Cloud Analytics Pipeline

Introduction –

With the growth in streaming services such as Netflix and Amazon Video, binge-watching media has become a key source of entertainment for many people. On average, a person watches about 3.5 hours of television each day, thus making it a significant chunk of our daily routine [1]. About 329 movies were released in 2020 in the US and Canada alone, around one movie per day in a calendar year [2]. Choosing on a movie night from many options would have been so difficult without a central movie information online database such as IMDB and rotten tomatoes. Like many, I also refer to the movie rating and length, besides the trailer, before attempting to open one. The first motion picture film ever made in 1888 was only 2 seconds long compared to 90 minutes in 2021. This brought my curiosity to analyze how movies have evolved since the late 19th century. Is there any correlation between movie length and average rating? Do we see an interaction between release year and movie length for a movie rating? We will answer all these questions by building a data pipeline in a distributed environment.

The IMDB movie data contains over 250k movies released between late 1800 and early 2021. Since the data is huge, we will leverage distributed environment to process and store data and results. We will deploy everything in the google cloud platform for its reliability, availability, scalability, and fault tolerance. The importance of the pipeline is to streamline reporting and publishing results. We will build an IMDB data pipeline in distributed environment starting from the creation to publishing stage. All the steps are detailed in the latter part of the report.

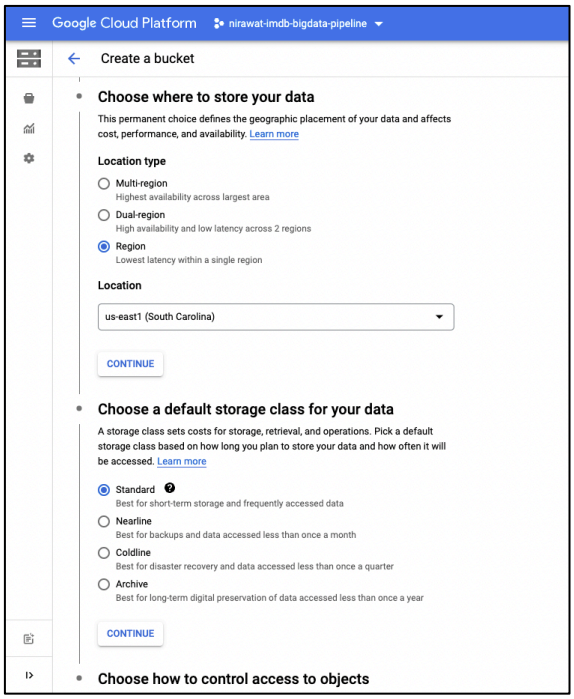
Furthermore, this report entails the importance of data pipeline and cloud computing & storage. The increase in the adaptation of smart or IoT devices has made it challenging to store and process big data in a single system. Traditional data warehouses have helped overcome this obstacle to an extent, but factors like reliability, availability and economic cost were still a challenge. For this reason, many companies have started moving their system to cloud-based data warehouses from the traditional ones. Cloud infrastructure has simplified and solved storage problems and provided computing resources to process data or launch applications at scale. The purpose of this report is to get acquainted with the data analysis pipeline in a distributed cloud environment.

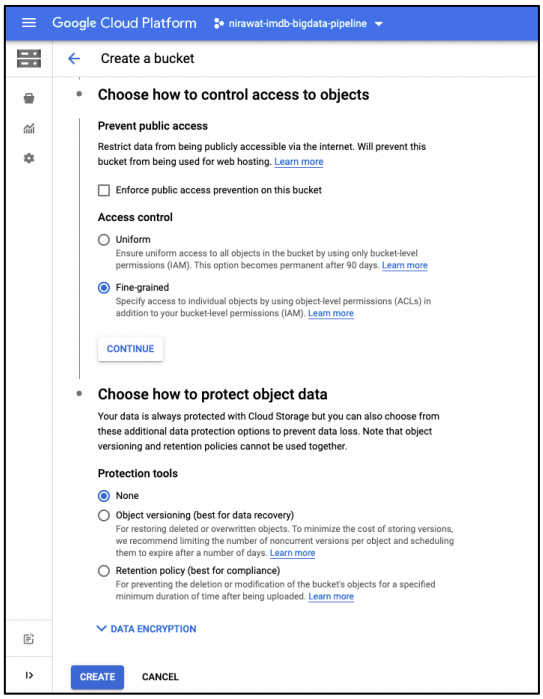
Set up Cloud Infrastructure:

- ✓ **Account Login**
  - Create a google cloud account and log in to the google cloud console.
  - Create a project

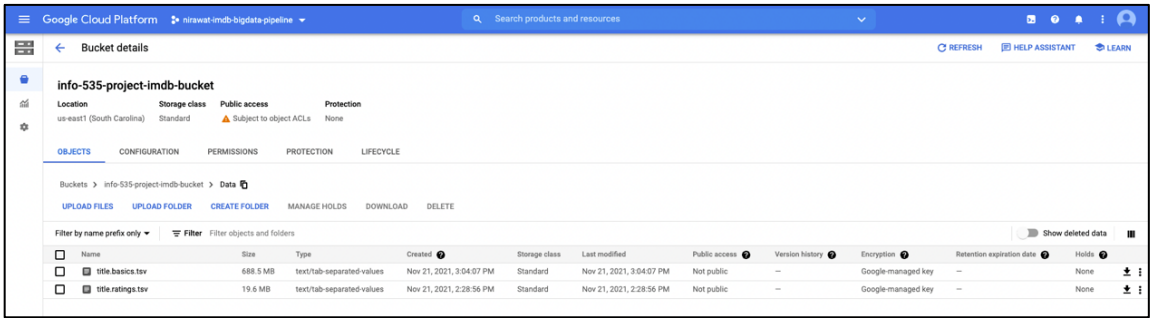
- ✓ **Create Storage**
  - **Create a bucket:**
    Name your bucket and choose the below settings. I enabled control access at the object

level in the bucket by selecting the fine-grained access control option. This brings flexibility to custom access at the file level. All other settings are kept the same as the default.





- **Upload files:**
  Once the bucket is created, click on *UPLOAD FILES* to upload files from the local machine. This will bring the data to the cloud storage.



✓ **Create cluster**

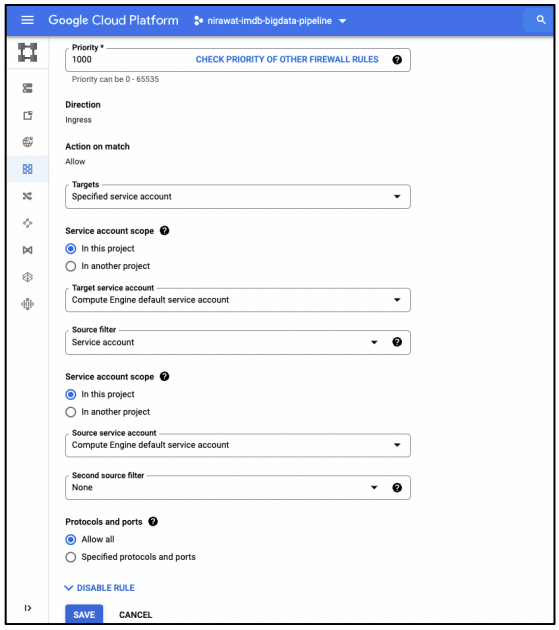
- **Enable the following APIs:**
  Go to *API Services* > *Library*, search, and enable each of the following APIs.
  - **Compute Engine API:** Creates and runs virtual machines on Google Cloud Platform.
  - **Cloud DataProc API:** Manages Hadoop-based clusters and runs jobs on Google Cloud Platform.
  - **Cloud Storage:** Google Cloud Storage is a RESTful service for storing and accessing your data on Google's infrastructure.

- **BigQuery API:** A data platform for customers to create, manage, share and query data.

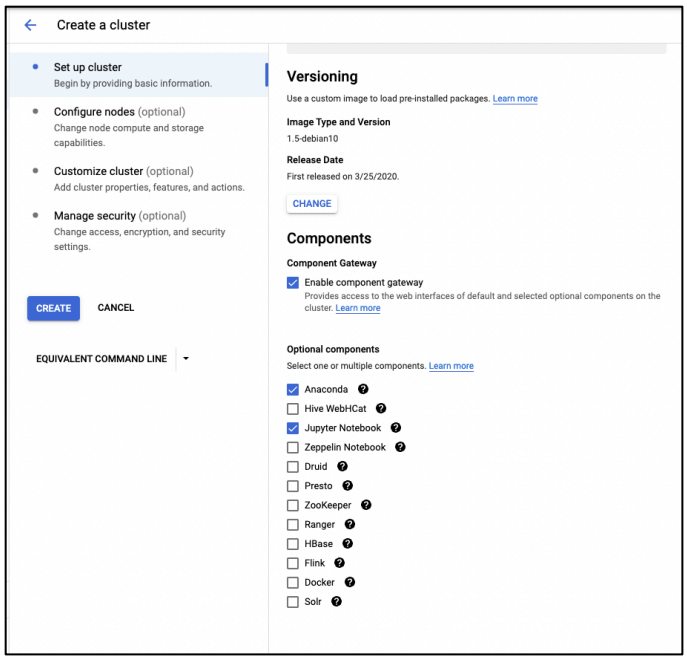
- **Create a VPC Network:**
  A Virtual Private Cloud (VPC) network is a virtual version of a physical network. It helps connect VM instances and other google cloud products in addition to load balancing.

  Go to *VPC Networks* > *Create a VPC Network*
  - Enter VPC Name
  - Set *Subnets* to **Automatic**
  - Hit *Create*

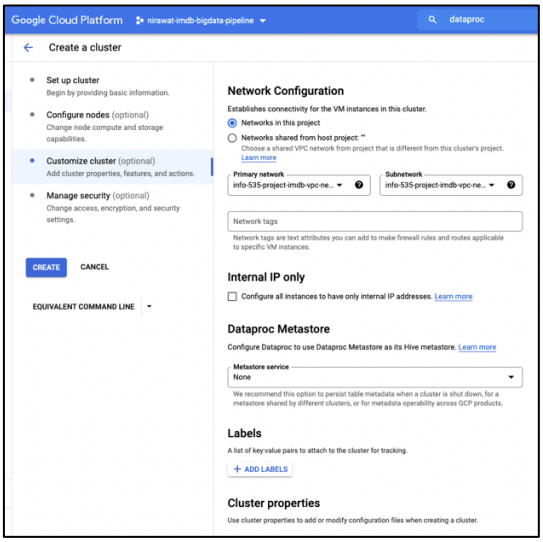
- **Create a firewall rule:**
  A firewall rule will target only a particular type of incoming and outgoing traffic in the VPC network.
  - Go to VPC Networks
  - Find and click *Firewall* on the left pane
  - Give this firewall a name
  - Set Target to the below values
  - Keep the rest as default

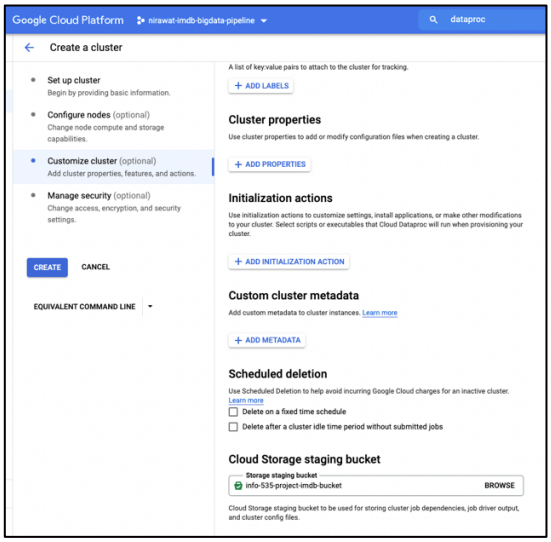


- **Create a cluster:**
  - Go to *DataProc* and click on *Create a Cluster* button.
  - Give the cluster a name and assign a location
  - Keep all the settings the same except for the below:



- Go to *Customize cluster* option and enter the following settings:

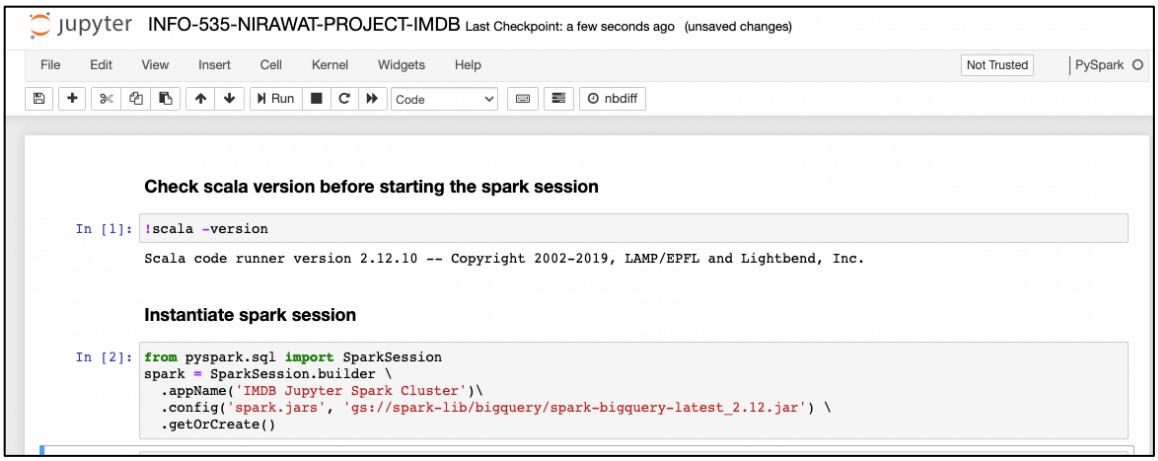




- Hit on the *CREATE* button

✓ **Connect to the Cluster and Data:**

- Go to the cluster created above and click on the *Jupyter* notebook under the *WEB INTERFACES* tab.
- Go to the *GCS* folder and create a python notebook. Creating any notebook here will also make it appear in the cloud bucket directory under /notebooks/jupyter/
- Instantiate spark session by configuring the correct spark.jar version that is compatible with the scala version.



- Read files in the cloud bucket from the following path gs://{{bucket_name}}/{{filename}}

Build Data Pipeline

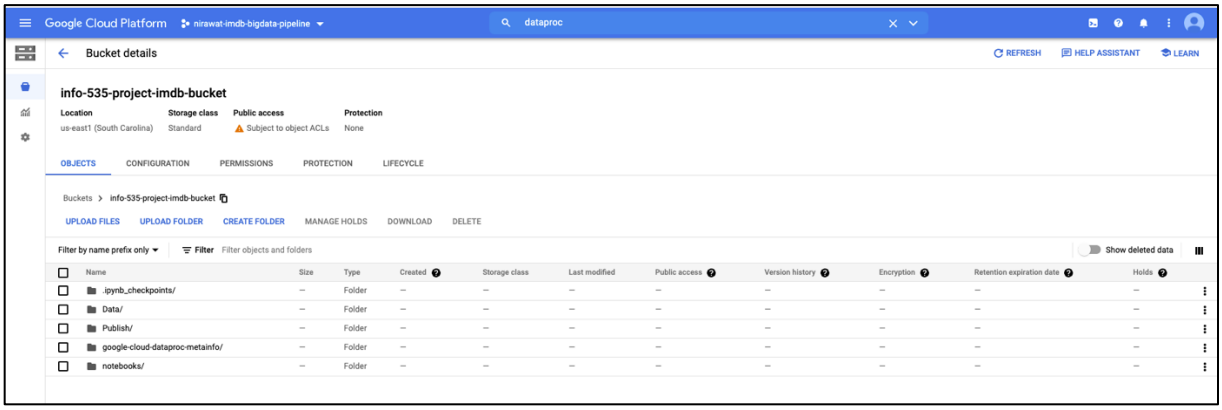
Similar to the HSGS model, the below data pipeline steps are followed to create the IMDB data pipeline.

1. **Create:**
   Data is downloaded to the local machine from the website https://datasets.imdbws.com/. The above steps were followed to create the bucket. Files were uploaded files into the bucket from the local machine. Post creating the storage in google cloud, we set up a Hadoop cluster and created a Jupyter instance.

2. **Process:**
   Following steps are employed to process data for analysis
   - Read Data: Read data from google cloud bucket
   - Import Libraries: All the required libraries were installed
   - Data Cleaning and Preparation: Merged movie basics and movies rating data using pyspark followed by treating missing values logically. Data transformation such as aggregation is done to do specific analysis.
     All the steps are detailed in the notebook: https://storage.cloud.google.com/info-535-project-imdb-bucket/Publish/INFO-535-NIRAWAT-PROJECT-IMDB.html?authuser=1

3. **Analyze:**
   Data is analyzed using pandas in the Jupyter notebook. Plots are made using matplotlib and seaborn. The result is mentioned below this section.
   All the steps are detailed in the notebook: https://storage.cloud.google.com/info-535-project-imdb-bucket/Publish/INFO-535-NIRAWAT-PROJECT-IMDB.html?authuser=1

4. **Preserve:**
   All the files are saved in the google cloud bucket for retrieval and editing in the future.



Data: gs://info-535-project-imdb-bucket/Data

Notebook: gs://info-535-project-imdb-bucket/notebooks/jupyter/

Published Files: gs://info-535-project-imdb-bucket/Publish/

5. **Share:**
   The final visualization is published at: https://storage.googleapis.com/info-535-project-imdb-bucket/Publish/imdb.png
   *The website is given public access.

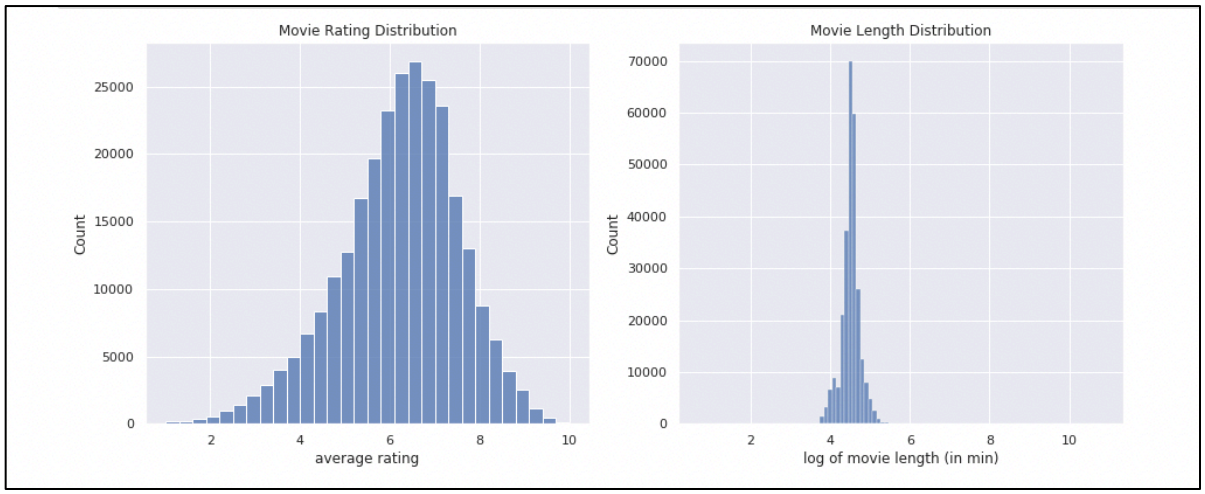
   Notebook is published at:
   https://storage.cloud.google.com/info-535-project-imdb-bucket/Publish/INFO-535-NIRAWAT-PROJECT-IMDB.html?authuser=1

   The notebook is given only iu.edu domain access for data security purposes.

Results-

Below are the results from our analysis in the notebook: https://storage.cloud.google.com/info-535-project-imdb-bucket/Publish/INFO-535-NIRAWAT-PROJECT-IMDB.html?authuser=1

Univariate Analysis-

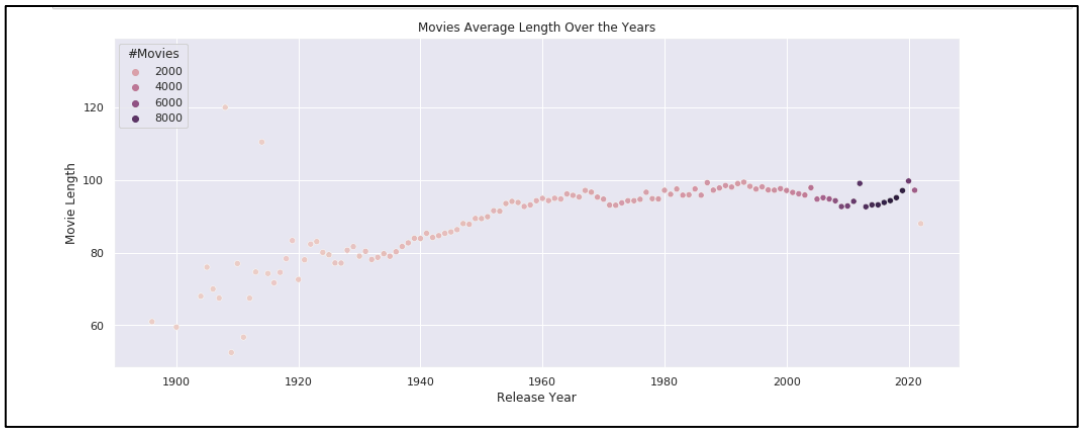


- The rating distribution doesn't have outliers and resembles a roughly normal distribution. On examining the runtime minutes distribution, I see that eliminating a few extreme outliers can bring the distribution close to normal but taking a log can eliminate skewness due to outliers.

- Since there are many data points compared to the distinct values available of runtime (n = 238), years (121), the scatter plot between the three looks cluttered for a unique combination of average rating and the other two variables. Thus, we will not be able to judge if there is a nonlinear pattern. Additionally, due to computational challenges, we will go ahead with the linear model instead.

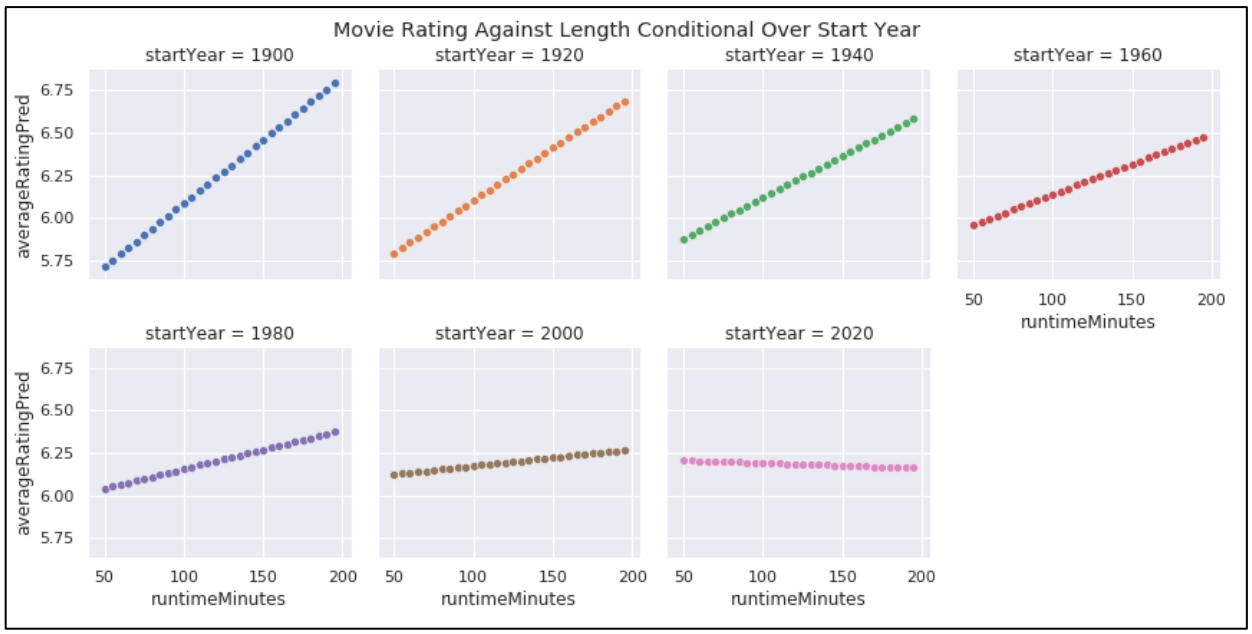
- A rating voted by many users is deemed to possess higher confidence that the rating is true or close to the calculated one compared to a movie rating rated by a few. So, we may assume that all movies are closer to the mean (our prior belief) if there are a few votes and closer to the true value for a higher number of votes. However, it seems difficult to incorporate several votes in movie rating since the rating will be manipulated in such a case and converge towards the mean for many movies rated lower. In this case, almost all movies would have converged to around 6.5. So, we won't weigh the rating based on the number of votes and thus eliminate it from our analysis. weighted_rating = mean_rating(1 - w) + wcurrent_rating where w = (n - min(n))/(max(n)-min(n)) and n = number of votes

Bivariate Analysis-



The average movie length has increased from 60 minutes to about 100 minutes over the years. One of the possible reasons for this would be technological advancement. Additionally, the number of movie releases has also considerably increased over the years because of the increase in production houses.

Trivariate Analysis-



We see a positive correlation between predicted rating and movie length, i.e., longer movies were used to rate higher in the early years. However, the trend seemed to decline/reverse as we progressed forward in time.

Discussion-

Longer movies released before the 21st century tend to have higher IMDB ratings than their shorter counterpart. However, the trend is reversed for the film released in the 21st century, i.e., longer movies tend to have a lower rating than their shorter counterpart. The positive correlation between predicted rating and movie length in the early years could be because fewer movies were present then, and longer films might have been considered more entertaining. However, one of the possible reasons for a reverse trend in the 21st century could be that people rate movies now based on the quality of the content rather than movie length. In the current fast-paced world, people find longer (> 120 mins) movies boring. At least I do.

I borrowed concepts from the HSGS model to build this data pipeline. I chose google cloud after accounting for the economy, ease of development, reliability, and high availability. All the files were stored in the google cloud bucket, and the Jupyter instance was created in the Hadoop cluster. The data was imported in the Jupyter notebook directly from the cloud bucket. We used a mix of Pyspark and python operations for the analysis and modeling. For its fault-tolerant and distributed nature, Pyspark was used for processing large data, whereas python was used for modeling and visualization purposes. The results were then exported and published from the google bucket.

I encountered multiple challenges throughout the project execution, majorly while setting up the cluster. Below are some of the major obstacles:

- Creating a cluster: Debian 2.0 VM doesn't support anaconda and Jupyter notebook. So I changed it to 1.5.
- Setup Global network: It seems like Indiana University has disabled the network connection. So, I had to create my own. The setting is given in the former steps of the cloud setup.
- Enabling Firewall: I could not find the settings anywhere, so I set it up using the trial-and-error method. The setting is given in the above steps.
- Link cluster with cloud bucket
- Instantiating Spark
- Saving image from the Jupyter notebook in the cluster to the cloud bucket.

Conclusion-

We analyzed and modeled IMDB movie data containing over 250k movies in a distributed computing environment by successfully building a data pipeline in the google cloud starting from the creation stage to publishing our results. In this project, we integrated google cloud storage for preserving our data and results, and the Hadoop cluster to run MapReduce jobs from the Jupyter instance. We used both pyspark and python for our analysis- Pyspark uses distributed computing technology which makes the analysis way faster and thus is suitable for processing and transformation of large data, whereas python is known for its flexibility and visualization. We saw one primary use case of the IaaS cloud infrastructure with this project.

References-

[1] https://www.studyfinds.org/survey-the-average-adult-will-watch-more-than-78000-hours-of-tv/
[2] https://www.statista.com/statistics/187122/movie-releases-in-north-america-since-2001/