**DS USING PYTHON LAB**

**EXPERIMENT: 01**

**AIM:**Data preparation using NumPy and Pandas

**THEORY:**

Dataset selected: <https://www.kaggle.com/datasets/bharatnatrayn/movies-dataset-for-feature-extracion-prediction>

The above Dataset consists of details of various movies and information related to it. It contains detailed information consisting of the name of the movie, its year of release, genre, rating, runtime, votes and actors starring in it. There’s also a one line description of the movie.

**Numpy:**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

This is the foundation on which almost all the power of Python’s data science toolkit is built, and learning NumPy is the first step on any Python data scientist’s journey.

Features:

* Provides fast, precompiled functions for numerical routines
* Array-oriented computing for better efficiency
* Supports an object-oriented approach
* Compact and faster computations with vectorization

**Pandas:**

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Features:

* It has a fast and efficient DataFrame object with the default and customized indexing.
* Used for reshaping and pivoting of the data sets.
* Group by data for aggregations and transformations.
* It is used for data alignment and integration of the missing data.

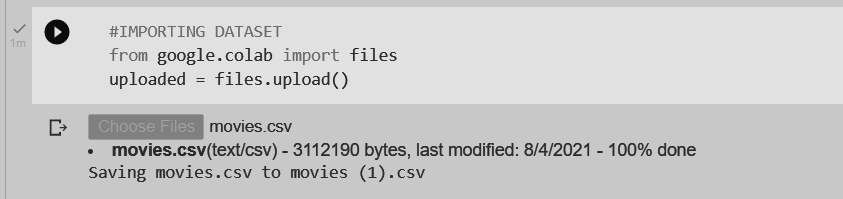
**COMMANDS:**

1. **Load data in Pandas.**

from google.colab import files

uploaded = files.upload()

>>The above command will enable us to upload the dataset file in the google collab notebook.



pd.read\_csv()

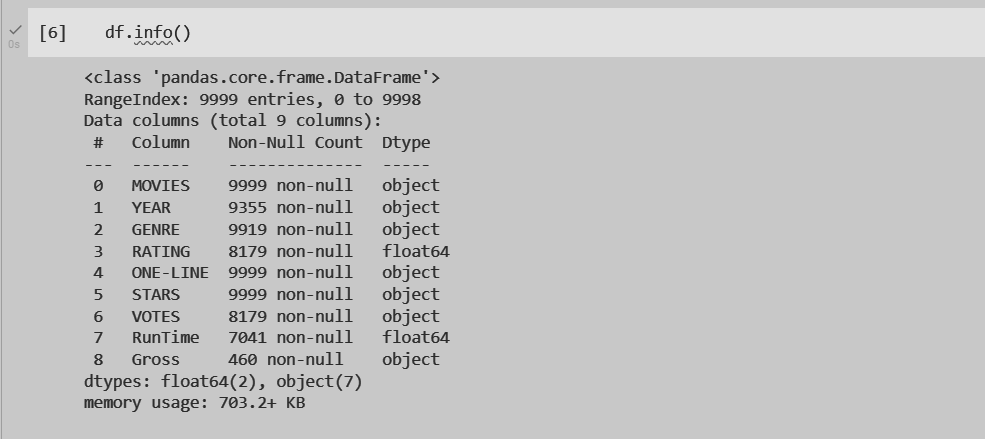
>>The following command will load the data in pandas and will show us some rows and columns from our dataset.



1. **Description of the dataset.**

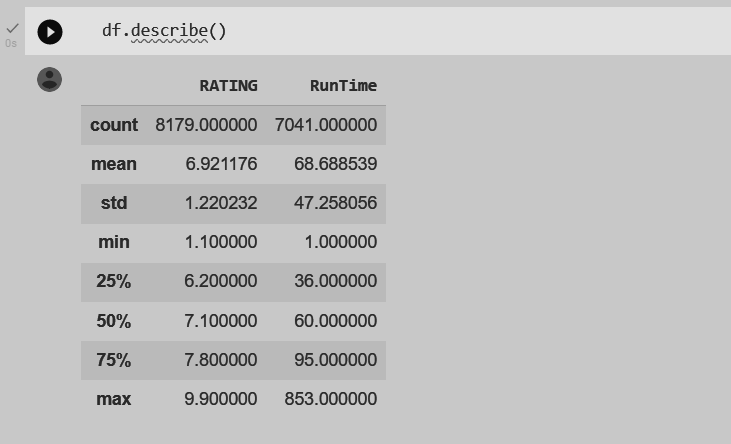
df.info()

>>This method prints information about a DataFrame including the index data type and columns, non-null values and memory usage.



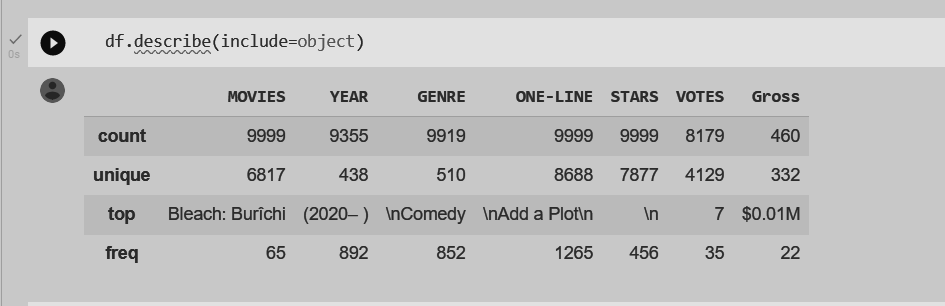
df.describe()

>>The describe() method returns a description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains this information for each column: count - The number of not-empty values. mean - The average (mean) value.



df.describe(include=object)

>>This command generates descriptive statistics. Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset’s distribution, excluding NaN values.



1. **Drop columns that aren’t useful.**

df.drop()

>>The drop() function is used to drop specified labels from rows or columns. Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names.

labels >> Index or column labels to drop.

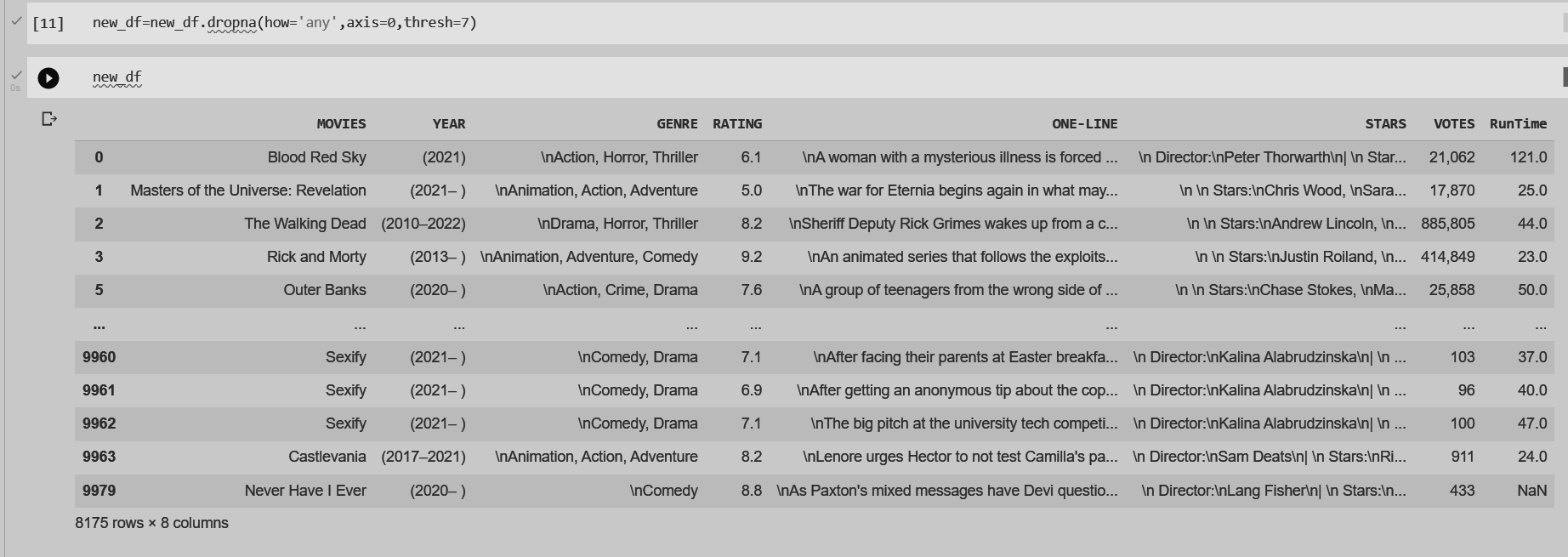
axis >> Whether to drop labels from the index (0 or ‘index’) or columns (1 or ‘columns’).



1. **Drop rows with maximum missing values.**

newdf=df.dropna(how='any')

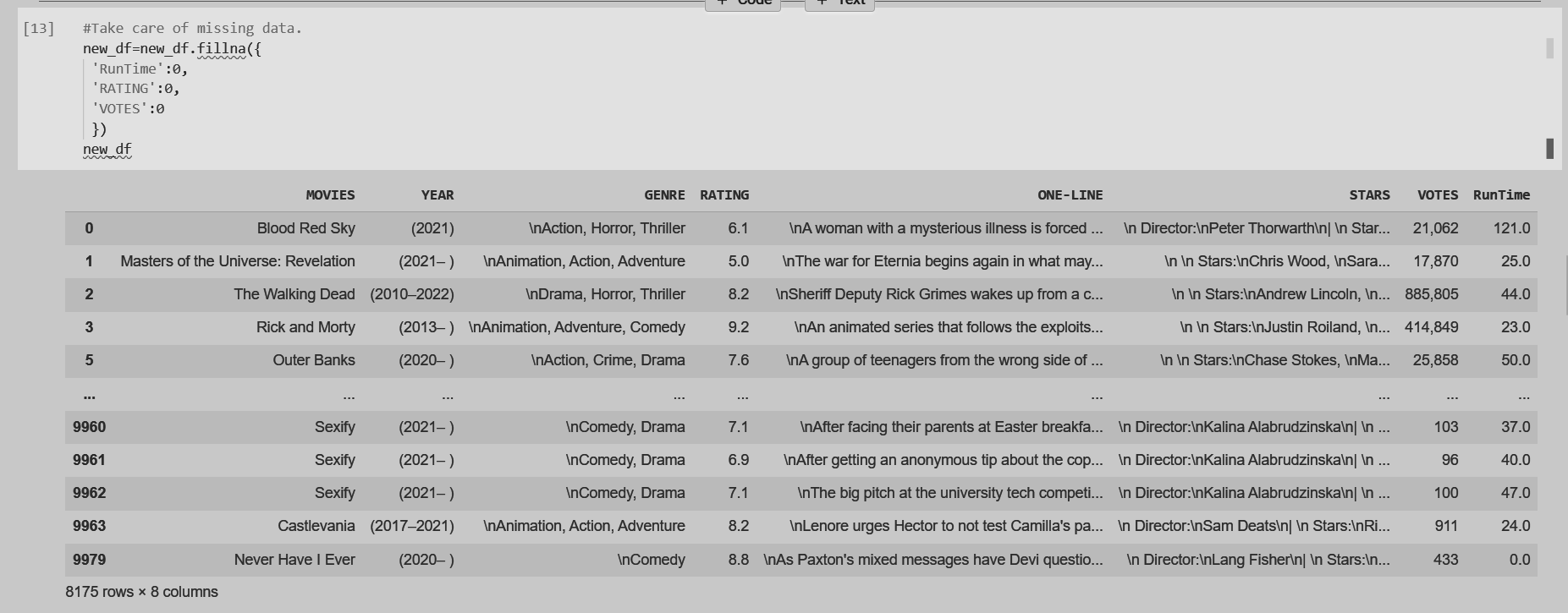
>> The dropna() method removes the rows that contain NULL values. The dropna() method returns a new DataFrame object unless the inplace parameter is set to True, in that case the dropna() method does the removing in the original DataFrame instead.



1. **Take care of missing data.**

new\_df=df.fillna(0)

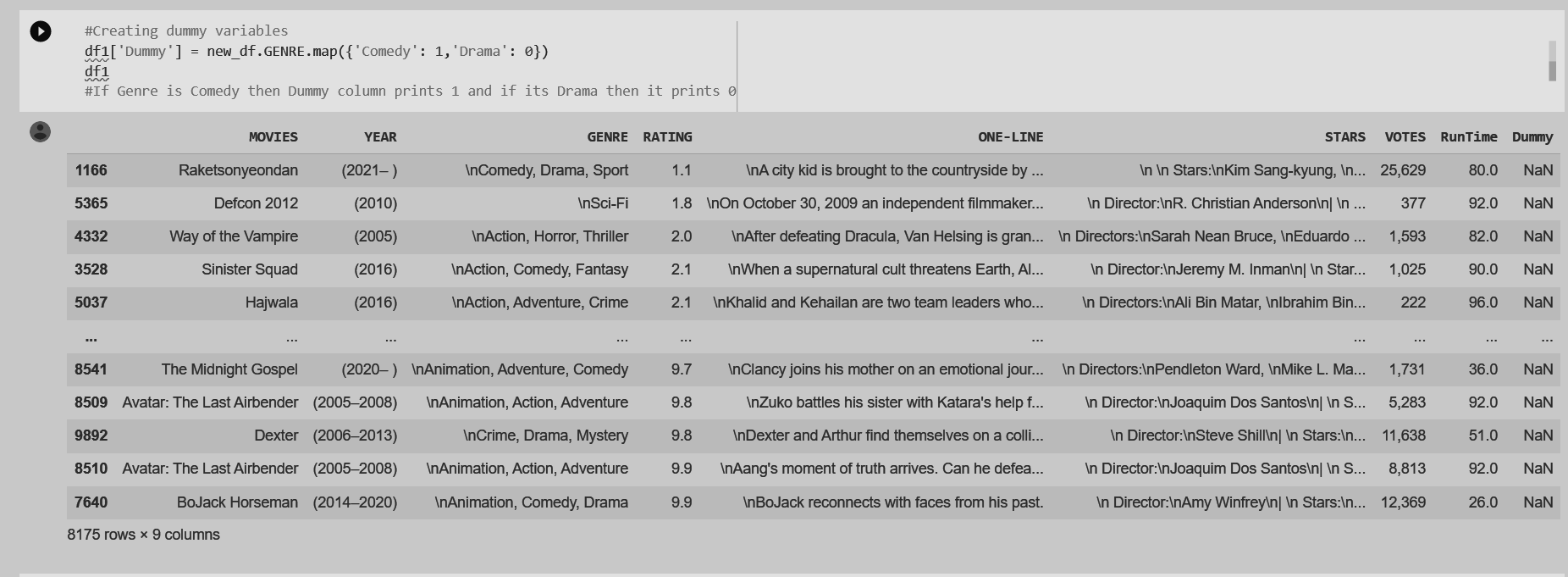
>> In pandas, the Data Frame provides a method fillna()to fill the missing values or NaN values in DataFrame. Whatever value you put in the () will be filled in the missing values



1. **Create dummy variables**.

new\_df=pd.get\_dummies(df)

>> pandas.get\_dummies() is used for data manipulation. It converts categorical data into dummy or indicator variables.

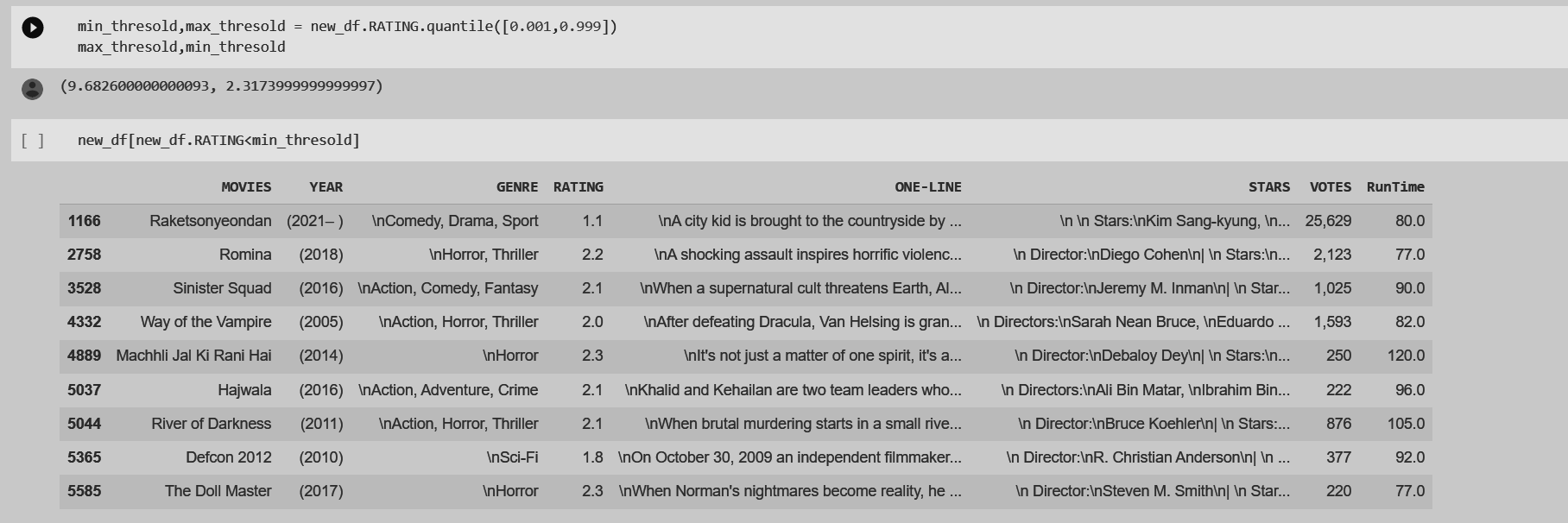


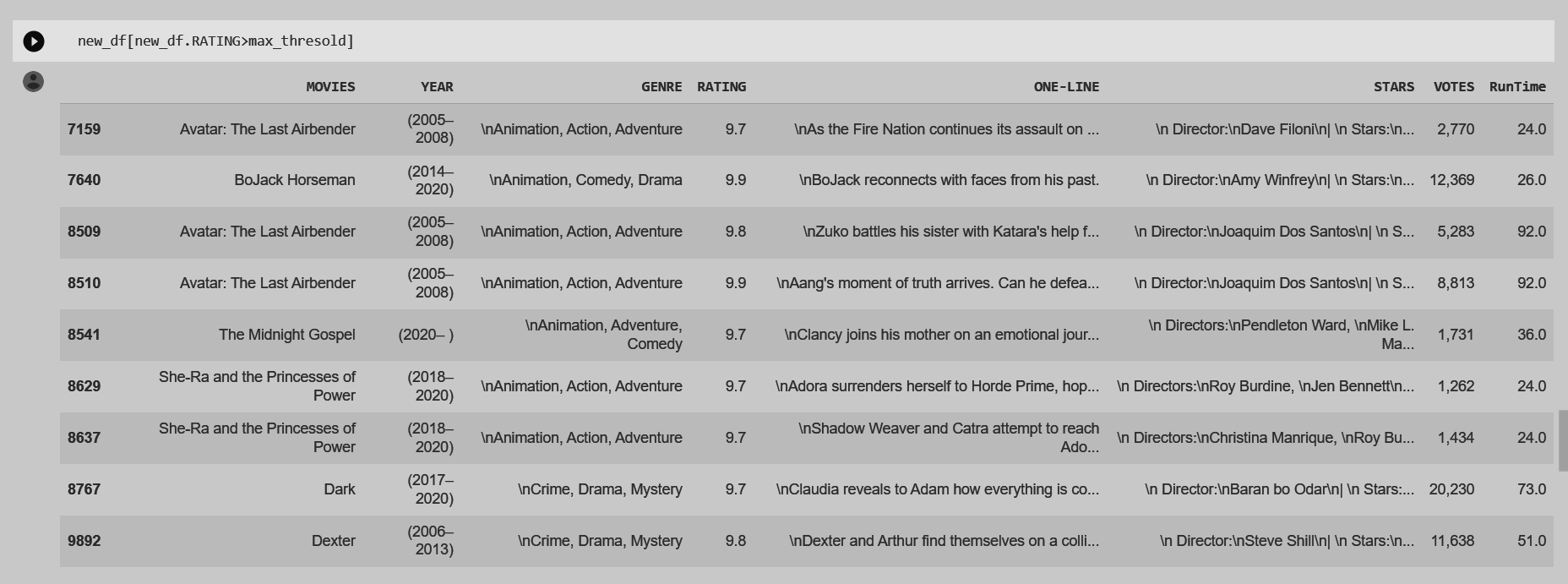
1. **Find out outliers (manually)**

In simple terms, an outlier is an extremely high or extremely low data point relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset you're working with. Outliers can give helpful insights into the data you're studying, and they can have an effect on statistical results. This can potentially help you discover inconsistencies and detect any errors in your statistical processes.

Methods:

1. Sorting your values from low to high and checking minimum and maximum values.
2. Using the interquartile range to create fences for your data.
3. Using statistical procedures to identify extreme values.

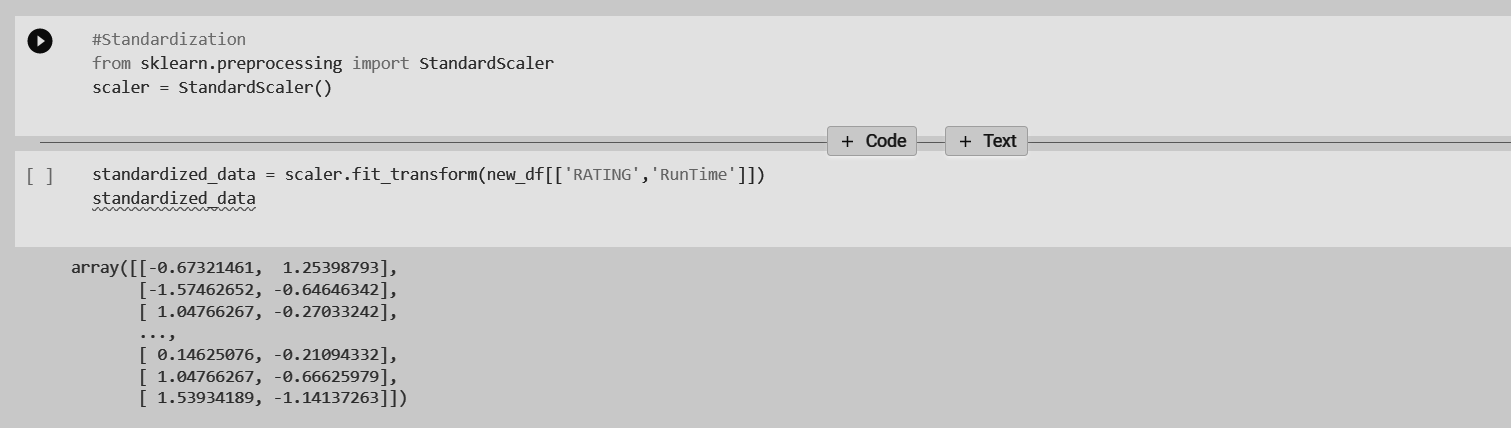


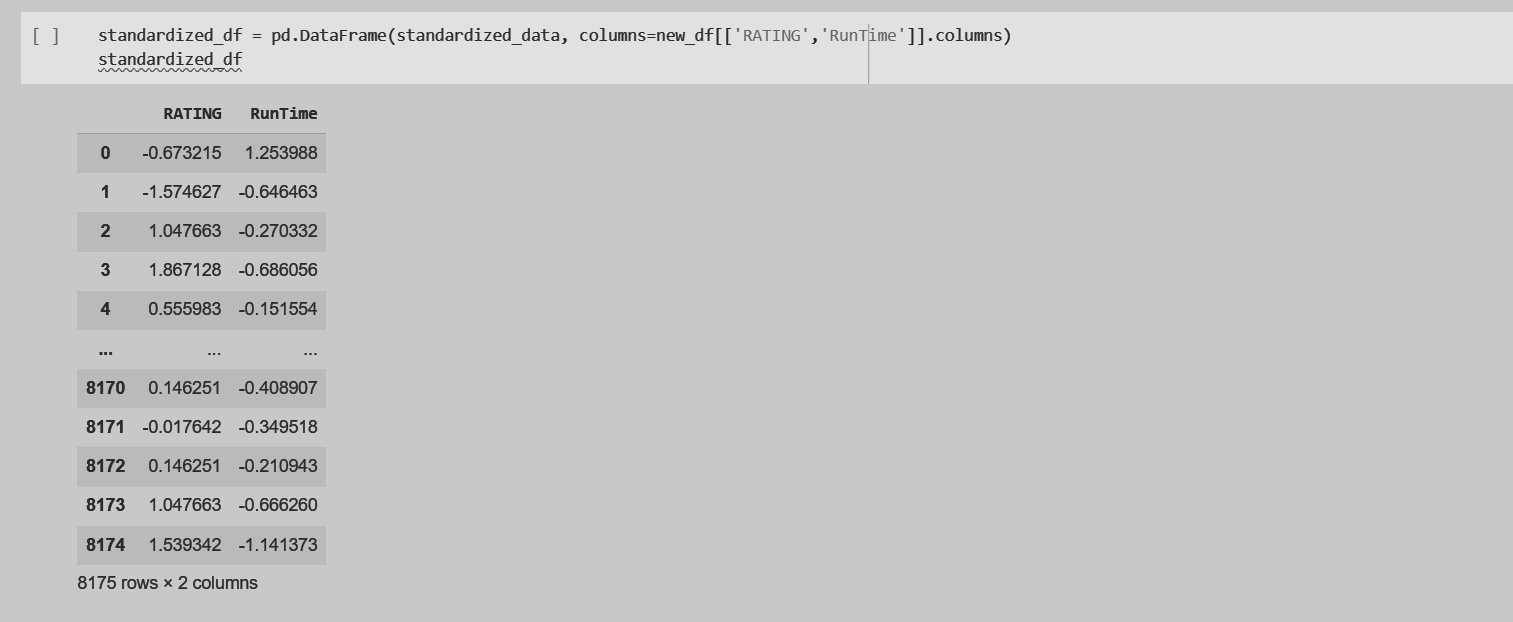


1. **Standardization and Normalization of Columns**

Standardization:

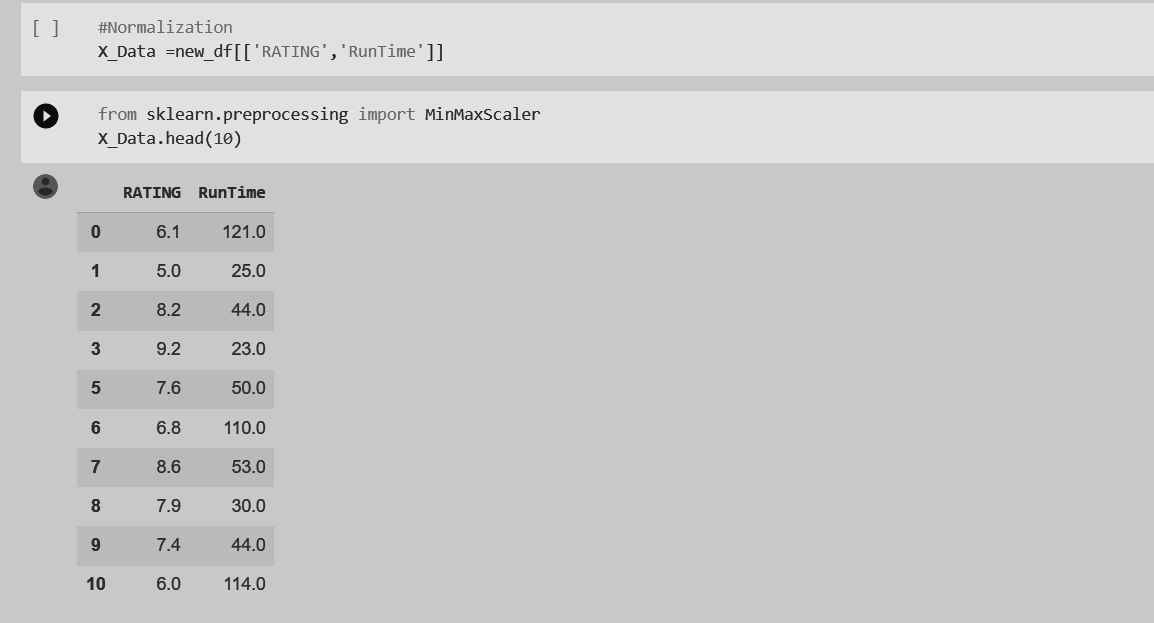
In statistics and machine learning, data standardization is a process of converting data to z-score values based on the mean and standard deviation of the data. The resulting standardized value shows the number of standard deviations the raw value is away from the mean.

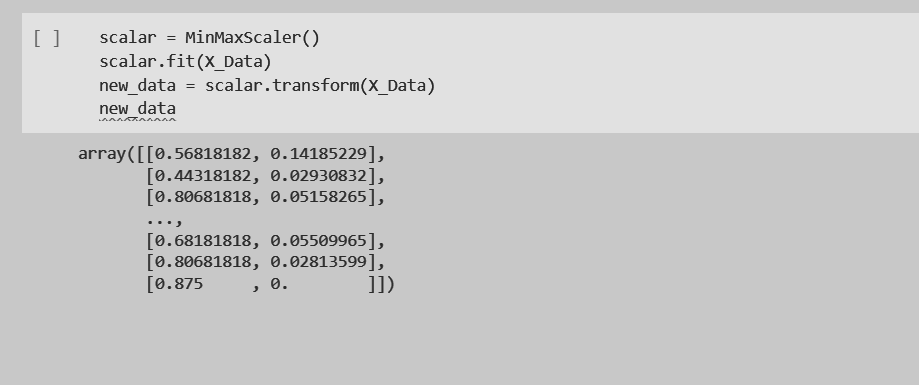




Normalization:

In statistics and machine learning, min-max normalization of data is a process of converting the original range of data to the range between 0 and 1. The resulting normalized values represent the original data on 0-1 scale. This will allow us to compare multiple features together and get more relevant information since now all the data will be on the same scale.





**CONCLUSION:**

In this experiment, we studied the dataset(movies.csv) thoroughly with the help of Python libraries - Numpy and Pandas. We were able to clean the data by dropping unnecessary columns and taking care of missing values by replacing them. We also recognized the outliers manually and studied standardization and normalization.