**A keyboard and a mouse

Description automatically generated**

***Python for Data Science***

***Final Project***

**Submitted To Professor Dr.Boxiang Dong**

***Submitted By Shivaprasad Dubyala***

2024

**Task 0 (0 pts). Load the netflix titles dataset. Use the magic command %sh and the Linux command wget to download the data from** [**https://msuweb.montclair.edu/~dongb/misc/netflix\_titles.csv**](https://msuweb.montclair.edu/~dongb/misc/netflix_titles.csv)

1. **Understand the Dataset**:

* The dataset contains information about movies and TV shows available on Netflix.
* [It includes details such as title, director, cast, country, release year, rating, show\_id, type, duration, listed\_in and description.](https://github.com/allenkong221/netflix-titles-dataset) .
* [There are over 8807 entries in the dataset.](https://github.com/allenkong221/netflix-titles-dataset)

1. **Download the Dataset**:

* We’ll use the wget command to download the dataset from the provided URL: Netflix Titles Dataset.
* Open a terminal or shell (you can use %sh in Jupyter Notebook or a regular terminal).
* Execute the following command to download the dataset:
* wget <https://msuweb.montclair.edu/~dongb/misc/netflix_titles.csv>

1. **Verify the Download**:

* A screenshot of a computer

  Description automatically generatedAfter running the command, check if the file netflix\_titles.csv has been successfully downloaded

1. **Load the Dataset into Python (using Pandas)**:
   * Once the dataset is downloaded, you can load it into Python using the Pandas library.
   * If you’re working in a Jupyter Notebook, you can use the following code snippet:
   * import pandas as pd
   * # Load the dataset
   * df = pd.read\_csv("netflix\_titles.csv")
   * # Explore the data (e.g., display the first few rows)
   * df.head()

**A screenshot of a computer

Description automatically generatedTask 1 (10 pts). Data Cleaning. The duration column describes the length of the movie/show. The rule is as follows: if it is a movie, the duration is described in the number of minutes; if it is a TV show, it is described in the number of seasons. Find the records that do not follow the rule and fix the error.**

Certainly! Let’s break down the provided code snippet step by step:

1. **Filtering Incorrect Records:**

* The code creates a DataFrame called incorrect\_records by filtering the original netflix\_titles\_df based on specific conditions.

**The conditions are as follows:**

* For movies (type == 'Movie'):
* If the “duration” column contains the word “season” (case-insensitive), it is considered incorrect.
* For TV shows (type == 'TV Show'):
* If the “duration” column contains the word “min” (case-insensitive), it is considered incorrect.
* Additionally, any records with missing values in the “duration” column are also considered incorrect.

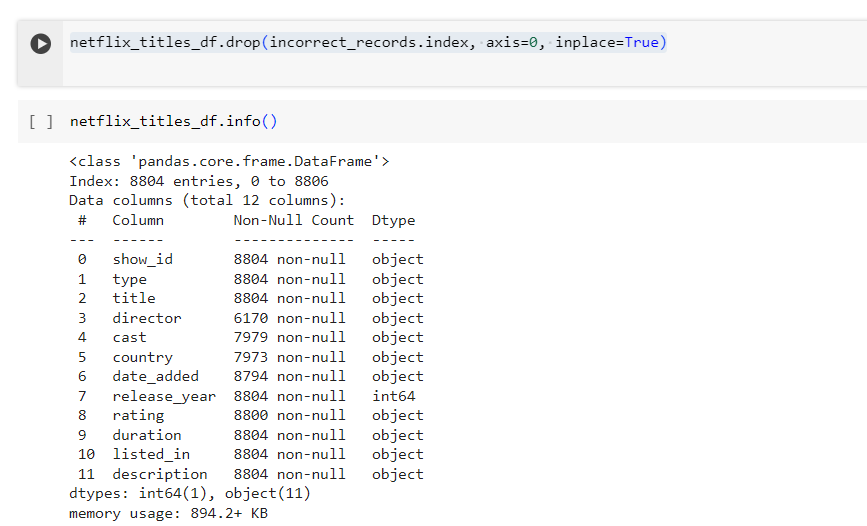
1. **Columns in incorrect\_records:**

* The incorrect\_records DataFrame includes the following columns:
* show\_id: Unique identifier for each title.
* type: Indicates whether it’s a movie or a TV show.
* title: Title of the movie or TV show.
* duration: The duration (either in minutes for movies or in seasons for TV shows).

1. **Printing Incorrect Records:**

* The code prints the incorrect records along with their relevant information (show ID, type, title, and duration).

**A screenshot of a computer

Description automatically generated **

**Task 2 (20 pts). Data Transformation**

**Task 2.1 (10 pts). The country column includes a string that lists the countries where the movie/show was produced. In case of multiple countries, they are concatenated with commas. Replace this column with single country.**

1. **Splitting Countries into a List**:

* The first line of code splits the values in the “country” column of the netflix\_titles\_df DataFrame.
* Specifically, it uses the .str.split(', ') method to split the comma-separated countries into a list.
* For example, if the original “country” value was "United States, Canada, United Kingdom", it will become ["United States", "Canada", "United Kingdom"].

1. **Exploding the DataFrame**:

* The second line of code applies the .explode('country') method to the DataFrame.
* This operation “explodes” the list of countries into separate rows, creating a new row for each country.
* The index of the original row is preserved, and the other columns (such as title, type, etc.) are duplicated for each country.
* For example, if a movie was originally associated with three countries, it will now appear as three separate rows, each with one of the countries.

1. **Resulting DataFrame**:

* After applying these operations, the netflix\_titles\_df DataFrame will have additional rows corresponding to each country associated with a title.
* The “country” column will now contain individual country names instead of lists.

**A white background with many small text

Description automatically generated with medium confidence**

**Task 2.2 (10 pts). The listed in column stores the movie/show categories. Similar to the country column, it may store multiple categories that are concatenated by comma. Similar to Task 1.1, create a new column named genre that stores a single category in each row.**

1. **Splitting Genres into a List**:

* The first line of code splits the values in the “listed\_in” column of the netflix\_titles\_df DataFrame.
* Specifically, it uses the .str.split(', ') method to split the comma-separated genres into a list.
* For example, if the original “listed\_in” value was "Action, Adventure, Sci-Fi", it will become ["Action", "Adventure", "Sci-Fi"].

1. **Exploding the DataFrame**:

* The second line of code applies the .explode('genre') method to the DataFrame.
* This operation “explodes” the list of genres into separate rows, creating a new row for each genre.
* The index of the original row is preserved, and the other columns (such as title, type, etc.) are duplicated for each genre.
* For example, if a movie was originally associated with three genres, it will now appear as three separate rows, each with one of the genres.

1. **Resulting DataFrame**:

* After applying these operations, the netflix\_titles\_df DataFrame will have additional rows corresponding to each genre associated with a title.
* The “genre” column will now contain individual genre names instead of lists.

A screenshot of a computer

Description automatically generated

**Task 3 (70 pts). Data Aggregation & Visualization.**

**Task 3.1 (10 pts). Count the total number of movies/shows by release year and draw a lineplot to show the number of movies/shows since 2000.**

netflix\_titles\_df.groupby('release\_year')['title'].nunique() step by step:

1. **netflix\_titles\_df**:

* This refers to the DataFrame containing information about Netflix titles (movies, TV shows, documentaries, etc.).

1. **.groupby('release\_year')**:

* The .groupby() method is used to group the data based on a specific column.
* In this case, we are grouping the data by the “release\_year” column. This means that we’ll create groups where titles have the same release year.

1. **['title']**:

* After grouping, we specify the column we want to analyze within each group.
* Here, we’re interested in the “title” column.

1. **.nunique()**:

* The .nunique() function calculates the number of unique values within each group.
* It counts the distinct titles (movies or TV shows) for each release year.

A screen shot of a computer

Description automatically generated

**Ploting the Line plot :**

1. **Filtering Recent Movies and Shows**:

* The first line of code creates a new DataFrame called recent\_movies\_shows\_df.
* It filters the original netflix\_titles\_df to include only titles released in or after the year 2000 (release\_year >= 2000).

1. **Grouping by Release Year**:

* The second line groups the filtered DataFrame (recent\_movies\_shows\_df) by the “release\_year” column.
* Within each group, it calculates the number of unique titles (movies or TV shows) using .nunique().

1. **Creating a Line Plot**:

* The subsequent lines of code create a line plot to visualize the trend in the number of movies and shows released since 2000.
* Here’s what each line does:
* plt.figure(figsize=(15, 10)): Sets the figure size for the plot.
* movies\_shows\_count.plot(kind='line', marker='o', color='b'): Plots the data as a line graph with circular markers (blue color).
* plt.title('Number of Movies/Shows Since 2000'): Sets the plot title.
* plt.xlabel('Release Year'): Labels the x-axis.
* plt.ylabel('Number of Movies/Shows'): Labels the y-axis.
* plt.grid(True): Adds gridlines to the plot.
* plt.xticks(range(2000, max(recent\_movies\_shows\_df['release\_year']) + 1, 1)): Sets the x-axis ticks (years from 2000 to the maximum release year).
* plt.tight\_layout(): Adjusts the layout for better spacing.
* plt.show(): Displays the plot.

1. **Interpreting the Plot**:

* The resulting plot shows how the number of movies and TV shows released on Netflix has changed over time since the year 2000.
* You can observe trends, spikes, or declines in content production based on the line graph.



A graph with a line going up

Description automatically generated

**Task 3.2 (10 pts). Find the top-30 productive directors. The productivity of a director is measured by the number of movies/shows. Visualize the top-30 productive directors and their number of movies/shows with a barplot.**

Certainly! Let’s break down the provided code snippet step by step:

1. **Grouping by Director**:
   * The first line of code groups the netflix\_titles\_df DataFrame by the “director” column using the .groupby('director') method.
   * This means that titles (movies or TV shows) are grouped together based on their respective directors.
2. **Calculating Unique Titles**:
   * Within each group (i.e., for each director), the .nunique() function is applied to the “title” column.
   * The .nunique() function counts the number of unique titles associated with each director.
   * For example, if a director has worked on multiple titles, this count represents the distinct titles they directed.
3. **Sorting the Results**:
   * The director\_counts Series (resulting from the .nunique() operation) is sorted in descending order using .sort\_values(ascending=False, inplace=True).
   * This ensures that the directors with the highest number of unique titles appear at the top.
4. **Resulting Series**:
   * The director\_counts Series now contains the count of unique titles for each director.
   * It provides insights into which directors have contributed the most content to Netflix.

**A computer screen shot of a computer

Description automatically generated**

**Creating a bar chart showing the top 30 productive directors based on the number of movies and shows they’ve directed:**

1. **Grouping by Director**:

* The first line of code groups the netflix\_titles\_df DataFrame by the “director” column using the .groupby('director') method.
* This means that titles (movies or TV shows) are grouped together based on their respective directors.

1. **Calculating Unique Titles**:

* Within each group (i.e., for each director), the .nunique() function is applied to the “title” column.
* The .nunique() function counts the number of unique titles associated with each director.
* For example, if a director has worked on multiple titles, this count represents the distinct titles they directed.

1. **Sorting the Results**:

* The director\_counts Series (resulting from the .nunique() operation) is sorted in descending order using .sort\_values(ascending=False, inplace=True).
* This ensures that the directors with the highest number of unique titles appear at the top.

1. **Selecting the Top 30 Directors**:

* The next line of code selects the top 30 directors based on the sorted counts.
* These are the most productive directors in terms of the number of titles they’ve directed.

1. **Creating a Bar Chart**:

* The subsequent lines of code create a bar chart to visualize the data.
* Here’s what each line does:
  + - plt.figure(figsize=(15, 10)): Sets the figure size for the plot.
    - top\_30\_directors.plot(kind='bar', color='orange'): Plots the data as a bar chart with orange bars.
    - plt.title('Top 30 Productive Directors'): Sets the plot title.
    - plt.xlabel('Director'): Labels the x-axis.
    - plt.ylabel('Number of Movies/Shows'): Labels the y-axis.
    - plt.xticks(rotation=90): Rotates the x-axis labels for better readability.
    - plt.tight\_layout(): Adjusts the layout for better spacing.
    - plt.show(): Displays the plot.

1. **Interpreting the Plot**:

* The resulting bar chart shows the top 30 directors and the number of unique titles they’ve directed.
* You can easily compare their productivity and identify the most prolific directors.

**A screen shot of a computer code

Description automatically generated**

A graph of a number of orange bars

Description automatically generated with medium confidence

**Task 3.3 (10 pts). Make a lineplot that shows the average movie length and 95% confidence interval for every year since 2000. (x-axis is the year, y-axis is the length in minutes).**

1. **Filtering Recent Movies**:

* The first line of code creates a new DataFrame called recent\_movies\_df.
* It filters the original netflix\_titles\_df to include only titles of type “Movie” released in or after the year 2000 (release\_year >= 2000).

1. **Calculating Average Movie Length per Year**:

* The second line groups the filtered DataFrame (recent\_movies\_df) by the “release\_year” column.
* Within each group, it calculates the average movie length (duration) using the .apply(lambda x: pd.to\_numeric(x.str.extract('(\d+)')[0]).mean()) operation.
* Here’s what happens:
* x.str.extract('(\d+)')[0] extracts the numeric part (minutes) from the “duration” column.
* pd.to\_numeric(...) converts the extracted values to numeric format.
* .mean() calculates the average of these numeric values.

1. **Calculating Confidence Intervals**:

* The third line calculates the confidence intervals for the average movie length.
* It uses the standard error of the mean (SEM) multiplied by 1.96 (which corresponds to a 95% confidence level).
* The SEM is calculated as the standard deviation divided by the square root of the sample size.

1. **Creating the Line Plot with Confidence Interval**:

* The subsequent lines of code create a line plot to visualize the average movie length over time.
* Here’s what each line does:
  + - plt.figure(figsize=(15, 10)): Sets the figure size for the plot.
    - sns.lineplot(data=average\_length\_per\_year, marker='d', color='b'): Plots the average movie length as a line graph with diamond markers (blue color).
    - plt.fill\_between(...): Fills the area between the upper and lower confidence intervals with a pink color (alpha = 0.5 for transparency).
    - plt.title('Average Movie Length and 95% Confidence Interval since 2000'): Sets the plot title.
    - plt.xlabel('Release Year'): Labels the x-axis.
    - plt.ylabel('Average Movie Length (minutes)'): Labels the y-axis.
    - plt.grid(True): Adds gridlines to the plot.
    - plt.xticks(range(2000, max(recent\_movies\_df['release\_year']) + 1, 2)): Sets the x-axis ticks (years from 2000 to the maximum release year, with a step of 2).
    - plt.tight\_layout(): Adjusts the layout for better spacing.
    - plt.show(): Displays the plot.

1. **Interpreting the Plot**:

* The resulting plot shows the average movie length over the years since 2000.
* The shaded pink area represents the 95% confidence interval around the average.
* You can observe trends or fluctuations in movie length based on the line graph.

A close-up of a computer screen

Description automatically generated

A graph with a line going up

Description automatically generated

**Task 3.4 (20 pts). Make a lineplot that shows the number of movies/shows produced every year since 2000 in each of the following countries (United States, India, United Kingdom, Japan, South Korea) respectively. (x-axis is the year, y-axis is the number of products, and each country has a line). Make sure that you add the legend.**

1. **Filtering Data**:

* The first line of code creates a new DataFrame called filtered\_df.
* It filters the original netflix\_titles\_df to include only titles released in or after the year 2000 (release\_year >= 2000) and from specific countries (United States, India, United Kingdom, Japan, and South Korea).

1. **Grouping by Release Year and Country**:

* The second line groups the filtered DataFrame (filtered\_df) by both the “release\_year” and “country” columns.
* Within each group, it calculates the number of unique titles (movies or TV shows) using .nunique().unstack(fill\_value=0):
  + - .nunique() counts the unique titles for each combination of release year and country.
    - .unstack(fill\_value=0) reshapes the data, converting the country names into separate columns.

1. **Creating the Line Plot**:

* The subsequent lines of code create a line plot to visualize the number of movies and shows produced by each country over time.
* Here’s what each line does:
  + - plt.figure(figsize=(15, 10)): Sets the figure size for the plot.
    - for country in countries:: Iterates through the specified countries.
    - plt.plot(...): Plots the data for each country as a line graph.
    - plt.title('Number of Movies/Shows Produced Since 2000'): Sets the plot title.
    - plt.xlabel('Release Year'): Labels the x-axis.
    - plt.ylabel('Number of Movies/Shows'): Labels the y-axis.
    - plt.legend(title='Country', loc='upper left'): Adds a legend with country names.
    - plt.grid(True): Adds gridlines to the plot.
    - plt.xticks(range(2000, max(filtered\_df['release\_year']) + 1, 2)): Sets the x-axis ticks (years from 2000 to the maximum release year, with a step of 2).
    - plt.tight\_layout(): Adjusts the layout for better spacing.
    - plt.show(): Displays the plot.

1. **Interpreting the Plot**:

* The resulting plot shows how the number of movies and TV shows produced by each of the specified countries has changed over time since 2000.
* You can observe trends, variations, and relative contributions of each country.

A computer screen shot of text

Description automatically generated

A graph of a number of people

Description automatically generated

**Task 3.5 (20 pts). Find all the countries that produced more than 50 movies/shows in history. Make a mapplot where each country is plotted on the map as a circle and annotated. The circle size is based on the number of movies/shows it produced.**

1. **Data Preparation**:
   * The code begins by calculating the number of unique titles (movies or shows) produced in each country. This is done using the groupby('country')['title'].nunique() operation.
   * The resulting country\_counts Series contains the count of unique titles for each country.
2. **Filtering Countries**:
   * The next step identifies countries that have produced more than 50 movies or shows. These countries are stored in the countries\_over\_50 list.
3. **World Map Data**:
   * The code loads a world map dataset using GeoPandas. Specifically, it uses the naturalearth\_lowres dataset, which provides low-resolution geometries for countries and their boundaries.
   * The world\_map DataFrame contains information about country shapes and geometries.
4. **Calculating Area**:
   * An additional column called “area” is created in the world\_map DataFrame. This column represents the area of each country’s geometry (polygon) on the map.
5. **Creating the Plot**:
   * The code sets up a figure with a size of 15x10 using plt.figure(figsize=(15, 10)).
   * It plots the world map using world\_map.plot(ax=plt.gca(), color='pink', edgecolor='black'). The ax=plt.gca() ensures that the plot is drawn on the current axis.
   * For each country in the countries\_over\_50 list:
     + Annotates the country name at the centroid of its geometry using plt.annotate(...).
     + Places a yellow scatter point at the centroid using plt.scatter(...). The size of the scatter point is proportional to the country’s area (multiplied by 5).
     + The alpha=0.5 parameter controls the transparency of the scatter points.
6. **Plot Customization**:
   * The title, x-axis label, and y-axis label are set using plt.title(...), plt.xlabel(...), and plt.ylabel(...).
   * The x-axis represents longitude, and the y-axis represents latitude.
   * The layout is adjusted using plt.tight\_layout().
7. **Interpreting the Plot**:
   * The resulting plot shows a world map with countries that have produced more than 50 movies or shows highlighted.
   * Each country is represented by a yellow scatter point at its centroid.
   * The size of the scatter points reflects the relative area of each country.

**A computer screen with colorful text

Description automatically generated**

A pink and yellow map with dots

Description automatically generated