kernel density plot to show the distribution of a continuous variable.
sns.displot(df['budget'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e69e9f85b10>
```

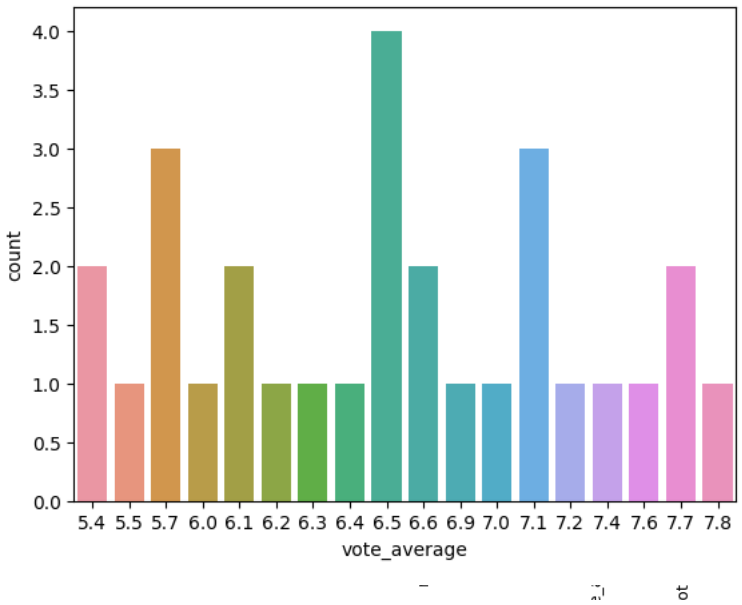



```
#HEATMAPS
#Displays a 2D representation of data using colors to visualize the intensity or correlation between two variables in a matrix-like format.
plt.figure(figsize=(10,5))
m= df.corr()
sns.heatmap(m,cmap="crest",annot=True)
m
```

```
<ipython-input-323-2e798e60d180>:2: FutureWarning: The default value of numeric_only in Data
  m= df.corr()
```

	adult	budget	id	revenue	runtime	vote_average	vote_count	popul
adult	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
budget	NaN	1.000000	-0.362167	0.381001	0.383734	-0.043858	0.244939	0.2
id	NaN	-0.362167	1.000000	-0.334727	-0.231079	-0.417071	-0.341587	-0.5
revenue	NaN	0.381001	-0.334727	1.000000	0.011378	0.304529	0.801702	0.6
runtime	NaN	0.383734	-0.231079	0.011378	1.000000	0.247026	0.185586	0.0

```
#COUNT PLOT
# Displays the count of occurrences of each category in a categorical variable using bars.
sns.countplot(x='vote_average',data=df)
plt.show()
```



Data Preprocessing

```
x=df.iloc[:,0:5] ##2D(Independent)
x
```

	adult	budget	id	imdb_id	original_title
0	False	30000000	862	tt0114709	Toy Story
1	False	65000000	8844	tt0113497	Jumanji
2	False	0	15602	tt0113228	Grumpier Old Men
3	False	16000000	31357	tt0114885	Waiting to Exhale
4	False	0	11862	tt0113041	Father of the Bride Part II
5	False	60000000	949	tt0113277	Heat
6	False	58000000	11860	tt0114319	Sabrina
7	False	0	45325	tt0112302	Tom and Huck
8	False	35000000	9091	tt0114576	Sudden Death
9	False	58000000	710	tt0113189	GoldenEye
10	False	62000000	9087	tt0112346	The American President
11	False	0	12110	tt0112896	Dracula: Dead and Loving It

```
y=df.iloc[:,5] ##1D(Dependent)
y
```

```
0                        I Love You... The Way You Are.
1              Roll the dice and unleash the excitement!
2     Still Yelling. Still Fighting. Still Ready for...
3     Friends are the people who let you be yourself...
4     Just When His World Is Back To Normal... He's ...
5                              A Los Angeles Crime Saga
6     You are cordially invited to the most surprisi...
7                               The Original Bad Boys.
8                            Terror goes into overtime.
9                 No limits. No fears. No substitutes.
10    Why can't the most powerful man in the world h...
11                       I Love You... The Way You Are.
12                       Part Dog. Part Wolf. All Hero.
13    Triumphant in Victory, Bitter in Defeat. He Ch...
14    The Course Has Been Set. There Is No Turning B...
15                     No one stays at the top forever.
16             Lose your heart and come to your senses.
17    Twelve outrageous guests. Four scandalous requ...
18              New animals. New adventures. Same hair.
19                           Get on, or GET OUT THE WAY!
20    The mob is tough, but it’s nothing like show b...
21    One man is copying the most notorious killers ...
22    In the shadows of life, In the business of dea...
23    An extraordinary encounter with another human ...
24                       I Love You... The Way You Are.
25                      Envy, greed, jealousy and love.
26    In every woman there is the girl she left behind.
27                       I Love You... The Way You Are.
28              Where happily ever after is just a dream.
Name: tagline, dtype: object
```

Encoding

Categorical------> Numeric means ENCODING Types of Encoding:- 1.ordinal encoding 2.Label encoding 3.one Hot encoding 4.get dummies encoding

```
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
x[["adult", "budget", "id", "imdb_id", "original_title"]]=oe.fit_transform(x[["adult", "budget", "id", "imdb_id", "original_title"]])
x
```

	adult	budget	id	imdb_id	original_title
0	0.0	7.0	4.0	27.0	27.0
1	0.0	16.0	11.0	16.0	13.0
2	0.0	0.0	23.0	14.0	11.0
3	0.0	4.0	27.0	28.0	28.0
4	0.0	0.0	20.0	10.0	7.0
5	0.0	14.0	6.0	15.0	12.0
6	0.0	13.0	19.0	24.0	22.0
7	0.0	0.0	28.0	1.0	26.0
8	0.0	9.0	13.0	26.0	24.0
9	0.0	13.0	3.0	13.0	10.0
10	0.0	15.0	12.0	2.0	25.0
11	0.0	0.0	21.0	9.0	6.0
12	0.0	0.0	26.0	4.0	2.0
13	0.0	10.0	17.0	19.0	17.0
14	0.0	17.0	7.0	8.0	5.0
15	0.0	12.0	2.0	5.0	3.0
16	0.0	5.0	9.0	25.0	23.0
17	0.0	2.0	0.0	11.0	8.0
18	0.0	7.0	15.0	0.0	0.0
19	0.0	14.0	18.0	18.0	16.0
20	0.0	8.0	10.0	12.0	9.0
21	0.0	0.0	8.0	7.0	4.0
22	0.0	11.0	16.0	3.0	1.0

```
df.head()
```

	adult	budget	id	imdb_id	original_title	tagline	revenue	runtime	vote_ave
0	False	30000000	862	tt0114709	Toy Story	I Love You... The Way You Are.	373554033	81	
1	False	65000000	8844	tt0113497	Jumanji	Roll the dice and unleash the excitement! Still Yelling.	262797249	104	

Model Building

Model building is the process of creating and training a predictive or statistical model using data to make accurate predictions or provide insights.

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets (80% training, 20% testing)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

LINEAR REGRESSION

Linear Regression (LR) is a supervised machine learning algorithm used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation.

```
from sklearn.linear_model import LinearRegression
#scikit_learn


x = df[['popularity']]
y = df['revenue']


# Check for and handle non-numeric values in the 'revenue' column
df['revenue'] = pd.to_numeric(df['revenue'], errors='coerce')  # Convert non-numeric values to NaN
# Drop rows with NaN values in the 'revenue' column
df = df.dropna(subset=['revenue'])



# Create a LinearRegression model
model = LinearRegression()

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

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



from sklearn.metrics import mean_squared_error, r2_score
# Evaluate the model performance
#Mean Squared Error (MSE) measures the average squared difference between predicted and actual values.
#R-squared (R²) indicates the proportion of variance explained by the model.
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: 1.0160633265107042e+16
R-squared: 0.3135233388361889
```

✓ 0s completed at 12:25 AM