Name:Atharva Shinde Roll No:54 Div:D15C

EXP₁

Aim:

- Load data in Pandas.
- Description of the dataset.
- Drop columns that aren't useful.
- Drop rows with maximum missing values.
- Take care of missing data.
- Create dummy variables.
- Find out outliers (manually)
- standardization and normalization of columns

Data preprocessing

Data preprocessing involves transforming raw data into a structured and meaningful format, making it suitable for analysis. It is a crucial step in data mining, as raw data often contains inconsistencies, missing values, or noise. Ensuring data quality is essential before applying machine learning or data mining algorithms to achieve accurate and reliable results.

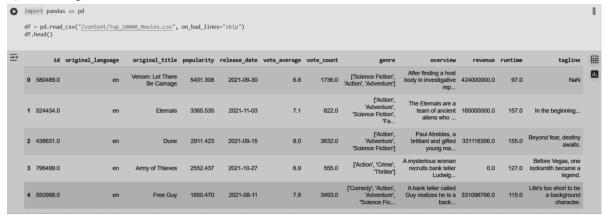
Why is Data Preprocessing Important?

Data preprocessing is essential for ensuring the quality and reliability of data before analysis. It helps improve the accuracy and efficiency of machine learning and data mining processes. The key aspects of data quality include:

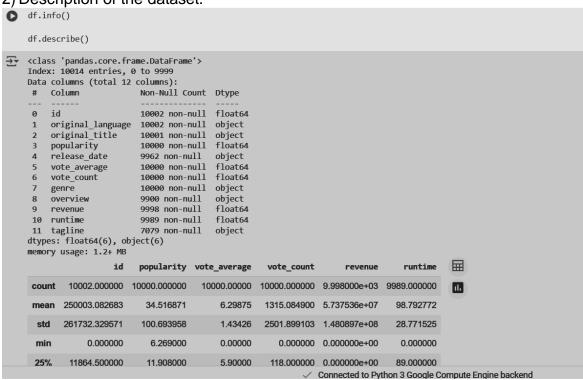
- Accuracy: Ensuring the data is correct and free from errors.
- Completeness: Checking for missing or unrecorded data.
- Consistency: Verifying that data remains uniform across different sources.
- Timeliness: Ensuring the data is up-to-date and relevant.
- **Believability:** Assessing whether the data is reliable and trustworthy.
- Interpretability: Making sure the data is clear and easy to understand.

Dataset: Top 10000 Movies

1) Loading Data in Pandas



2) Description of the dataset.



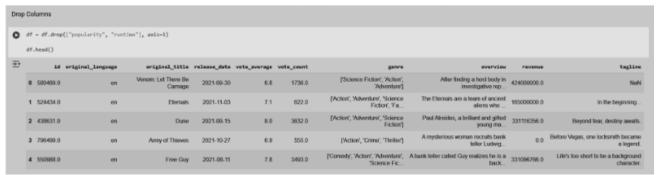
df.info(): Provides an overview of the dataset, including:

- Number of rows and columns.
- Data types of each column (e.g., int, float, object).
- Number of non-null (non-missing) values in each column.

df.describe(): Generates summary statistics for numeric columns, such as:

- count: Number of non-missing values.
- mean: Average value.
- std: Standard deviation.

- min, 25%, 50% (median), 75%, and max: Percentile values
- 3) Drop columns that aren't useful: Columns like Invoice ID may not contribute to analysis (it's often just an identifier). Removing irrelevant columns reduces complexity.

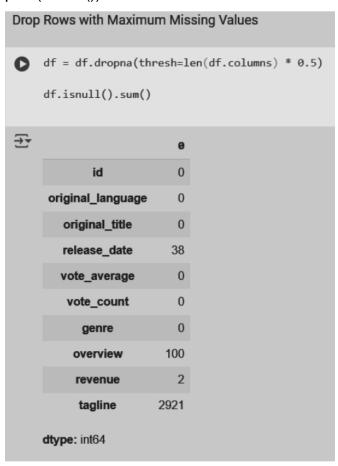


4) Drop rows with maximum missing values.

df.dropna(thresh=int(0.5 * len(df.columns))):

- Drops rows where more than half the columns have missing (NaN) values.
- thresh=int(0.5 * len(df.columns)): Ensures that a row must have at least 50% non-null values to remain.

df = ...: Updates the DataFrame after dropping rows. print(df.info()): Confirms that rows with excessive missing values have been removed.



5) Take care of missing data.

df.fillna(df.mean()): Replaces missing values (NaN) in numeric columns with the mean of that column.

Div:D15C

```
Handle Missing Data

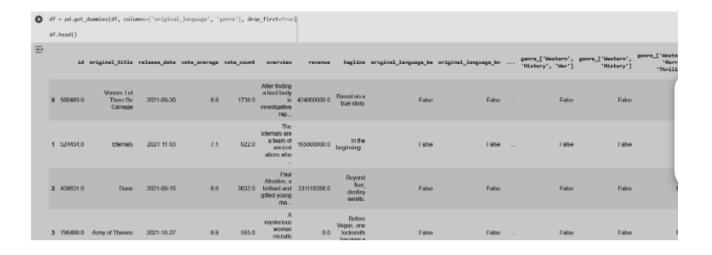
[ ] df["release_date"] = pd.to_datetime(df["release_date"], errors="coerce")

[ ] df.fillna(df.mode().iloc[0], inplace=True)
```

6) Create dummy variables.

pd.get_dummies(): Converts categorical variables into dummy variables (binary indicators: 0 or 1).

columns=['...']: Specifies which columns to convert. drop_first=True: Avoids the "dummy variable trap" by dropping one dummy variable to prevent multicollinearity.



7) Find out outliers (manually)

```
# Compute Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df["revenue"].quantile(0.25) # First Quartile (25%)
Q3 = df["revenue"].quantile(0.75) # Third Quartile (75%)
# Compute Interquartile Range (IOR)
IQR = Q3 - Q1
# Define lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
print(f"Q1 (25th percentile): {Q1}")
print(f"Q3 (75th percentile): {Q3}")
print(f"IQR (Interquartile Range): {IQR}")
print(f"Lower Bound: {lower_bound}")
print(f"Upper Bound: {upper bound}")
Q1 (25th percentile): 0.0
Q3 (75th percentile): 47645488.0
IQR (Interquartile Range): 47645488.0
Lower Bound: -71468232.0
Upper Bound: 119113720.0
```

8) standardization and normalization of columns

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Standardization equation

$$X' = \frac{X - \mu}{\sigma}$$

To standardize your data, we need to import the StandardScalar from the sklearn library and apply it to our dataset.

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

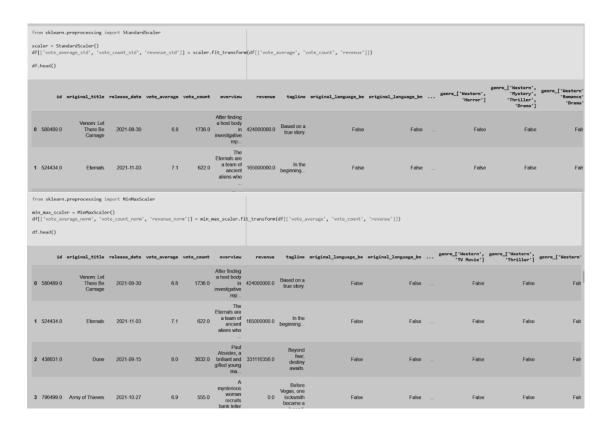
Normalization equation

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

To normalize your data, you need to import the MinMaxScalar from the sklearn library and apply it to our dataset.



Conclusion:

Thus we have understood how to perform data preprocessing which can further be taken into exploratory data analysis and further in the Model preparation sequence.