**NETFLIX\_TITLES ANALYSIS**

Importing Necessary Libraries for Data Analysis in Big Data.

The first step is importing various libraries that are commonly used in data analysis, visualization, and machine learning. These libraries provide essential tools and functionalities required to handle, manipulate, analyze, and visualize data efficiently. Each library has a specific role, ranging from data manipulation and visualization to executing complex statistical computations and model training. Using these tools correctly improves productivity and guarantees precise and insightful analysis. Few libraries and functions are explained below:

* Pandas (`pd`): A robust data manipulation and analysis library for Python, that provides data structures like DataFrame and Series used for data cleaning and transformation.

e.g. import pandas as pd || data = pd.read\_csv('data.csv')

* NumPy (`np`): A default package for numerical computations in Python which support arrays, matrices, and often used for performing mathematical operations on large datasets.

e.g. import numpy as np || array = np.array([1, 2, 3, 4])

* Matplotlib (`plt`): A comprehensive library for creating static, and interactive visualizations in Python. This can be used for plotting graphs and charts, such as line plots, bar charts, and scatter plots. e.g.

import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [50, 100, 150])

plt.show()

* Seaborn (`sns`): this is a Python visualization library based on Matplotlib providing a high-level interface for drawing attractive and informative statistical graphics. It is used for complex plots like heatmaps and pair plots. e.g.

import seaborn as sns

sns.heatmap(data.corr())

* Scikit-Learn

- TfidfVectorizer: This converts text data into numerical feature vectors. e.g.

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

vectors = vectorizer.fit\_transform(text\_data)

- Cosine Similarity: Measures the cosine similarity between vectors. e.g.

from sklearn.metrics.pairwise import cosine\_similarity

similarity = cosine\_similarity(vectors)

- MinMaxScaler: Scales features to a given range.

from sklearn.preprocessing import MinMaxScaler

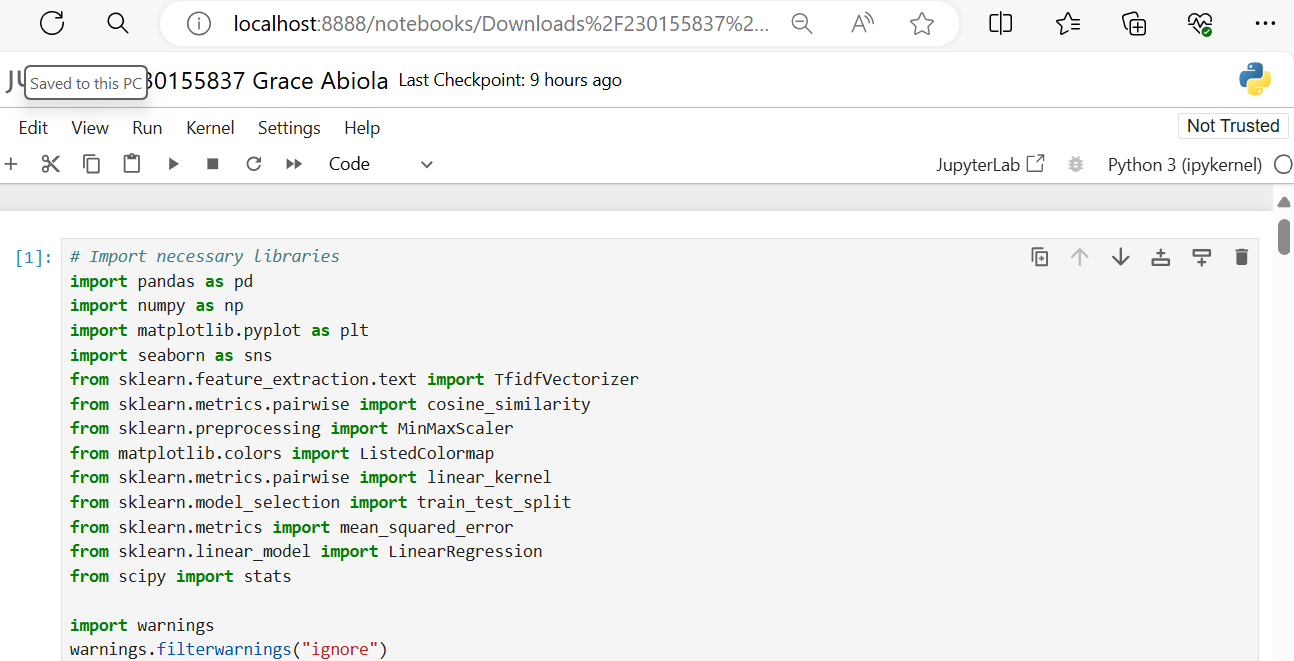
scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data)

* WarningsA module to manage warning messages used to filter out warning messages, making the output cleaner. E.g.

import warnings

warnings.filterwarnings("ignore")

****

**Fig 1.** Importing Necessary Libraries for Data Analysis

**Data Understanding and Pre-processing.**

Loading the dataset and displaying its initial entries is done using the following:

netflix\_data = pd.read\_csv('netflix\_titles.csv'). This reads the contents of the CSV file and loads it into a DataFrame, which is a tabular data structure with labeled axes (rows and columns). The loaded DataFrame is then assigned to the variable `netflix\_data` which contains all the data from the CSV file in a structured format, making it easier to manipulate and analyze.

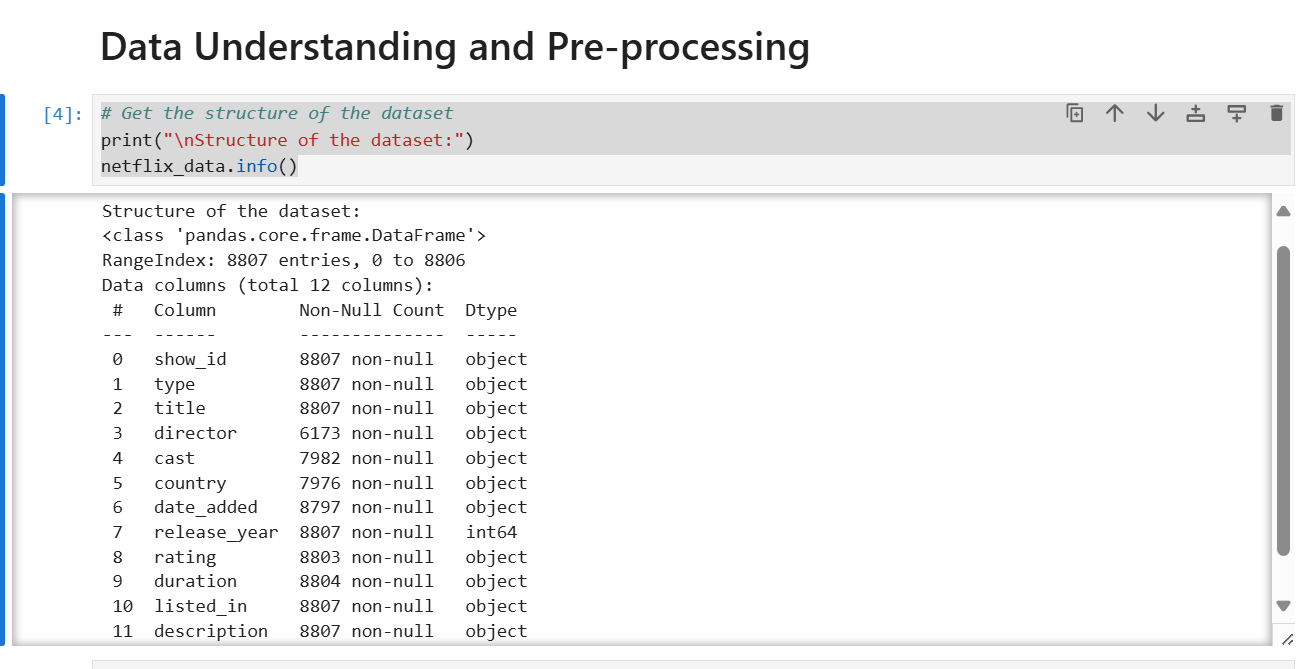
**Displaying the Initial Entries with netflix\_data.head()** - This head() method returns the first five rows of the DataFrame by default. It is useful for quickly inspecting the dataset to understand its structure and the type of data it contains. It also helps to verify that the data has been loaded correctly by previewing of the dataset's columns and a sample of its rows.

A screenshot of a computer

AI-generated content may be incorrect.

**Fig .2 Loading the dataset & Displaying the Initial Entries**

**Data Understanding and Pre-processing**

****

**Fig .3 Getting to know the data types of all columns**

The dataset contains 8807 rows (entries) and 12 columns (features).

Each row represents a piece of content available on Netflix (e.g., movies, TV shows).

The columns provide information about various aspects of the content.

There are some column descriptions stated below:

**show\_id:** A unique identifier for each content entry.

**type**: Indicates whether the content is a ‘Movie’ or a ‘TV Show’.

**title**: The title of the content.

**director**: The director(s) of the content (if available).

**cast:** The cast members (actors/actresses) in the content (if available).

A screenshot of a computer

AI-generated content may be incorrect.

**Fig .4** Summary Statistics for Each Column

The snippet above generates summary statistics for each column in a DataFrame. Here’s a brief explanation of the output:

**Count:** The number of non-null observations for each column.

**Unique:** The number of unique values in each column.

**Top**: The most frequent value in each column.

**Frequency (Freq):** The count of the most common value.

**Mean:** The average value for numeric columns.

**Std:** The standard deviation (spread) of the data.

**Min and Max:** The minimum and maximum values.

A white background with red text

AI-generated content may be incorrect.

**Fig .5 Getting to know the columns with missing values.**

**Column Names**

These columns encompass a variety of data types, including categorical, numerical, and textual data. Grasping the structure and data types in each column is essential for efficient data preprocessing and analysis.

The column names are as follows: ‘show\_id’, ‘type’, ‘title’, ‘director’, ‘cast’, ‘country’, ‘date\_added’, ‘release\_year’, ‘rating’, ‘duration’, ‘listed\_in’, and ‘description’.

Values: The numeric values represent the count of missing values (NaNs) in each column.

* ‘director’ column has 2634 missing values.
* ‘cast’ column has 825 missing values.
* ‘date\_added’ column has 10 missing values.

A screenshot of a computer

Description automatically generated

**Fig .5 Getting to know the columns with missing values.**

## Data Wrangling

Which leads us to the data wrangling and cleaning stage The main tasks include data cleaning, data transformation, and data exploration.

**Data Cleaning**

* **Handling Missing Values**: Identify and fill or remove missing values. For example, missing values in director and cast can be filled with 'Unknown'. Missing values in country can be imputed with the most frequent value or inferred from similar entries.
* **Removing Duplicates**: Ensure each entry is unique by removing duplicate rows.
* Correcting Inconsistencies: Standardize formats in columns like country, rating, and duration.

**Data Transformation**:

* **Encoding Categorical Variables**: Convert categorical data into numerical values. One-hot encoding can be used for *type* and *rating*. The *listed\_in* column, which contains multiple genres, can be split and encoded separately.
* **Date Transformation**: Convert the *date\_added* column to a datetime format and extract useful features like the year and month added.
* **Feature Selection**: Selecting relevant features based on the analysis goal. For instance, if analysing the popularity of genres over time, focus on *listed\_in*, *date\_added*, and *release\_year.*

A screenshot of a computer program

AI-generated content may be incorrect.

**Fig .6** Filling or Remove Missing Values

**Data Exploration:** Identifying & treating the outliers in the column 'release\_year'

Outliers can impact statistical analyses and should be handled appropriately based on their cause (e.g., true outlier. To define the upper and lower bounds for outliers, you can use the interquartile range (IQR)

The IQR is the range between the 25th percentile (Q1) and the 75th percentile (Q3) of your data.

IQR = Q3 - Q1

Define the upper and lower bounds

Upper Bound: Q3 + (1.5 \* IQR)

Lower Bound: Q1 - (1.5 \* IQR)

 Fig 7a.

A screenshot of a computer screen

AI-generated content may be incorrect.  
Fig 7b.

Convert 'duration' column to numeric and showing the dataset has been transformed and structured with no null values.

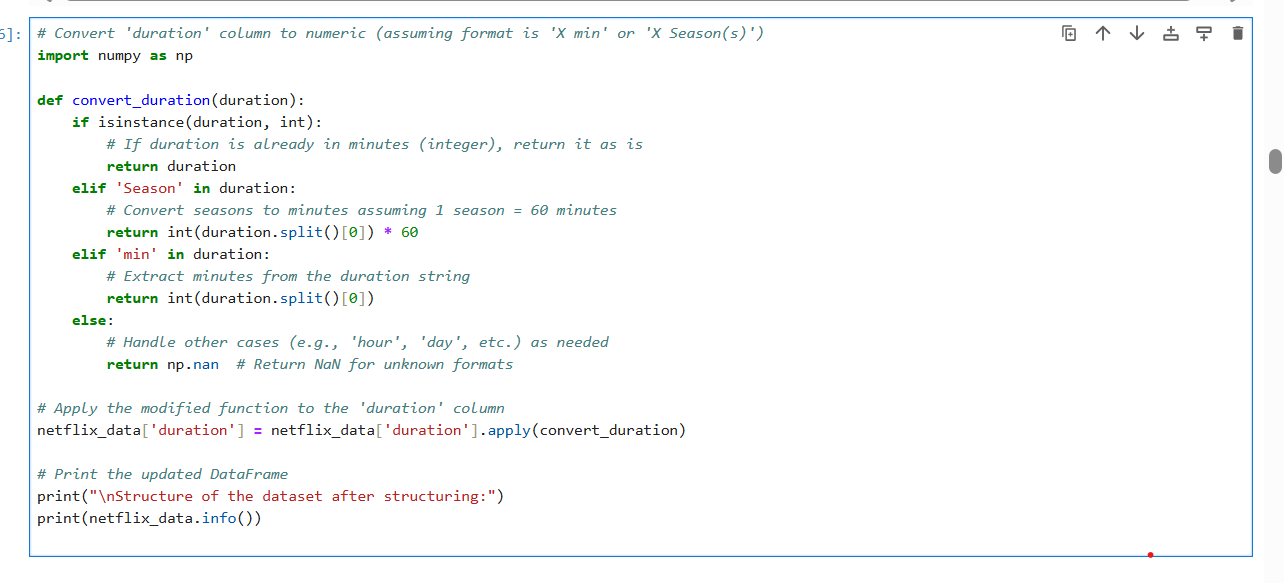


Fig 8. Convert duration column to numeric value.

**Data Reduction**

This involves selecting important features and reducing dimensionality using PCA.

The dataset focuses on the relevant features for subsequent analysis and can reduce dimensionality and improve the efficiency of your analysis.

Also, adding a new feature to classify content as 'Old' or 'New' based on release\_year and showing the dataset with the new feature.

* The goal is to add a new feature called 'content\_age' to the netflix\_data DataFrame.
* This feature will classify content as either ‘Old’ or ‘New’ based on the 'release\_year'
* Calculate the current year using pd.to\_datetime('now').year.

Apply a lambda function to the 'release\_year' column: If the release year is within the last 5 years, label it as ‘New’ otherwise, label it as ‘Old’.

****

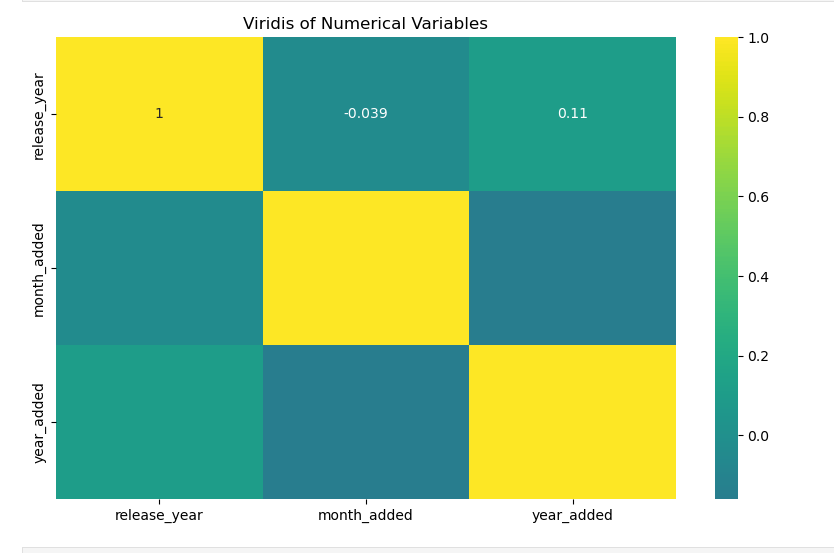
**Fig 9**

* As shown below, this is to analyze the correlation between relevant numerical columns in the netflix\_data dataset, such as columns: ('release\_year', 'month\_added', and 'year\_added').

The following procedures were used:

* A new DataFrame called netflix\_dt was created containing only the selected columns, then converting the 'release\_year' column to numeric format (if not already done).
* Calculate the correlation matrix between these numeric columns.
* Create a heatmap to visualize the correlations.
* The color scale (using the ‘viridis’ colormap) represents the strength of correlation (from negative to positive).

This helps identify any significant relationships between the release year and the added year (month) of content on Netflix.



**Fig.10** Correlation Between Relevant Numerical Columns.

The fig below shows the netflix\_data dataset containing information about Netflix shows or movies, using the sns.boxplot() function from the Seaborn library to create a box plot. The plot is a box plot that visualizes the relationship between the **rating** (e.g., TV-MA, TV-14, etc.) and the **release year** of Netflix content. The x-axis represents the different ratings, and the y-axis represents the release years and each box represents the distribution of release years for a specific rating category.

A screen shot of a graph

AI-generated content may be incorrect.

**Fig.11** Analysing the relationship between release year and rating.

The code below shows filtered the netflix\_data DataFrame to include only rows where the 'type' column has the value 'Movie’ and TV Shows. This creates a new DataFrame called ‘movie\_count’ & ‘tvscount’ containing information about Netflix movies.

The .head(2) method is used to display the first 2 rows of the movie\_count DataFrame.

Each row represents a movie, and the columns contain various attributes such as the movie’s title, release year, genre.

**A screenshot of a computer

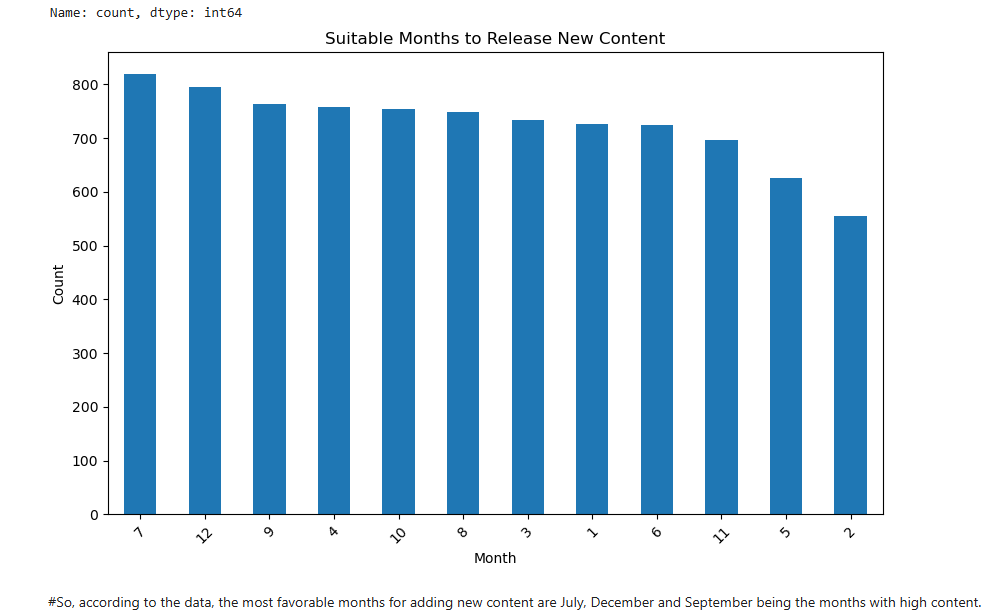
AI-generated content may be incorrect.**

**Fig.12** Creating a Dataframe each for movies and TV Shows.

* This method shown below calculates the number of content items (e.g., movies, TV shows) added in each month. new column called 'month\_added' by extracting the month from the 'date\_added' column using the .dt.month
* It uses the .value\_counts() method on the 'month\_added' column to get a count of how many times each month appears in the dataset.
* The table shows the count of content items (such as movies or TV shows) added to Netflix for each month which are represented by their numerical values (e.g., 1 for January, 2 for February, etc.).

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**Fig.13.** Suitable month to release new content.

July (month 7) has the highest number of content additions with 819 items. December (month 12) closely follows with 796 additions and September (month 9), while February (month 2) has the lowest count at 556 items.

Based on the data, the most suitable months for releasing new content on Netflix in terms of additions are: **July (month 7)**: Highest content additions and **December (month 12):** Close second

However, recommendation is based on historical data, and other factors (such as holidays, viewer interest.

## Diagnostic Statistics

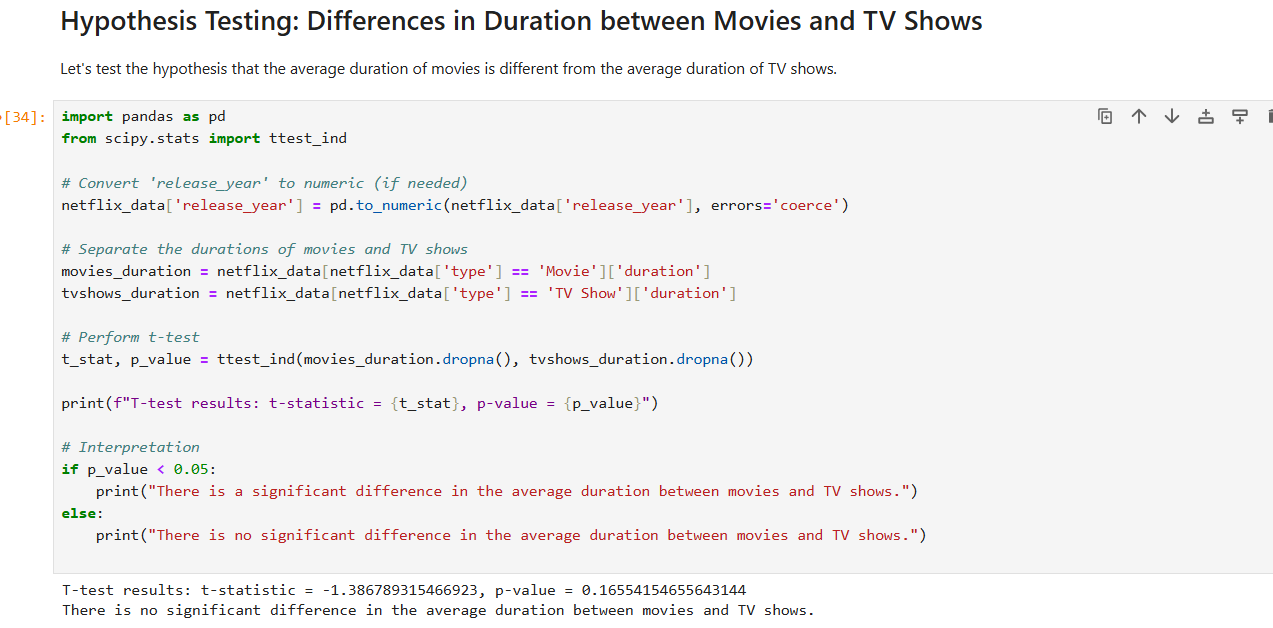
**Hypothesis Testing: Differences in Duration between Movies and TV Shows**

The above converts the 'release\_year' column in the netflix\_data DataFrame to a numeric data type. Using the pd.to\_numeric() function with the errors='coerce' argument, which means that any non-numeric values will be converted to NaN (missing values).

* movies\_duration: Contains the durations of movies (if the content type is 'Movie').
* tvshows\_duration: Contains the durations of TV shows (if the content type is 'TV Show').
* A t-test (independent two-sample t-test) is conducted between the durations of movies and TV shows. The t\_stat represents the t-statistic, and the p\_value represents the p-value from the t-test.

**Condition:** If the p-value is less than 0.05 (common significance level), we conclude that there is a significant difference in the average duration between movies and TV shows.

Otherwise, if the p-value is greater than or equal to 0.05, we conclude that there is no significant difference in average duration.



**Fig.14**.

Summary statistics for numeric and categorical columns provide insights into central tendencies and variations.

Data Visualization:

The analysis is to classify movies in the Netflix dataset is creating a new column duration\_binary which categorizes movies based on their duration. Movies with a duration such as "short" (0) if their duration is less than ( < )or equal to 90 minutes, or

"long" (1) if their duration is greater than(>) 90 minutes. A model is built to predict this binary target variable using features such as *release year* and *content rating.* The selected release\_year and rating as features (X) for the prediction model and duration\_binary as the target variable (y).

We used label encoding to convert the ratings into numeric form, which is required for most machine learning algorithms.

We split the data into training and testing sets with a 80-20 split. The random\_state is set to 8 to ensure reproducibility.

Conclusion:

This pre-processing workflow sets the stage for building and evaluating a predictive model for the binary classification of movie duration. By converting the categorical variable rating into numeric form and standardizing the features, we ensure that the model training process is efficient and effective. The training and testing split allows us to validate the performance of the model on unseen data.

A screenshot of a computer code

Description automatically generated

**Fig 15. Label Encoder**

A screenshot of a computer code

AI-generated content may be incorrect.

**Fig 16. Decision Tree Model**

 **Performance for Short Movies (0):**

* The model's **precision** for predicting short movies is 0.56, indicating that when the model predicts a movie as short, it is correct 56% of the time.
* The **recall** for short movies is 0.41, showing that the model correctly identifies 41% of the actual short movies.
* The **F1-score** is 0.47, which is a balance between precision and recall, indicating moderate performance.

 **Performance for Long Movies (1):**

* The model's **precision** for predicting long movies is 0.62, meaning it is correct 62% of the time when predicting a movie as long.
* The **recall** for long movies is 0.75, suggesting the model successfully identifies 75% of the actual long movies.
* The **F1-score** is 0.68, reflecting better performance in predicting long movies compared to short movies.

 **Overall Performance:**

* The model has an **overall accuracy** of 60.39%, indicating that it correctly classifies movies as short or long 60.39% of the time.
* The **macro average** and **weighted average** F1-scores are close to each other, around 0.58 and 0.59 respectively, indicating balanced performance across both classes despite slightly favoring long movies.

 **Confusion Matrix Analysis:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Fig 17. Logistic Regression**

* The model tends to predict long movies more accurately (370 true positives) compared to short movies (156 true negatives).
* There is a considerable number of false positives (224) and false negatives (121), suggesting that the model confuses short and long movies quite frequently.

Conclusion

The decision tree model demonstrates moderate performance with an overall accuracy of 60.39%. It performs better at predicting long movies than short movies, as reflected by higher precision, recall, and F1-scores for the long movie class. The confusion matrix indicates a tendency to misclassify movies, highlighting potential areas for model improvement. Further tuning of the model, feature engineering, or exploring other classification algorithms may help enhance the predictive performance.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Decision Tree** | **Logistic Regression** |
| **Precision (Class 0)** | 0.56 | 0.57 |
| **Precision (Class 1)** | 0.62 | 0.65 |
| **Recall (Class 0)** | 0.41 | 0.53 |
| **Recall (Class 1)** | 0.75 | 0.69 |
| **F1-Score (Class 0)** | 0.47 | 0.55 |
| **F1-Score (Class 1)** | 0.68 | 0.67 |
| **Accuracy** | 0.60 | 0.62 |
| **Confusion Matrix** | [[156, 224], [121, 370]] | [[200, 180], [153, 338]] |
| **Macro Avg Precision** | 0.59 | 0.61 |
| **Macro Avg Recall** | 0.58 | 0.61 |
| **Macro Avg F1-Score** | 0.58 | 0.61 |
| **Weighted Avg Precision** | 0.60 | 0.62 |
| **Weighted Avg Recall** | 0.60 | 0.62 |
| **Weighted Avg F1-Score** | 0.59 | 0.62 |
| **Accuracy Score** | 0.6039 | 0.6177 |

### Recommendation

Based on the comparison:

* **Decision Tree Model**:
  + Higher recall for class 1 (0.75) compared to Logistic Regression (0.69).
  + Lower recall for class 0 (0.41) compared to Logistic Regression (0.53).
  + Lower overall accuracy (0.60) compared to Logistic Regression (0.62).
* **Logistic Regression Model**:
  + Higher precision for class 1 (0.65) compared to Decision Tree (0.62).
  + Higher precision for class 0 (0.57) compared to Decision Tree (0.56).
  + Higher recall for class 0 (0.53) compared to Decision Tree (0.41).
  + Higher overall accuracy (0.62) compared to Decision Tree (0.60).

Considering the overall metrics, **the Logistic Regression model performs better** in terms of precision, recall, F1-score for both classes, and overall accuracy.

# References

Abhishek Jaiswal, 2024. *JSON: Introduction, Benefits, Applications, and Drawbacks.* [Online]   
Available at: https://www.turing.com/kb/what-is-json  
[Accessed 26th April 2024].

Aditi Prakash, August 18, 2023. *Parquet vs. Avro: A Detailed Comparison of Big Data File Formats.* [Online]   
Available at: https://airbyte.com/data-engineering-resources/parquet-vs-avro  
[Accessed 26 april 2024].

Ben Baranovsky, MAY 3, 2023. *CSV Files: Use cases, Benefits, and Limitations.* [Online]   
Available at: https://www.oneschema.co/blog/csv-files  
[Accessed 26 April 2024].

Cheyenne Kolosky, April 5, 2024. *The Ultimate Guide to CSV Formatting.* [Online]   
Available at: https://www.knack.com/blog/csv-formatting/  
[Accessed 26 April 2024].

Eran Levy, April 20, 2023. *What is the Parquet File Format? Use Cases & Benefits.* [Online]   
Available at: https://www.upsolver.com/blog/apache-parquet-why-use  
[Accessed 26th April 2024].

Jerry Franklin, october 26, 2022. *Parquet, ORC, and Avro: The File Format Fundamentals of Big Data.* [Online]   
Available at: https://www.upsolver.com/blog/the-file-format-fundamentals-of-big-data  
[Accessed 26 april 2024].

Kishan Yadav, 2022. *All About Big Data File Formats.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2022/05/all-about-big-data-file-formats/  
[Accessed 26 April 2024].

Scott Robinson, Alexander S. Gillis,, 2024. *DEFINITION 5V's of big data.* [Online]   
Available at: https://www.techtarget.com/searchdatamanagement/definition/5-Vs-of-big-data#:~:text=Volume%20is%20the%20amount%20of,the%20data's%20quality%20and%20accuracy.  
[Accessed 22 April 2024].

Yongjia Wang, Sep 15, 2017. *JSON and the Confusion of Formats in Big Data.* [Online]   
Available at: https://engineering.creditkarma.com/json-and-the-confusion-of-formats-in-big-data  
[Accessed 26 April 2024].