

Contents

[GitHub URL 2](#_heading=h.gjdgxs)

[Abstract 2](#_heading=h.30j0zll)

[Introduction 2](#_heading=h.1fob9te)

[Dataset 2](#_heading=h.3znysh7)

[Justification: 2](#_heading=h.2et92p0)

[Source: 2](#_heading=h.tyjcwt)

[Implementation Process 3](#_heading=h.3dy6vkm)

[Results 7](#_heading=h.1t3h5sf)

[Insights 9](#_heading=h.4d34og8)

[References 12](#_heading=h.2s8eyo1)

# GitHub URL

https://github.com/conortcahalane/UCDPythonProject2023

# Abstract

For this project, I chose to do Python analysis on IMDB (An internet movie database website) and Netflix.

The purpose of the application was to develop a Python project to analyse a real-world scenario and generate valuable insights by visualising information. The project aims to analyse data from different data sources, manipulate information and visualise to generate insights.

Using Jupyter notebook as my platform and a host of useful resources such as packages NumPy, Sklearn and Pandas, Visualisation tools from Seaborn and Plotly, I was able to develop and implement the functionality required for the project.

# Introduction

As described in paper information provided by University College Dublin, the goal of the assignment is to demonstrate how you are thinking about putting course concepts and learning into practice to demonstrate the course learning outcomes.

Prior to these UCD courses, I had minimal previous experience with the programming language Python or Data Analysis tools such as data visualisation. Due to this I knew a substantial portion of time during this project would be dedicated to investigating, understanding, and testing elements of functionality, as well as figuring out different implementations to try better understanding how the code was operating and what I thought would be the best solution to each bullet point set out by UCD.

Due to this for my use case I chose IMDB Movies Dataset and Netflix Movies and TV Shows Dataset. I chose these datasets as I knew there would be so many new concepts and new functionality to understand that my dataset should be from a topic that I am extremely familiar with. Using this dataset allowed me to focus on the coding and functionality aspects of the project more and did not impede me in having to wrap my head around aspects of the dataset due to not understanding the content.

# Dataset

## Justification:

Both of these datasets stuck out to me on Kaggle due to my love of film and TV series. Since I know the topic very well, more insightful analysis can be gained from it. I would have better understanding to manipulate the data and clearer goals for what I wanted to achieve with my analysis. If I was to choose Badminton instead of Movies and TV shows, then many interesting data points might’ve been missed as I wouldn’t be as sure whether the data was insightful or mundane.

The dataset is the Top 1000 Movies by IMDB Rating and is comprised of 16 columns with the headers:

* **Poster\_Link** - Link of the poster that imdb using
* **Series\_Title** = Name of the movie
* **Released\_Year** - Year at which that movie released
* **Certificate** - Certificate earned by that movie
* **Runtime** - Total runtime of the movie
* **Genre** - Genre of the movie
* **IMDB\_Rating** - Rating of the movie at IMDB site
* **Overview** - mini story/ summary
* **Meta\_score** - Score earned by the movie
* **Director** - Name of the Director
* **Star1,Star2,Star3,Star4** - Name of the Stars
* **No\_of\_votes** - Total number of votes
* **Gross**

The dataset consists of 1000 rows.

Following, I supplemented this with the Netflix Movies and TV shows dataset which contains all information regarding movies/Tv netflix data.

## Source:

<https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows>

<https://www.kaggle.com/datasets/shivamb/netflix-shows>

# Implementation Process

## 2. Importing

First things foremost, I import all of the packages to be used in the project and create a class called colour to use to present our results. I use Pandas to get our IMDB dataset into a DataFrame. I use the function .read\_csv() to make a DataFrame of our csv.

I also import a Netflix movies and tv shows database. I create a relational database on **ln 22** where I find the common values in the 'title' & 'Series\_Title' column, filter the DataFrames based on the common values and then perform a merge based on the common values. This allows us to have a dataframe of the IMDB’s top 1000 movies that exist within Netflix, giving us additional columns to work with. This comes with the caveat that only 171 titles on the IMDB top 1000 are available within Netflix. Less scope but more in depth data. This being an example of the project where merging DataFrames took place.

## 3. Exploring

### Sorting

Toying around with the data I use regex functionality to pull specific data. This can be seen on **ln 26**.

Exploring the data further I use iterations in order to present the Title, Release Year, Genre, Director and its IMDB score. I also make use of the colour class to improve its readability. Using iterators like this can present the data in the DataFrame in a readable fashion and allows the user to present this without having to use the print() function repeatedly. This can be seen in the project on **ln 27**.

### Python

Playing around with some python fundamentals in the project I decided to use NumPy to display different interesting data points that can be found within our merged netflix/IMDB dataframe. In **ln 25** Iextract the 'Rating' column as a NumPy array. I calculate the average rating using NumPy, I calculate the maximum rating using NumPy, count the number of movies using NumPy, I create a boolean mask for movies with rating above a threshold using NumPy and finally I select and print only four columns from the filtered dataframe. This displays the only four movies in our merged dataframe that meet our NumPy thresholds demand.

On **ln 28** we can see a dictionary being used in the project using the function to\_dict(). This is used to create a dictionary of all the records within the merged DataFrame. This is useful for storing information in key value pairs which is appropriate for our dataset.

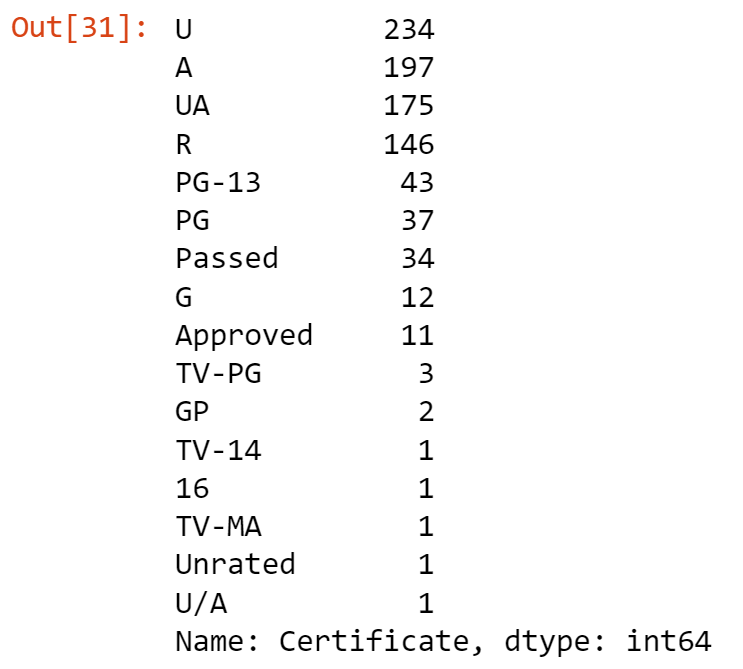
## 4. Data Preparation and Processing

I decided for the machine learning portion of the project that it would be better to just focus on a singular DataFrame with this being the IMDB dataset. This way I could analyse the dataset in detail, identify areas to clean, simplify and improve the data in order to prepare it for our eventual machine learning algorithm.

I run the .info() function on the code to see if I can find anything. I notice that the release year, which should be a float, is returning an object. I investigate and find that on row 966 the release year is valued as “PG”. I googled the movie and remedy this to its rightful release year of “1995” to fix the column.

Moving on to runtime I remove the “min” that is appended to each value. Thus turning each one from a string to an integer. This will better feed into my algorithm.

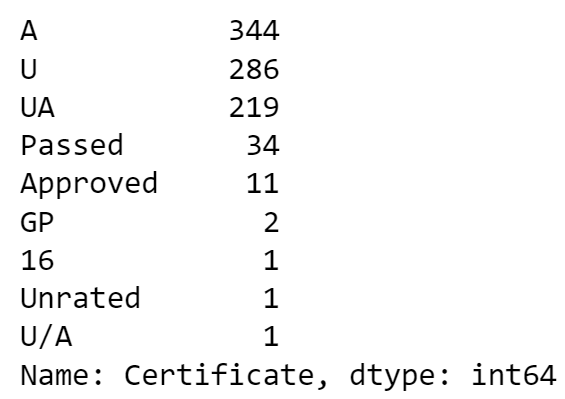
Starting on **ln 31** this with the Certification column. Using the value count function I find the certificate column has errors. As seen below:



We are going to fix the Certificates column to be categorised into 3 values.

**U** - PG, G… **UA** - PG-13… **A** - NC-17,R…

In **ln 39** I create a series of for loops that will categorise the certificate ratings into our 3 bands in the array certificate\_mapping{}



With still some outstanding values not in our three designated buckets we deal with these in **ln 40**.

After this remains 102 blank values, To fill these had to find a reasoning, my reasoning ended up being genres. My assumption is that films categorised as War, Horror or crime for example could be commonly categorised as mature (A), whilst movie genres such as Family, History or music could be categorised as universal (U) typically.

With this assumption in mind I checked how many individual genres existed in the dataset. Then checked the frequency at which each was bucketed into my 3 bands. Thankfully each of my above 6 assumptions turned out correct so I decided to fill the blank certificate using their genres.

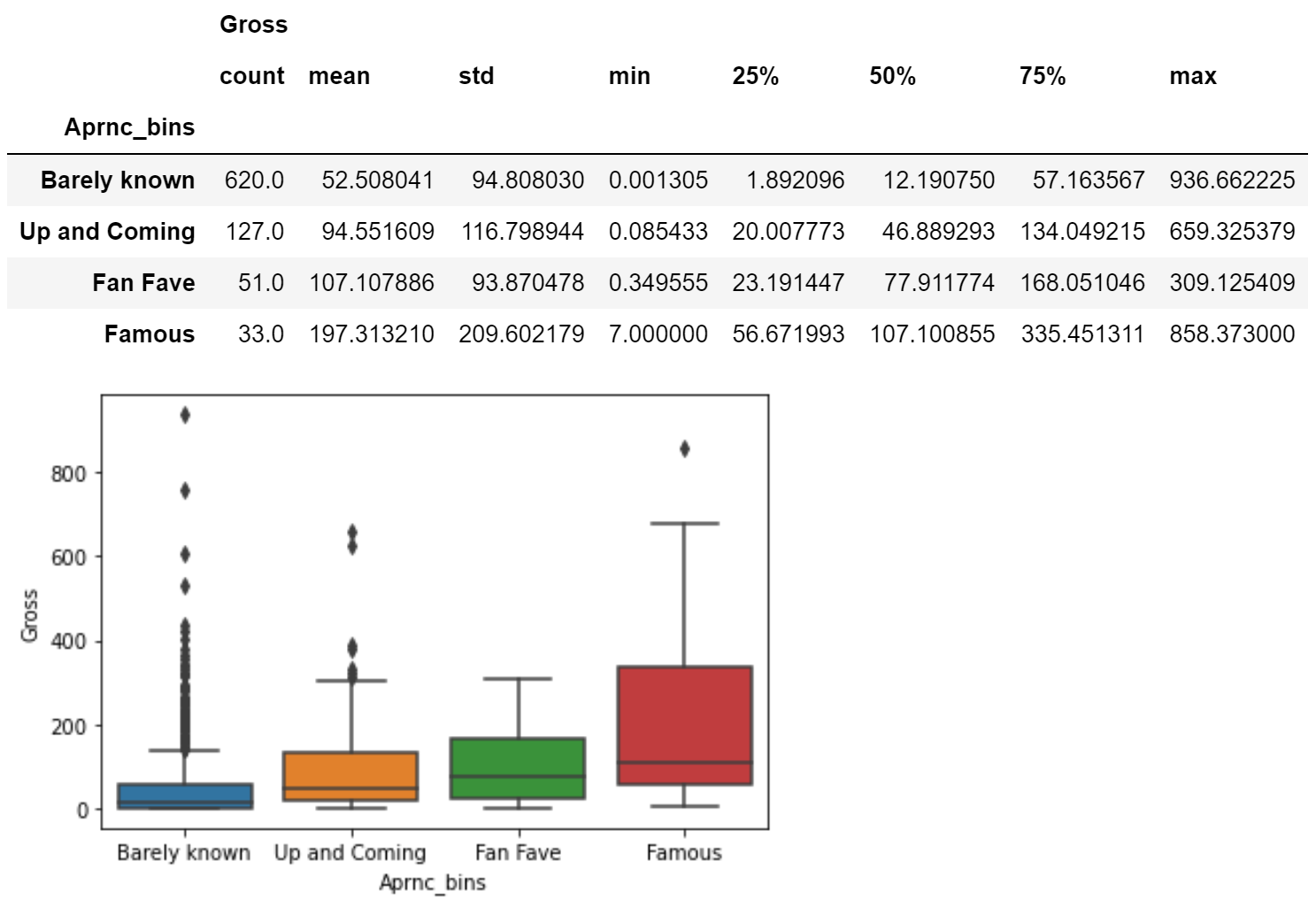
Finally I convert the certificate to numeric values. certificate\_mapping = {'A': 0, 'UA': 1, 'U': 2}. This finished off my data preparation of the certificate column. **Ln 46**

I simplified a couple columns such as putting “Gross” into millions and “No\_of\_Votes” into thousands. **Ln 45**

To simplify further I amalgamate the star columns into one singular cast column. It's important to know that IMDB sorts the stars by who is credited first in the end credits, therefore it isn't determined by popularity but importance to the film. To create this attribute, we are going to use the following logic: the higher the class cast of actors composing a starring group, the more likely a movie will attract a large audience and generate large revenues. The first instance of an actor appearing in the dataset is set to be 0. It will increment by 1 each subsequent appearance in chronological order, if any.

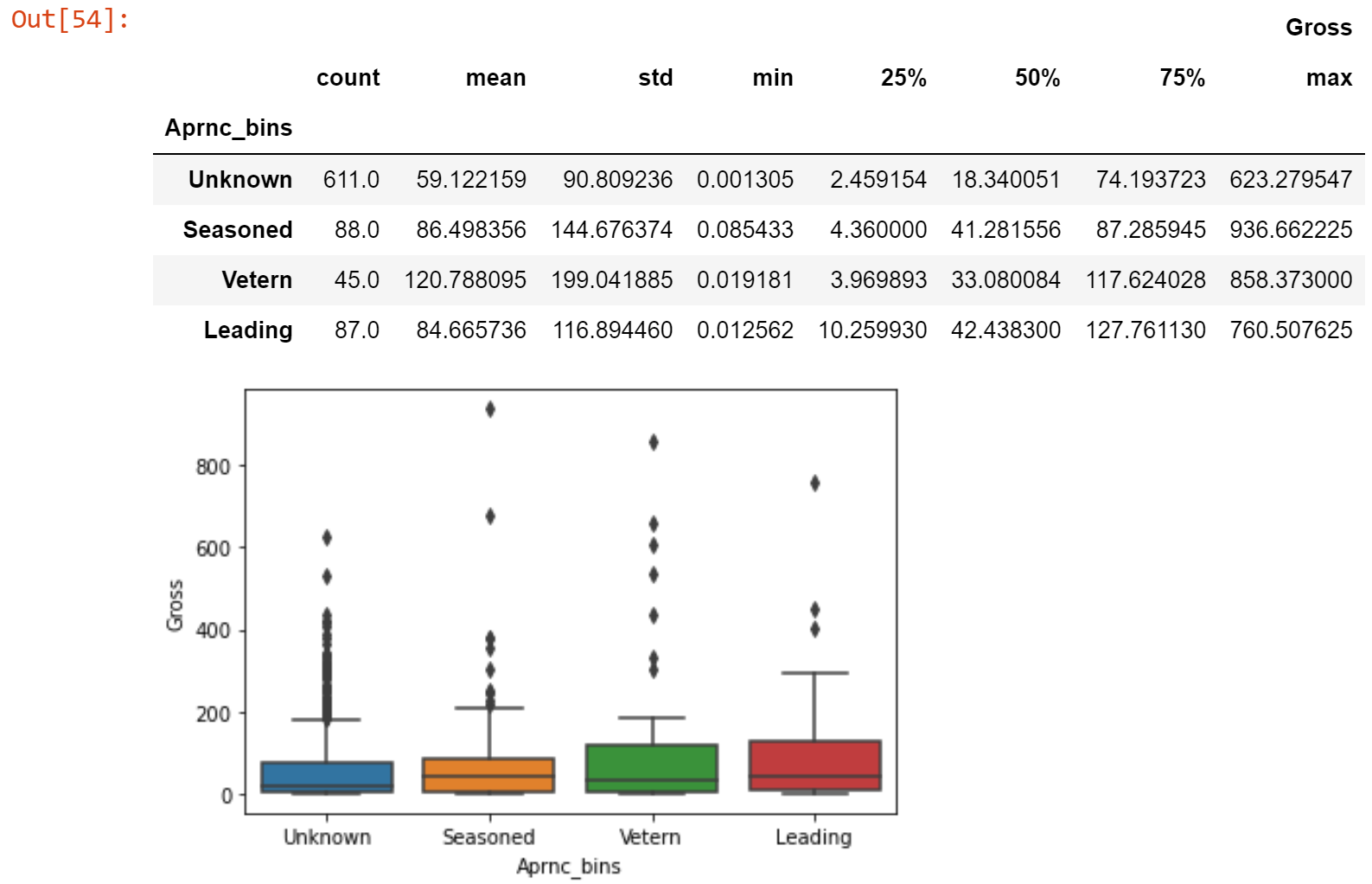
I created a custom feature called Act\_aprnc (actor appearance amount), which will reflect the sum of appearances of all four actors by series.

I then created 4 appearance bands to check if it correlated with the movies “Gross”. Which it definitely did.



In conclusion,movies with less known actors tend to gather less cash than those with a famous cast.

I took the above and applied a similar methodology to the director column. This time creating the bands 'Unknown', 'Seasoned', 'Vetern', 'Leading' to differentiate from the level of director.



This followed a very similar trend to the actors, showing that the more prestigious a director leading a film, the more gross revenue it had a chance of generating.

Finally before the algorithm, I drop any excess columns that wouldn't have made sense. I also also drop any missing info for gross.

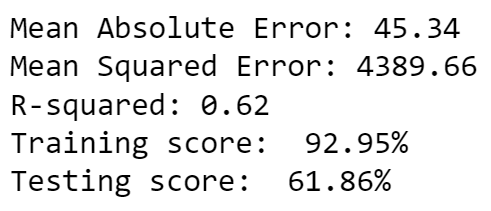
## 5. Machine Learning

Lastly, I train a model and predict the gross revenue. Since the Gross feature is a continuous variable, models selection points on those, capable of performing a regression task. I choose to use random decision forests to identify relationships between our output and the columns we created above.

I use sklearn to split my dataset into training and testing datasets. The train\_test\_split() function splits arrays or matrices into random train and test subsets.

It’s a quick utility that wraps input validation, next(ShuffleSplit().split(X, y)), and application to input data into a single call for splitting (and optionally subsampling) data into a one-liner.

After feeding my train and test datasets into my RandomForestRegressor model and print the metrics returned I get the below results:



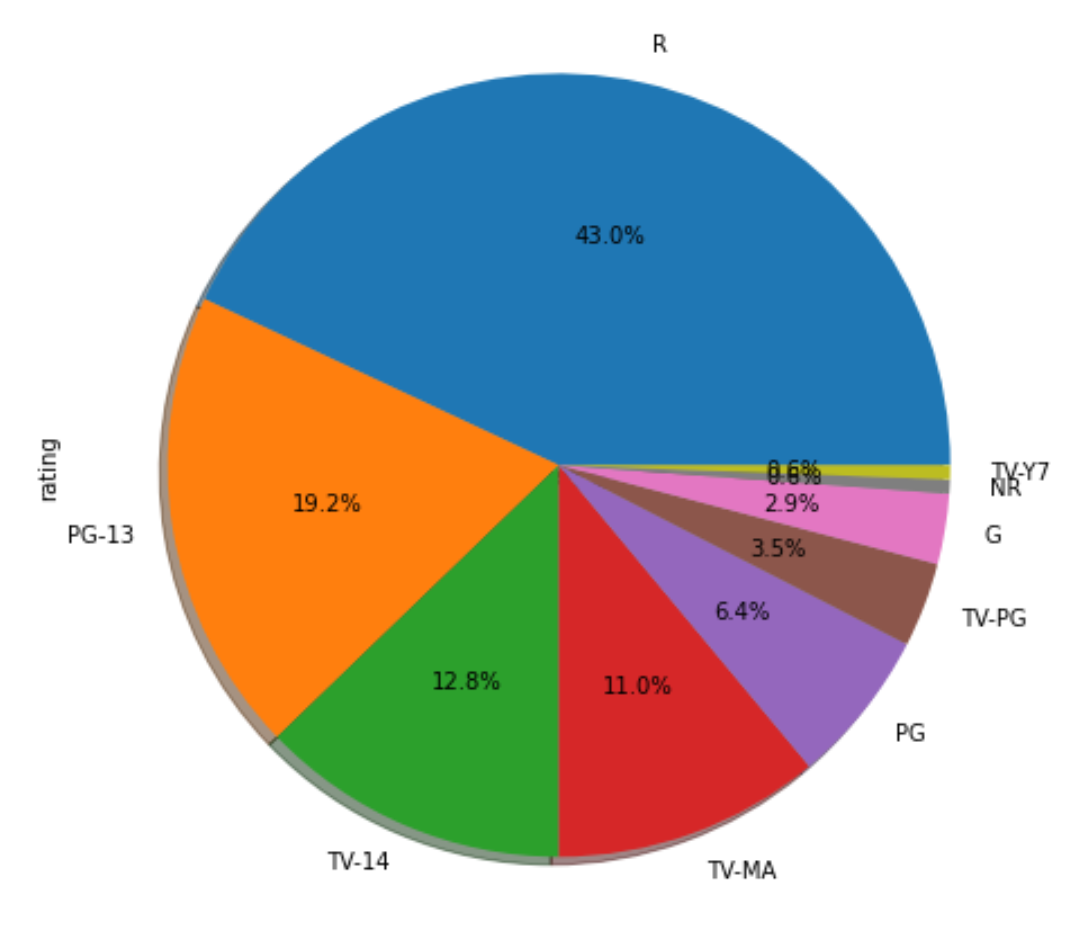
The training score performing much better than the testing score is interesting and could be seen as overfitting.

# Results

The results of the sklearn machine learning algorithm can be seen in the below two charts.

## 

This chart was created using Seaborn and Matplotlib. The bar chart shows the relation between each column fed into the algorithm against Gross. This shows you which attribute most strongly influences the gross revenue of a film.



The above graph on the other hand shows a Matplotlib pie chart clearly showing the percentages of different ratings which can be seen amongst the movies from IMDB’s top 1000 that are also available on Netflix.

# 

# 

# 

# 

# 

# 

# 

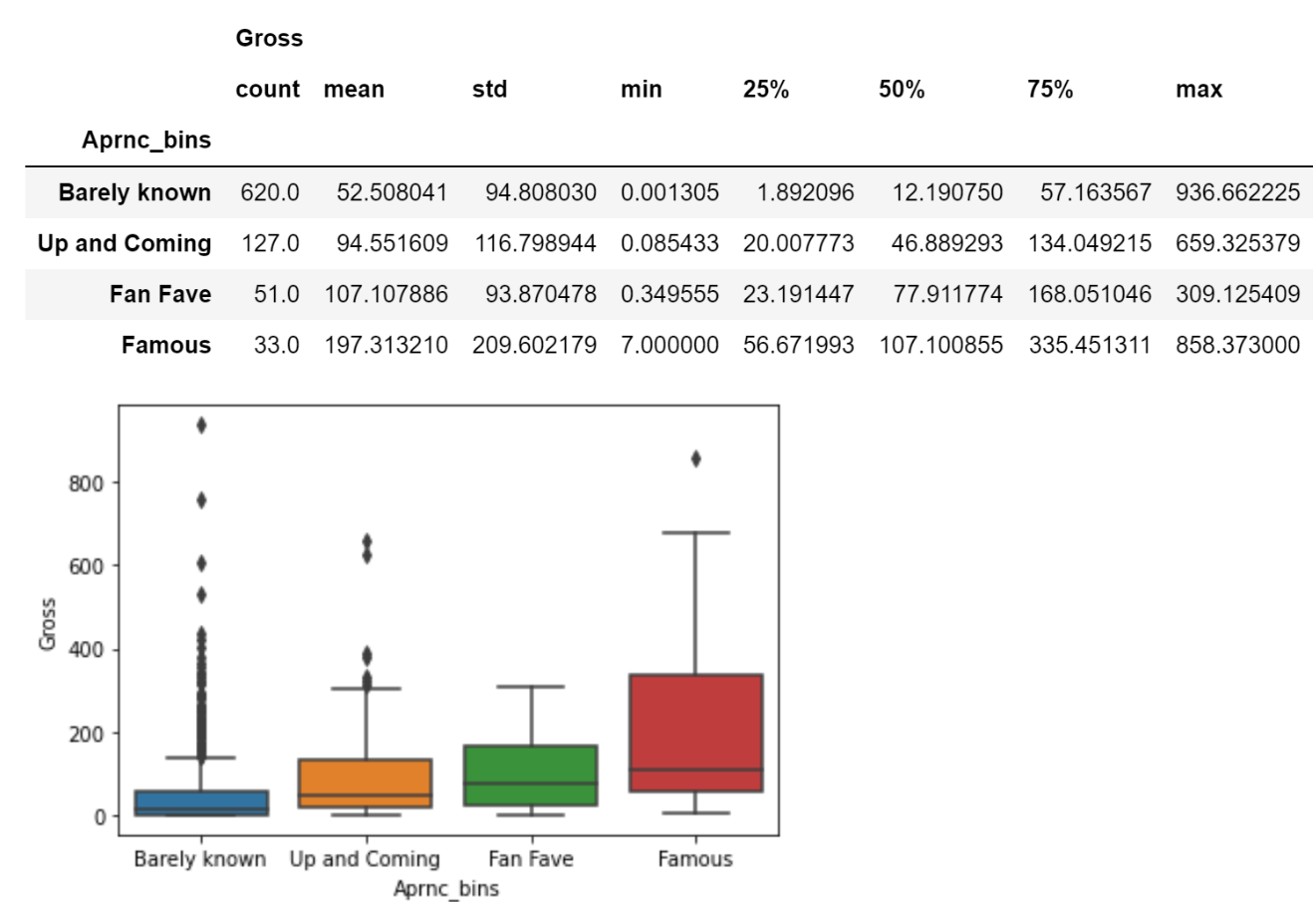
# 

# 

# Insights

* One major insight that can be gleaned from my machine learning algorithm is that of all the columns used, the Director is the clear leader in what attribute most signifies a high grossing film. This might come as surprising to people who assume that with the amount of marketing done around them that the cast would be the strongest signifier. It makes more sense when you come to the realisation that three of the top four highest grossing films of all time were directed by the same man, James Cameron.

* Another being that release has no significance in terms of gross revenue. This is shown in reality again through James Cameron that Avatar (2009), Avatar: The Way of Water (2022), and Titanic (1997) are the highest, third, and fourth highest-grossing films of all time, respectively. Each over a decade apart from the previous.
* In the below chart, we can gain the insight that whilst a star studded cast isn’t always a sure fire way of gaining a box office smash (See: famous min 7mill), that definitely will help your movies chance greatly of being a success. Looking at the mean values steady rise according to the pedigree of star power used.

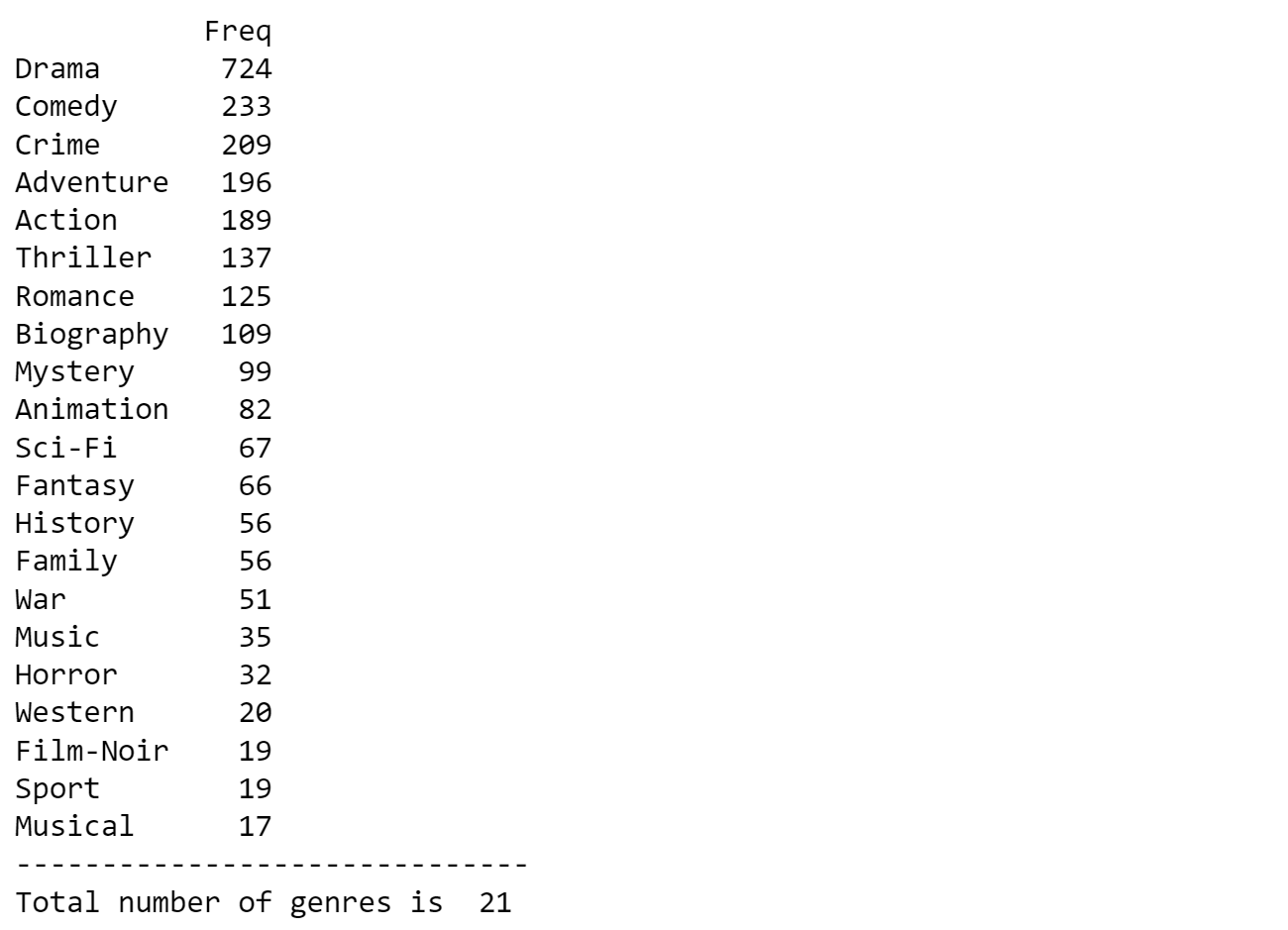


* Another insight gained would be that with reasonable confidence you can predict what rating a film will have given its genre. Using the below chart of IMDB top 1000 movies data you can see that



Just by assumption you can guess that a film in the Sports genre would be universal for everyone, where a horror movie would highly trend towards being adult orientated. This graph backs that assumption that aside from a few outliers that genres like pointed towards a certain type of content be it safe for all or more serious topics generally reflect this with their ratings also.

* Seen below is the frequency of genres found in the top 1000 IMDB movies. It follows the trend where more niche genres such as sport and musicals are less popular than more universal broad genres like comedy and crime. This doesn’t signify that they are inherently bad as they still can be seen in the top 1000 rated movies, but maybe that they are less likely to be produced as the more common genres with wider audience appeal such as a drama.



# References

<https://github.com/conortcahalane/UCDPythonProject2023>

<https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows>

<https://www.kaggle.com/datasets/shivamb/netflix-shows>

<https://jupyter.org/>

<https://pandas.pydata.org/>

<https://numpy.org/>

<https://plotly.com/>

<https://scikit-learn.org/stable/>

<https://seaborn.pydata.org/>