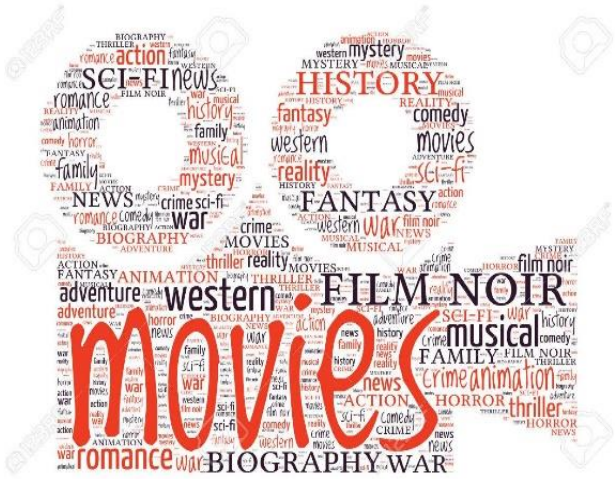
MACHINE LEARNING PROJECT

INTRODUCTION



Movies as a whole are a work of visual art that simulates experiences in addition to conveying ideas, stories, perceptions, sensations, beauty, or ambiance through the use of moving images. Films are cultural artifacts created by specific cultures, facilitating intercultural dialogue. It is regarded as a significant artistic medium that offers amusement and historical significance, often graphically chronicling a specific period. The making and showing of motion pictures became a source of profit almost as soon as the process was invented. The showbiz or the movie world in the modern era has grown to an extent where some movies have grossed over 2.5 billion which is greater than the GDP of some countries.

The film industry is involved in the creation and distribution of films as a product. This industry is a worldwide phenomenon, but some countries have a greater role in production than others. This industry generally employs a wide number of people directly through production and distribution process, but it also employs a sizable number of people indirectly because of the money spent to produce films. In most cases, the film industry is focused on films as a medium for entertainment, but occasionally, it may utilize films for educational purposes as well. There are certain places such as Hollywood California and Hong Kong that tend to become centres for the film industry. As a result, certain nations generate a disproportionately large number of films compared to others. These countries will then distribute their products to the rest of the world, translating the languages and taking relevant steps to increase sales.

In the contemporary period, there are numerous production organizations that handle various movie projects with diverse casts in a variety of genres on an annual basis. It is challenging to keep track of all the movies being produced at once due to the large number of movies being created globally. A movie's rating is determined on its plot and content. This rating aids in weeding out younger viewers who are not yet emotionally and mentally mature enough to watch a movie which is inappropriate for their age Then there are websites that provide ratings, such as IMDB, Rotten Tomatoes, and Metacritic, and these websites essentially assist us, the audience, in deciding whether to invest our time in a certain movie depending on the score they provide. The higher the score the better the movie.

A lot of work has been done in the movie industry. As this field has a history which goes back up to a hundred years, there's a lot of data available to general public to draw some interesting insights and conclusions. Highly proficient movie recommendations systems have been built already which are being used to by many big multinational companies like Netflix, Hulu, Amazon , Apple etc. The only way forward is to optimize these recommendations using the latest accurate data. There are other areas such as visual effects/postproduction, computer animation, web-based marketing but

these sectors don't come under the field of machine learning. This study tries to classify movies into positive or negative based on the reviews or feedback of the audience and recommend movies of similar interest to people who previously enjoyed a movie of similar style/genre.
Profit is a key force in the industry, due to the costly and risky nature of filmmaking; many films have large cost overruns. Yet many filmmakers strive to create works of lasting social significance. Revenue in the industry is sometimes volatile due to the reliance on block buster films released in movie theatres. The rise of alternative home entertainment has raised questions about the future of the cinema industry, and Hollywood employment has become less reliable, particularly for medium and low-budget films.

ANALYSIS

Gathering and Loading Data

To cover the dynamics of the movie industry, only a dataset which contained information on various aspects of a particular movie would do justice. I wanted to find a dataset which covered all these aspects and I managed to gather good data on Kaggle. The data which has been gathered has information regarding all the movies which had released from the year 1980 to the year 2016. It had information regarding the director, writers, cast, gross, budget, release date, company, country and so on. The score has been scraped from TMDB. A snippet of the data has been included below.

```
In 15
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8)

In 150
df= pd.read_csv("movies.csv")
pd.set_option('display.max_columns', None)
print(df.columns.values)

['name' 'rating' 'genre' 'year' 'released' 'score' 'votes' 'director'
 'writer' 'star' 'country' 'budget' 'gross' 'company' 'runtime']

In 151
df.head(20)
```

Out 151

	name	rating	genre	year	released	score	votes	director	writ
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Her
2	Star Wars: Episode V - The Empire Strik...	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	
5	Friday the 13th	R	Horror	1980	May 9, 1980 (United States)	6.4	123000.0	Sean S. Cunningham	
6	The Blues Brothers	R	Action	1980	June 20, 1980 (United States)	7.9	188000.0	John Landis	
7	Raging Bull	R	Biography	1980	December 19, 1980 (United States)	8.2	330000.0	Martin Scorsese	

20 rows × 15 columns Open in new tab

The dataset contains 7688 rows and 15 columns, and these rows and columns contain data on various aspects of a particular movie. This data frame above shows the first 20 movies in this dataset. This dataset contains categorical variables as well as numerical variables.

Cleaning Data

This data set has a lot of missing values, and these missing values need to be addressed in order to achieve good results for exploratory data analysis. Missing values can be easily addresses using the drop_na() method.

```
In 25
#check the dimensionality
df.shape
```

```
Out 25
(7668, 15)
```

```
In 30
#Drop rows with missing data
cols = ['rating', 'released', 'score', 'votes', 'writer', 'star', 'country', 'budget', 'gross', 'company', 'runtime']
for col in cols:
    df.dropna(subset=[col], inplace=True)
df.reset_index()
df.head()
```

Out 30

	score	votes	director	writer	star	country	budget	gross	company	runtime
ed States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.	146.0
ed States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures	104.0
ed States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm	124.0
ed States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramount Pictures	88.0
ed States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Orion Pictures	98.0

5 rows × 15 columns Open in new tab

```
In 31
#check dimensionality after dropping na values
df.shape
```

```
Out 31
(5421, 15)
```

The data types of the columns need to be converted to integer from float to avoid future hindrance with analysis of the dataset. The categorical variables also had to be converted to numerical data.

```
In 54
#Convert our categorical features into numerical values
df_n = df.copy()
for col in df_n.columns:
    if df_n[col].dtype == 'object':
        df_n[col] = df_n[col].astype('category').cat.codes
df_n.head()
```

Out 54

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime
5445	386	5	0	2009	527	7.8	1100000	785	1263	1534	47	237000000	2847246203	1382	162.0
7445	388	5	0	2019	137	8.4	903000	105	513	1470	47	356000000	2797501328	983	181.0
3045	4909	5	6	1997	534	7.8	1100000	785	1263	1073	47	200000000	2201647264	1382	194.0
6663	3643	5	0	2015	529	7.8	876000	768	1806	356	47	245000000	2069521700	945	138.0
7244	389	5	0	2018	145	8.4	897000	105	513	1470	47	321000000	2048359754	983	149.0

5 rows × 15 columns Open in new tab

The categorical variables had to be converted to the numerical data because in my EDA I found that I was unable to draw proper correlations between various columns of the dataset as the data was in needed was in categorical form and some data was in numerical form. I needed data which in a uniform manner so I had to convert the data into numerical data to draw proper conclusions.

```
In 32  1  #check datatypes for each columns
       2  df.dtypes
```

```
Out 32
                data
        year    int64
    released   object
       score  float64
       votes  float64
    director   object
      writer   object
        star   object
     country   object
```

Length: 15, dtype: object Open in new tab

```
In 33  1  #Convert Budget,Gross and votes columns to integers
       2  df['budget']=df['budget'].astype('int64')
       3  df['gross'] = df['gross'].astype('int64')
       4  df['votes'] = df['votes'].astype('int64')
```

```
In 34  1  df.dtypes
```

```
Out 34
                data
        name    object
      rating    object
       genre    object
        year     int64
```

I used df.isna() method to check for missing values in columns and when I noticed which columns had missing values, I created a subset and passed this subset to drop.na() method to drop all the missing values.

The dimensionality of our dataset was 7668 rows with 15 columns, but the missing values were dropped, and the dimensionality reduced to 5421 rows with 15 columns of data. The dataset now has no missing values and only contains good data indexed by name of the movie. This is helpful because it will be easy for to draw insights from the dataset.

ANALYSES: ARM

Due to data limitations acquiring more data became crucial for ARM. ARM requires data in the form of transactional data. The data I possessed was a which was a combination of categorical and numeric data. Proper cleaning of the dataset was required for ARM to convert it to transactional data. Many columns were dropped to reduce dimensionality and rows with NA values were filtered out.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

ARM.Rmd*

Knit on Save | Knit | Run

Source | Visual

```r
movie <- read.csv("movie.csv")
ratings <-  read.csv("rating.csv")
```

```r
library(tidyverse)
movie <- as.tibble(movie)
ratings <- as.tibble(ratings)
```

```r
dataset <- merge(ratings, movie, by = "movieId")
dataset <- dataset[,-c(1,4)]
dataset
```

Description: df [20,000,263 × 4]

userId	rating	title	genres
124152	5.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
93599	4.5	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
136201	2.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
8863	5.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4903	4.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
28307	5.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
92085	4.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
63879	5.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
539	5.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
20040	3.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

1-10 of 20,000,263 rows Previous 1 2 3 4 5 6 ... 100 Next

```r
d <- dataset %>% select(userId, rating , title)
d <- d %>% filter(rating > 4) %>% arrange(userId)
d <- d %>% select(userId,title) %>% arrange(userId)
head(d)
```

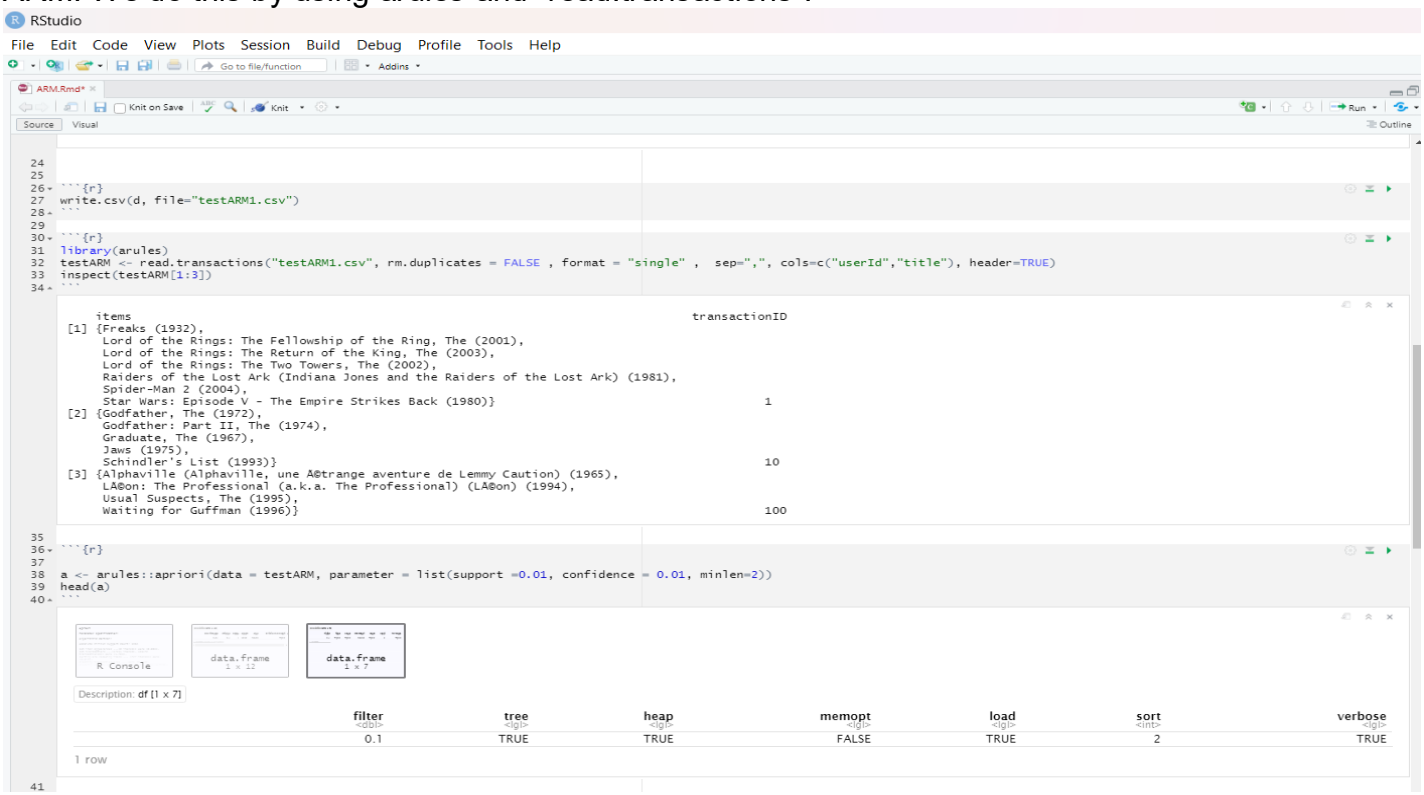
Description: df [6 × 2]

	userId	title
1	1	Star Wars: Episode V - The Empire Strikes Back (1980)
2	1	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
3	1	Lord of the Rings: The Fellowship of the Ring, The (2001)
4	1	Lord of the Rings: The Two Towers, The (2002)
5	1	Lord of the Rings: The Return of the King, The (2003)
6	1	Freaks (1932)

6 rows

Here we have two data sets movie.csv and rating.csv. They had to be merged by “movie ID” into a variable named dataset. It has 20 million rows and it has userID, rating, tittle, and genres as columns. The dataset has been cleaned according to our requirement and is arranged by userID and rating greater than 4. The final cleaned dataset is stored in a variable d and it has around 4.5 million rows.

The dataset ‘d’ has been written to csv format and we start to convert our csv to transaction data for ARM. We do this by using arules and “read.transactions”.



The format for the transaction data is “single”. The preview for the transaction has been shown. We then use the Apriori algorithm to generate rules.

HOW ARM WORKS

Association rule mining is a procedure which aims to observe frequently occurring patterns, correlations, or associations from datasets found in various kinds of databases such as relational databases, transactional databases, and other forms of repositories.

ARM has three main factors associated with it to determine whether a rule is interesting or not. It is defined as interestingness. The three main factors are support, confidence, and lift. Lets take Iron man and Avatar as examples.

Support: - Support is the number of times a transaction occurs in a dataset. This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. Support can be calculated by

$$\frac{\text{Number of times a transaction occurs in a dataset}}{\text{Total number of transactions}}$$

Confidence: - Confidence is basically conditional probability. For example, what is the probability that a person will watch avatar given that he has watched iron man?

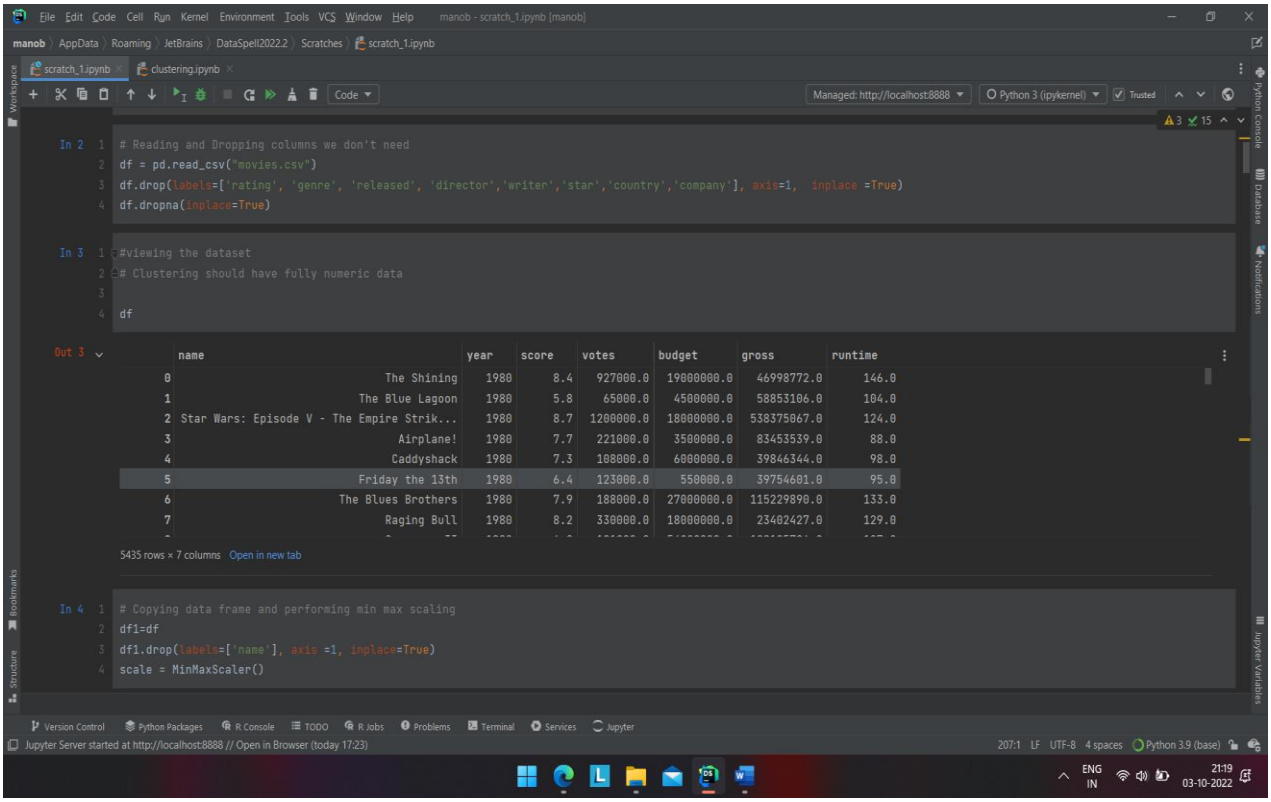
$$\frac{\text{(Count of Iron man and Avatar together)}}{\text{(Count of Iron man)}} \quad \text{OR} \quad \frac{\text{Support(Ironman, Avatar)}}{\text{Support(Ironman)}}$$

Lift :- Lift tells us whether a rule is worth exploring or not. A lift value greater than 1 means that item Y is *likely* to be bought if item X is bought, while a value less than 1 means that item Y is *unlikely* to be bought if item X is bought.

$$\frac{\text{Support(Ironman, Avatar)}}{\text{Support(Ironman) * Support(Avatar)}}$$

ANALYSES: CLUSTERING

Clustering requires data in numeric form. The data in possession was completely in record format. Conversion of data to numeric format was achieved through Min Max scaling and dropping out string-based columns.



```
In 2
# Reading and Dropping columns we don't need
df = pd.read_csv("movies.csv")
df.drop(labels=['rating', 'genre', 'released', 'director','writer','star','country','company'], axis=1,  inplace =True)
df.dropna(inplace=True)

In 3
#viewing the dataset
# Clustering should have fully numeric data

df
```

Out 3

	name	year	score	votes	budget	gross	runtime
0	The Shining	1980	8.4	927000.0	19000000.0	46998772.0	146.0
1	The Blue Lagoon	1980	5.8	65000.0	4500000.0	58853106.0	104.0
2	Star Wars: Episode V - The Empire Strik...	1980	8.7	1200000.0	18000000.0	538375067.0	124.0
3	Airplane!	1980	7.7	221000.0	3500000.0	83453539.0	88.0
4	Caddyshack	1980	7.3	108000.0	6000000.0	39846344.0	98.0
5	Friday the 13th	1980	6.4	123000.0	550000.0	39754601.0	95.0
6	The Blues Brothers	1980	7.9	188000.0	27000000.0	115229890.0	133.0
7	Raging Bull	1980	8.2	330000.0	18000000.0	23402427.0	129.0

5435 rows × 7 columns Open in new tab

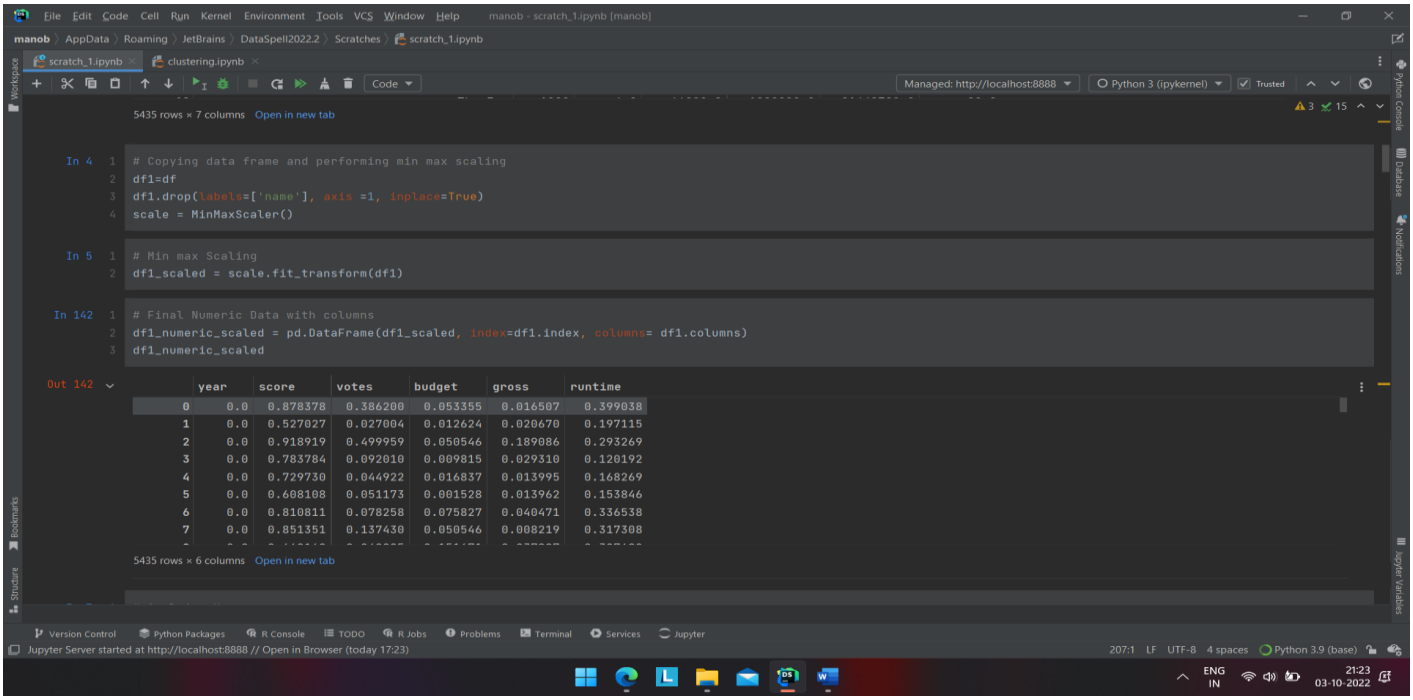
```
In 4
# Copying data frame and performing min max scaling
df1=df
df1.drop(labels=['name'], axis =1, inplace=True)
scale = MinMaxScaler()
```

The name of the movie was a crucial element which could not be dropped. But for MinMax scaling the name column will be dropped and added back once data has been scaled.

After scaling the data and converting it into numeric format we can perform various clustering algorithms on it to extract patterns and results.

SCALING AND FORMMATING

Data is scaled using min max scaling and the name column has been added back to the data frame for analysis. The data is fully numeric and ready for clustering.



Unsupervised algorithms typically draw conclusions from datasets using only input vectors without considering predetermined or labelled results.
Grouping unlabeled examples is called clustering.
As the examples are unlabelled, clustering relies on unsupervised machine learning. If the examples are labelled, then clustering becomes classification.

KMEANS CLUSTERING

A cluster refers to a collection of data points aggregated together because of certain similarities. You'll specify a goal number k, or the quantity of centroids required in the dataset. The location that, whether real or imagined, serves as the cluster's centroid. Each data point is assigned to a certain cluster by minimizing the total of squares within each cluster. In other words, the K-means method finds k centroids and then assigns each data point to the closest cluster while minimizing the size of the centroids. The *'means'* in the K-means refers to averaging of the data; that is, finding the centroid.
The K-means technique in data mining uses a first set of randomly chosen centroids as the starting points for each cluster in order to process the learning data. From there, iterative (repetitive) calculations are performed to optimize the positions of the centroids within the cluster. It stops developing and enhancing clusters when either:

1) Since the clustering was effective, the centroids have stabilized; their values have not changed.
2) Iterations have reached the predetermined number.

HIERARCHICAL CLUSTERING

A measure of dissimilarity between sets of data is necessary in order to choose which clusters should be joined (for agglomerative) or where a cluster should be divided (for divisive). This is done using a linkage criterion, which determines the dissimilarity of sets as a function of the pairwise distances of observations in the sets, and an appropriate metric (a measure of distance between pairs of observations). The choice of an appropriate metric will influence the shape of the clusters, as some elements may be relatively closer to one another under one metric than another. For example, in two dimensions, under the Manhattan distance metric, the distance between the origin (0,0) and (0.5, 0.5) is the same as the distance between the origin and (0, 1), while under the Euclidean distance metric the latter is strictly greater.
The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations. There are many additional measurements of

dissimilarity. Particularly, correlation-based distances such the Spearman, Pearson, Eisen cosine, and Kendall correlation distances, which are frequently employed for the analysis of gene expression data. By deducting the correlation coefficient from 1, one can determine the correlation-based distance. Correlation-based distances cannot technically be utilized as metrics, although their square root may.
Some common examples of distance measures that can be used to compute the proximity matrix in hierarchical clustering, including the following:

- Euclidean Distance
- Mahalanobis Distance
- Minkowski Distance

Euclidean Distance: - Calculating the distance between two points, the Euclidean distance is a non-negative quantity. On the basis of the Pythagoras Theorem, the distance between these two places is calculated.
Mahalanobis Distance: -
As a type of tscore, the Mahalanobis distance is employed to determine the separation between tw o points. The Mahalanobis distance also considers the data's normalization and dispersion.
It is defined by the formula below.

$$d_S^2(x_i, x_k) = (x_i - x_k)' S^{-1} (x_i - x_k)$$

It is helpful for distributions that are not spherical because even if points A and B are equally dista nt from point X by Euclidean distance, the distribution may not be uniform.
Minkowski Distance: -The following formula provides a definition for the Minkowski distance.

$$d_M(x_i, x_k) = \sqrt[M]{|x_i - x_k|' |x_i - x_k|}$$

The weight given to greater and smaller variances varies based on the value of M, where M is an integer. If M = 10 and xi = (1,3) and xk = (2,3), for instance, then d10 = Square-root(|1-3|+|2+3|) = Square-root (3)
Since clustering is by its very nature an exploratory activity, the dataset can use a variety of clustering algorithms. Even if we used various clustering algorithms, we would anticipate getting the same result. However, as each technique employs a different computation to arrive at the result, the computations could vary.

DENSITY CLUSTERING

Density-Based A cluster in a data space is defined as a contiguous zone of high point density, separated from other similar clusters by contiguous regions of low point density. Clustering is a general term for unsupervised learning techniques that find different groups or clusters in the data. Outliers or noise are commonly defined as the data points in the dividing zones of low point density.
DBSCAN does not require the number of clusters as a parameter, in contrast to k-means. Instead, it extrapolates the number of clusters from the data and can find clusters of any shape (for comparison, k-means usually discovers spherical clusters). As I mentioned previously, DBSCAN relies on the -neighbourhood to roughly approximation local density, hence the algorithm includes two parameters:

1) ε: The radius of our neighbourhoods around a data point *p*.
2) *minPts* : The minimum number of data points we want in a neighborhood to define a cluster.

Using these two parameters, DBSCAN categories the data points into three categories:

Core Points: A data point p is a core point if Nbhd(p,ε) [ε-neighbourhood of p] contains at least minPts ; |Nbhd(p,ε)| >= minPts.

Border Points: A data point *q is a border point if Nbhd(q, ε) contains less than minPts data points, but q is reachable from some core point p.

Outlier: A data point o is an outlier if it is neither a core point nor a border point. Essentially, this is the “other” class.

EXPECTATION FROM DATA COLLECTED

The data is completely in numeric format so it should be clustered accordingly as we apply different clustering methods to the data. For my data I expect K-means and hierarchical to be clustered very well. I do not expect DBSCAN to perform that well with respect to my data.

ANALYSIS :- DECISION TREES

Overview of Decision Trees

A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered. The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization.

Before we dive into how a decision tree, let’s define some key terms of a decision tree.

Root node: The base of the decision tree.

Splitting: The process of dividing a node into multiple sub-nodes.

Decision node: When a sub-node is further split into additional sub-nodes.

Leaf node: When a sub-node does not further split into additional sub-nodes; represents possible outcomes.

Pruning: The process of removing sub-nodes of a decision tree.

Branch: A subsection of the decision tree consisting of multiple nodes.

A decision tree resembles, well, a tree. The base of the tree is the root node. From the root node flows a series of decision nodes that depict decisions to be made. From the decision nodes are leaf nodes that represent the consequences of those decisions. Each decision node represents a question or split point, and the leaf nodes that stem from a decision node represent the possible answers. Leaf nodes sprout from decision nodes similar to how a leaf sprouts on a tree branch. This is why we call each subsection of a decision tree a “branch.”

Building a decision tree involves construction, in which you select the attributes and conditions that will produce the tree. Then, the tree is pruned to remove irrelevant branches that could inhibit accuracy. Pruning involves spotting outliers, data points far outside the norm, that could throw off the calculations by giving too much weight to rare occurrences in the data.

GINI

The gini impurity measures the frequency at which any element of the dataset will be mislabelled when it is randomly labelled.

The minimum value of the Gini Index is 0. This happens when the node is pure, this means that all the contained elements in the node are of one unique class. Therefore, this node will not be split again. Thus, the optimum split is chosen by the features with less Gini Index. Moreover, it gets the maximum value when the probability of the two classes is the same.

Ginimin=1–(12)=0

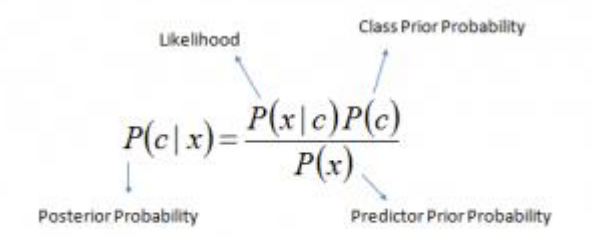
Ginimax=1–(0.52+0.52)=0.5

ENTROPY

Entropy is a measure of information that indicates the disorder of the features with the target. Similar to the Gini Index, the optimum split is chosen by the feature with less entropy. It gets its maximum value when the probability of the two classes is the same and a node is pure when the entropy has its minimum value, which is 0:
Entropymin=−1·log2(1)=0
Entropymax=–0.5·log2(0.5)–0.5·log2(0.5)=1
GINI vs ENTROPY
Gini Index has values inside the interval [0, 0.5] whereas the interval of the Entropy is [0, 1]. In the following figure, both of them are represented. The gini index has also been represented multiplied by two to see concretely the differences between them, which are not very significant.
Computationally, entropy is more complex since it makes use of logarithms and consequently, the calculation of the Gini Index will be faster.
Modelling Data
My data was completely numeric and high dimensional. It had many features ranging from movies budget to actor's rating , age etc. I planned to predict the collection a movie made by taking into account all the features a create a decision tree which would basically predict how much my movie would make based on said features.
Decision tree classifier was a not a good choice for my data as I was getting poor accuracy scores, hence I used a decision tree regressor to model my data. The accuracy score for my decision tree regressor was pretty good. I had to do a bit of data cleaning to completely get my data into the format required in order implement my decision tree.
Decision Tree Classifier is used to solve classification problems. For example, they are predicting if a person will have their loan approved. Decision Tree Regressor is used to solve regression problems. For example, prediction of how many people will die because of an opiate overdose.
I used mean squared error and R2_score to calculate my accuracy. My regression tree shows these values in each of the leaf nodes which separating the nodes based on a variable.
I had to implement a maximum depth of 3 based. I tried different depths too, but these images are not included in the document but can be found in the code.

ANALYSIS-NAÏVE BAYES

Overview Naïve Bayes
It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.
Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.
Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

There are three types of Naive Bayes model under the scikit-learn library:

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.
- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".
- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Modelling Data

My data consists of movie reviews along with a label indicating whether a movie review was positive or negative. I had to drop all other columns which were not useful in influecing my label and only kept those labels which made an impact on my label. My data consisted of 10,000 movie reviews but I reduced it to 5,000 and evenly divived positive and negative movies to maintain a balance in the label outcome.

As the movie reviews were entirely text data I had to clean the data extensively and convert them to bag of words format in order to continue with naïve bayes. I had to remive the stopwords by tokenizing the data and going through the data sentence by sentence. I haad to remove any unwanted punctuations in the data. I had to then stem the words inorder to decrease the length of the words. Lemmatizing took a lot of time, hence I avoided lamatizing my data.

Overall, after the cleaning was done I left with clean data. I had to then use count vectorizer to convert them into bag of words format inorder to conduct naïve bayes on the data.

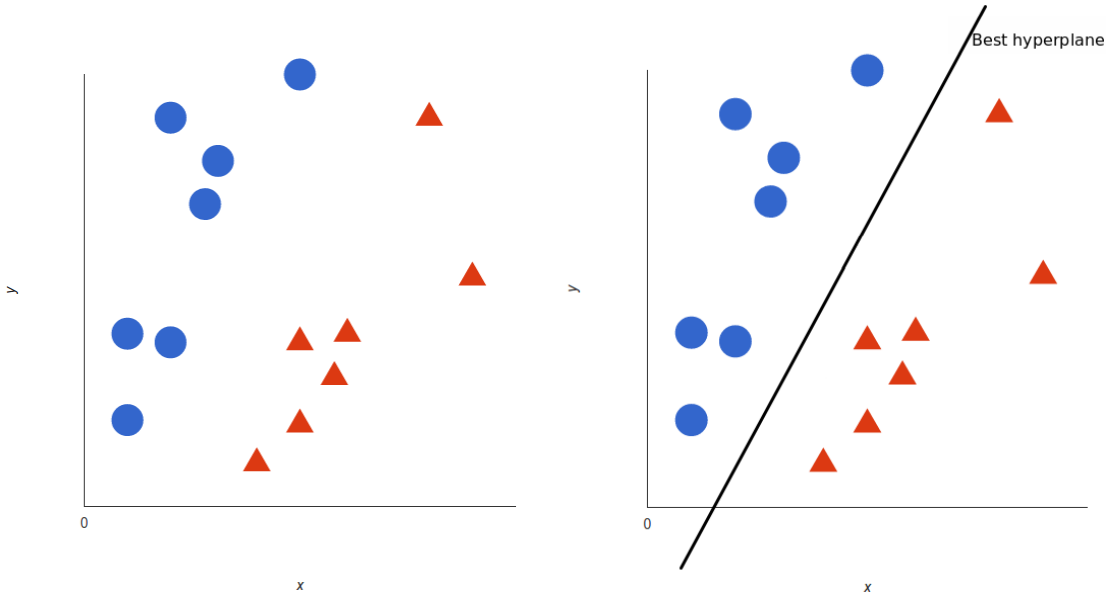
ANALYSIS :- SVM

Overview SVM

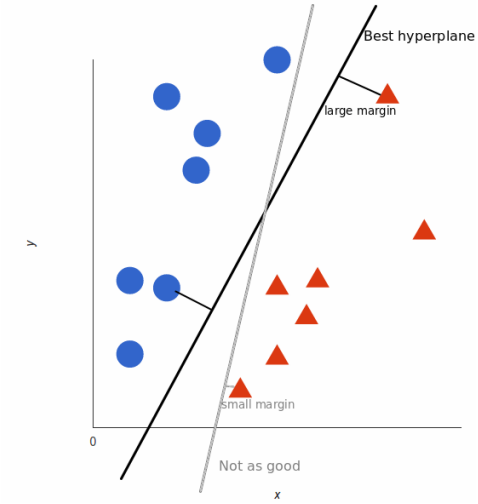
A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousands). This makes the algorithm very suitable for text classification problems, where it's common to have access to a dataset of at most a couple of thousands of tagged samples.

The basics of Support Vector Machines and how it works are best understood with a simple example. Let's imagine we have two tags: *red* and *blue*, and our data has two features: *x* and *y*. We want a classifier that, given a pair of *(x,y)* coordinates, outputs if it's either *red* or *blue*. We plot our already labeled training data on a plane:



A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the **decision boundary**: anything that falls to one side of it we will classify as ***blue***, and anything that falls to the other as ***red***.
But what exactly is ***the best*** hyperplane? For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element of each tag is the largest.



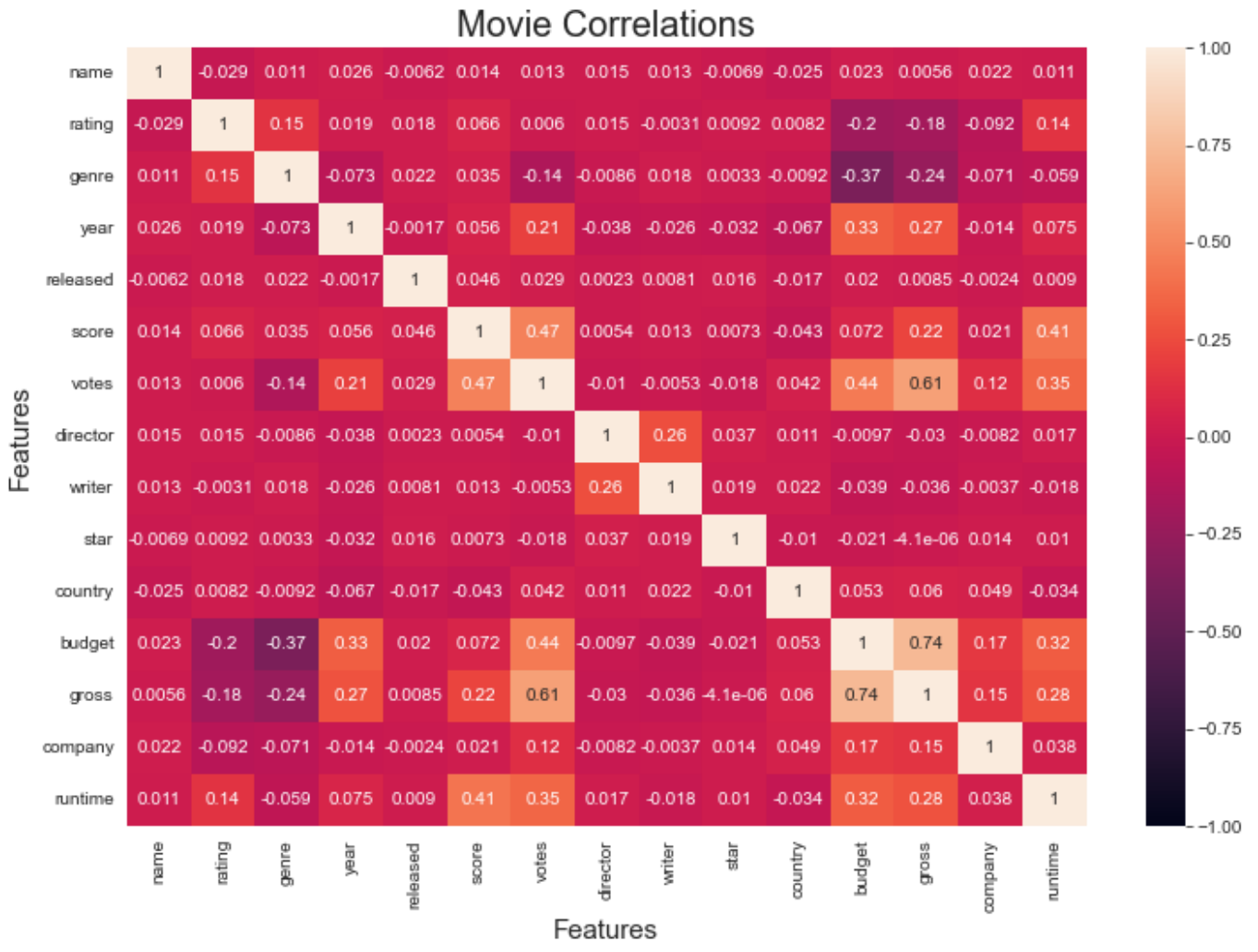
<u>Modelling Data</u>
My data here for Support vector machines was all numeric and high dimensional. I used this predict whether a particular movie would win any type of technical Oscar. My data set had around 600 rows of data with 20 features with an evenly balanced label of whether a movie had an Oscar or not. I had to convert the label outcome to 0's and 1's to implement SVM's.
I implemented SVM's using three kernels linear, sigmoid, polynomial. I mapped out the confusion matrix for the three SVM's.

RESULTS

Let us try to find correlations between various columns in the data set. This task is now easier as much of our data has been converted to numerical format. This can help map out a heatmap with help of seaborn. There are positive correlations and negative correlations. Focusing on positive correlations will help us gain knowledge on which columns in the dataset complement each other and which columns do not complement each other.

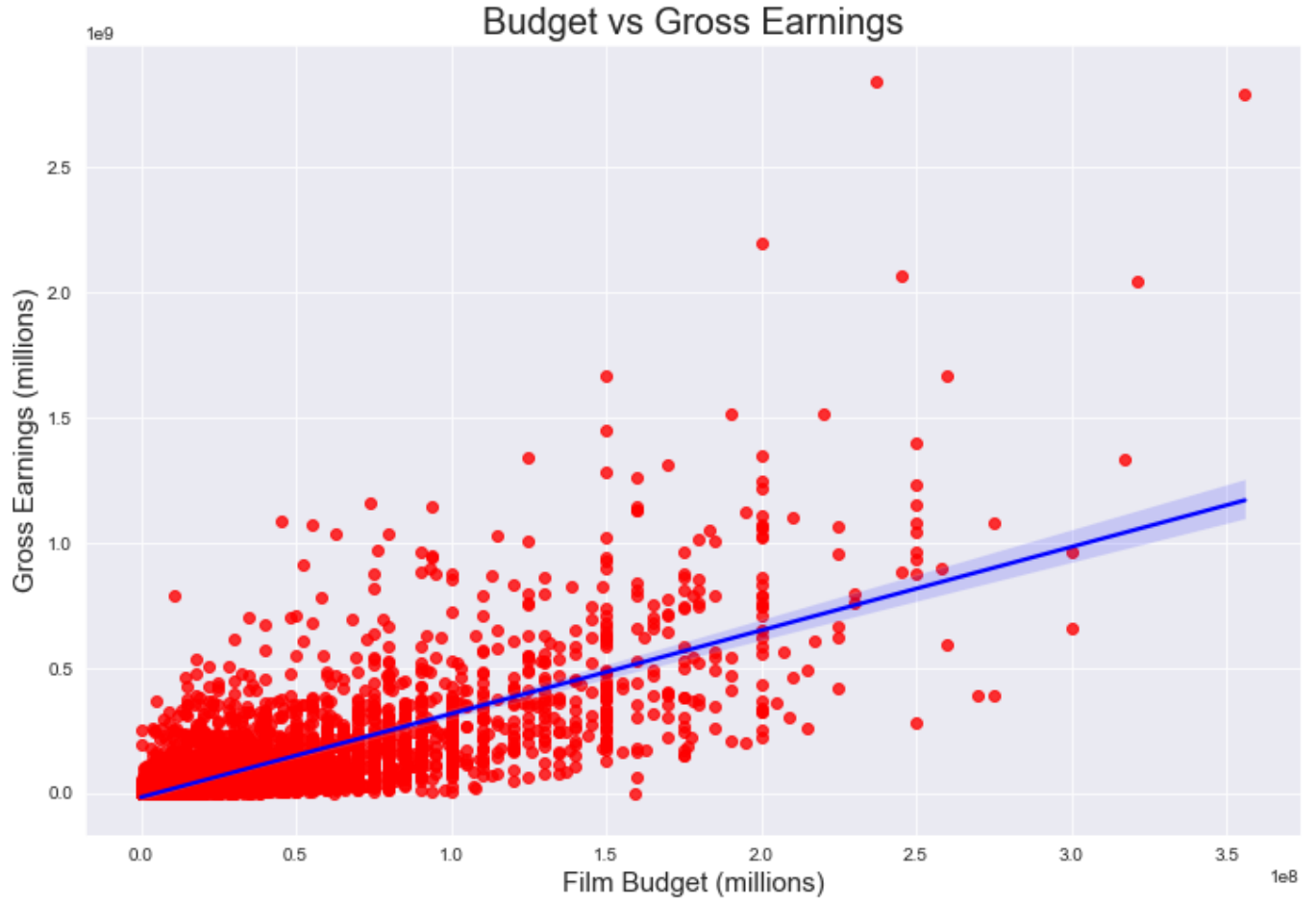
CORRELATION MATRIX

```
In 56  1  #creating a full correlation matrix with all the numeric features
       2  full_corr = df_n.corr()
       3  sns.heatmap(full_corr, annot=True, vmin=-1)
       4  plt.title('Movie Correlations', size=20)
       5  plt.xlabel('Features', size=15)
       6  plt.ylabel('Features', size=15)
       7  plt.show()
```



From our correlation matrix we can clearly see that despite the inclusion of non-numerical features (e.g., company, genre etc.) budget and gross earning are still the most correlated. However, we can also see that votes and gross show a high correlation as well. This is understandable as movies with high gross earnings

would most likely have much more voters than unpopular movies. Budget & gross have a correlation of (0.74) and votes & gross have a correlation of (0.61).



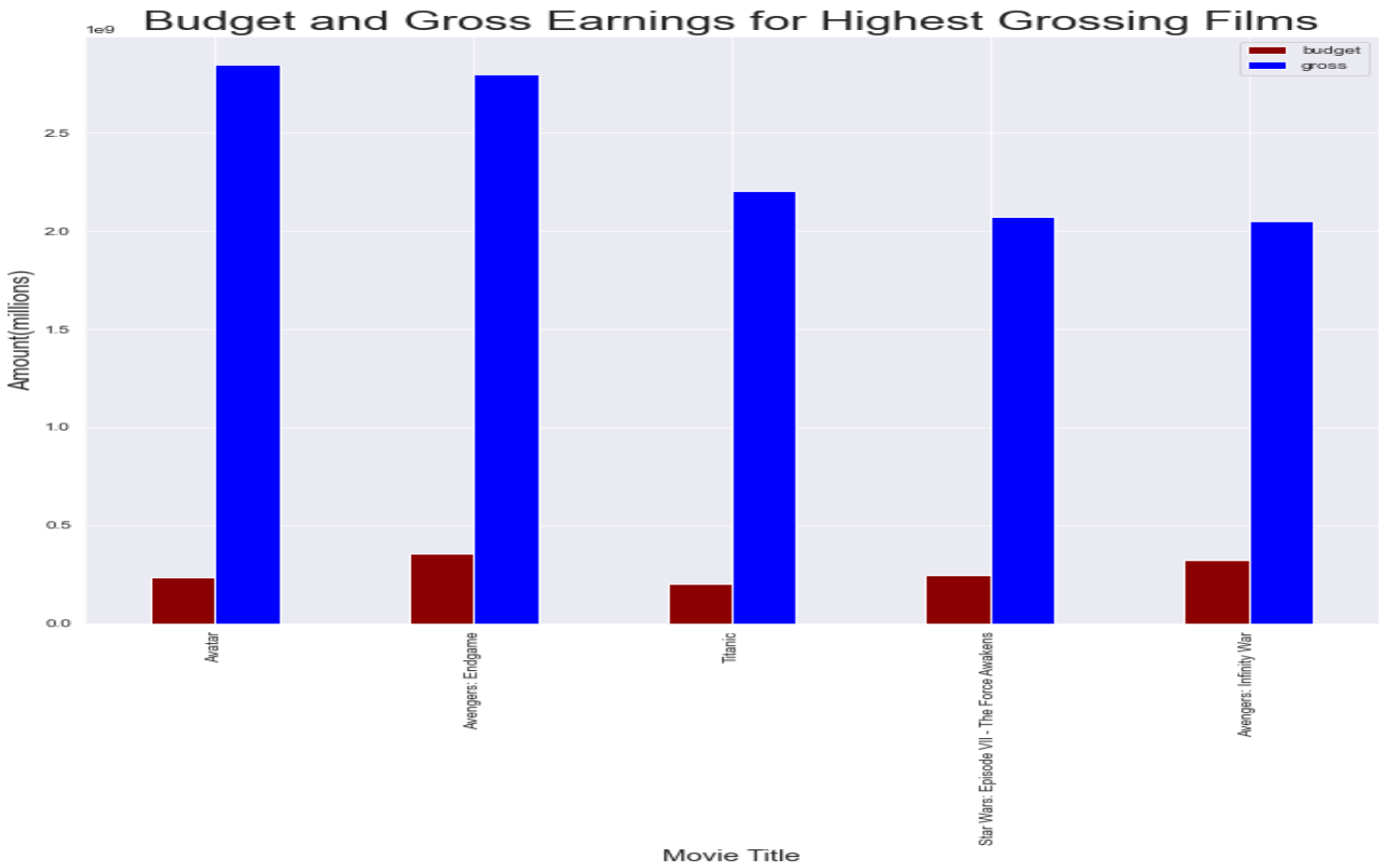
From the scatter plot fitted with regression line we can clearly see that budget and gross earnings are positively correlated.

Now, let's look at the following trends in the dataset.

- Budget vs gross earnings for highest grossing films
- Budget vs gross earning for highest grossing companies
- Budget vs gross earning for genres
- Writer vs number of movies
- Stars vs number of movies
- Gross earning for highest rated movies

BUDGET VS GROSS EARNINGS FOR FILMS

```
In 73  1  df_top5.plot(kind='bar', x='name', y=['budget','gross'], color={'budget':'darkred', 'gross':'blue'}, figsize=(12,10))
       2  plt.title('Budget and Gross Earnings for Highest Grossing Films', size=25)
       3  plt.xlabel('Movie Title', size=15)
       4  plt.ylabel('Amount(millions)', size=15)
       5  plt.show()
```



These are the top 5 most profitable movies released till date. Their gross earning is 5 times their budget which signifies that these movies were a huge success in the box office and captured audience's interest worldwide.

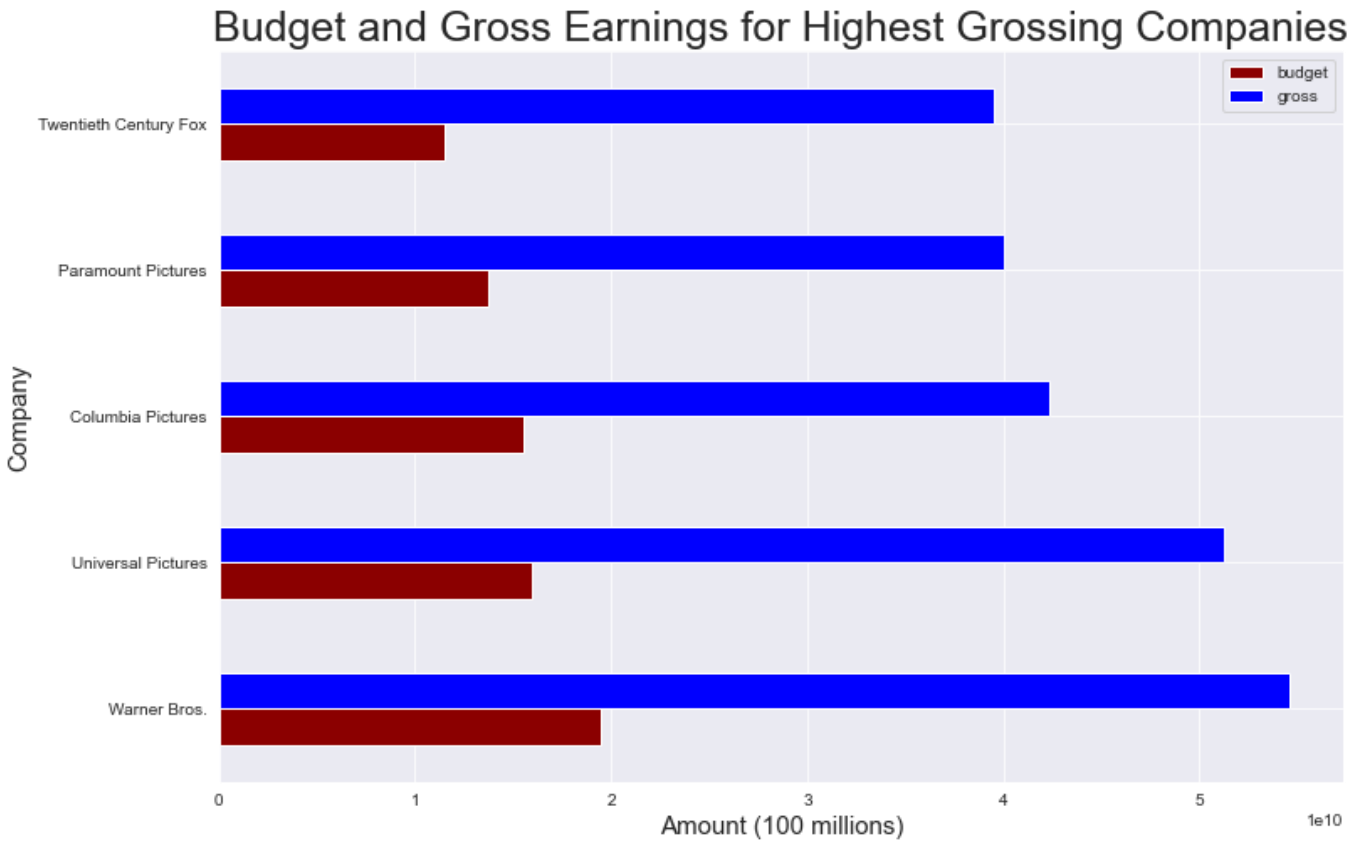
BUDGET VS GROSS EARNINGS FOR COMPANIES

In 77
```
companies = df.groupby(['company'], as_index=False)['budget', 'gross'].sum().sort_values(by='gross', ascending=False)
top5_companies = companies.head()
print(top5_companies)
```

```
                company       budget        gross
1426          Warner Bros.  19503300000  54610959970
1397     Universal Pictures  15989730001  51241105418
452       Columbia Pictures  15512107000  42356430218
1108     Paramount Pictures  13723450000  40021704691
1382  Twentieth Century Fox  11474600000  39542573303
```

```
C:\Users\manob\AppData\Local\Temp\ipykernel_9388\90982053.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  companies = df.groupby(['company'], as_index=False)['budget', 'gross'].sum().sort_values(by='gross', ascending=False)
```

In 78
```
top5_companies.plot(kind='barh', x='company', y=['budget', 'gross'],color={'budget':'darkred', 'gross':'blue'}, figsize=(12,8))
plt.title('Budget and Gross Earnings for Highest Grossing Companies', size=25)
plt.xlabel('Amount (100 millions)', size=15)
plt.ylabel('Company', size=15)
plt.show()
```



This graph is generated with the group_by menthod with budget and gross as the main grouping variables and is sorted to descending order. The graph is a bar graph with companies on y asix and amount in x axis.From the graph we can notice companies and their profits. Companies have a lot less budget compared to their gross earnings and they are hugely profitable in general when it comes to movies. These are the top 5 companies in the dataset.

GENRE VS GROSS EARNINGS

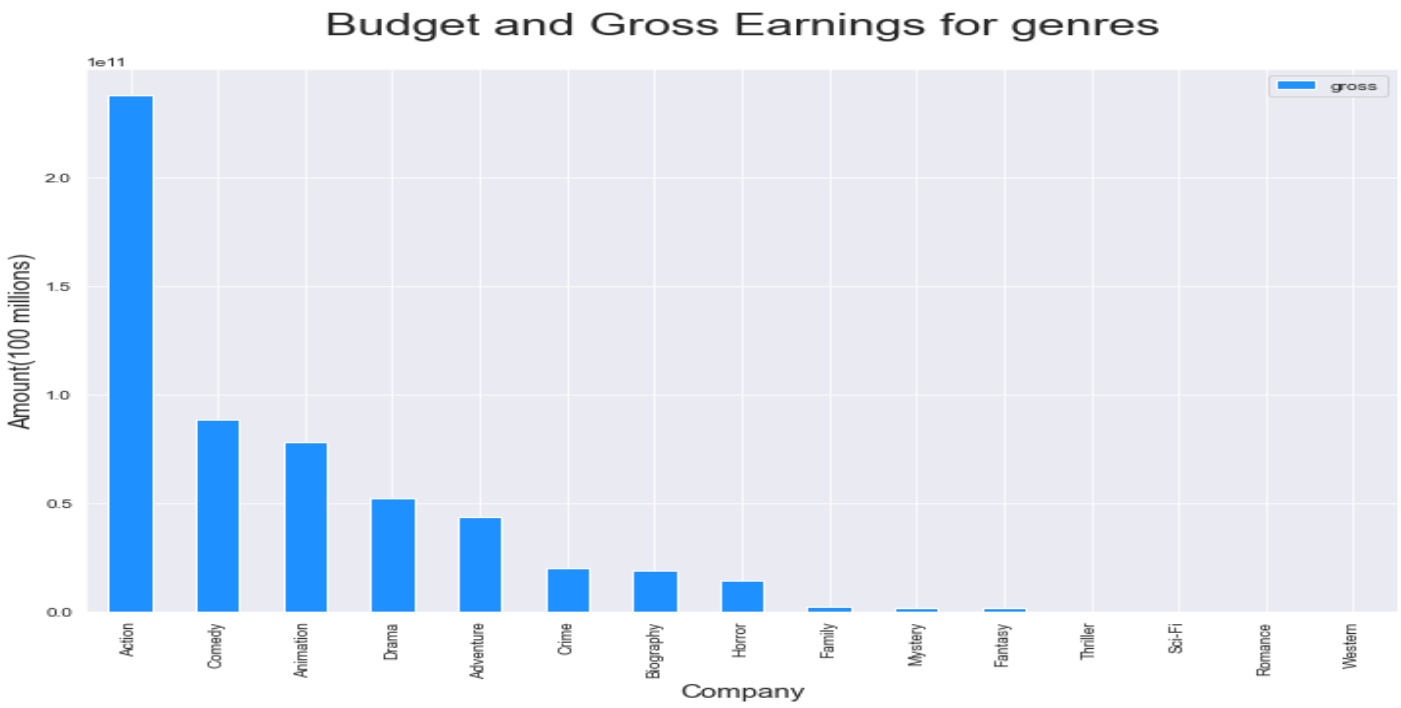
```
In 36
genre_gross = df.groupby(['genre'], as_index=False)['gross'].sum().sort_values(by='gross', ascending=False)
genre_gross.reset_index()
genre_gross
```

Out 36

	genre	gross
6	Drama	52098564769
1	Adventure	43578711908
5	Crime	20017662162
3	Biography	19093930296
9	Horror	14261055032
7	Family	2074332587
10	Mystery	2004091467
8	Fantasy	1635026609

15 rows × 2 columns Open in new tab

```
In 152
genre_gross.plot(kind='bar', x='genre', y='gross', color='dodgerblue', figsize=(12,8))
plt.title('Budget and Gross Earnings for genres', size=25, y=1.05)
plt.xlabel('Company', size=15)
plt.ylabel('Amount(100 millions)', size=15)
plt.show()
```



This graph is generated with the group_by menthod with genre as the main grouping variable and is sorted to descending order. The graph is a bar graph with gross earnings on y asix and genres in x axis.From the graph we can notice which genres have the highest grossing. Action genres seem to have to have the highest grossing followed by comedy,animation and dramas etc,.

WRITERS VS NUMBER OF MOVIES

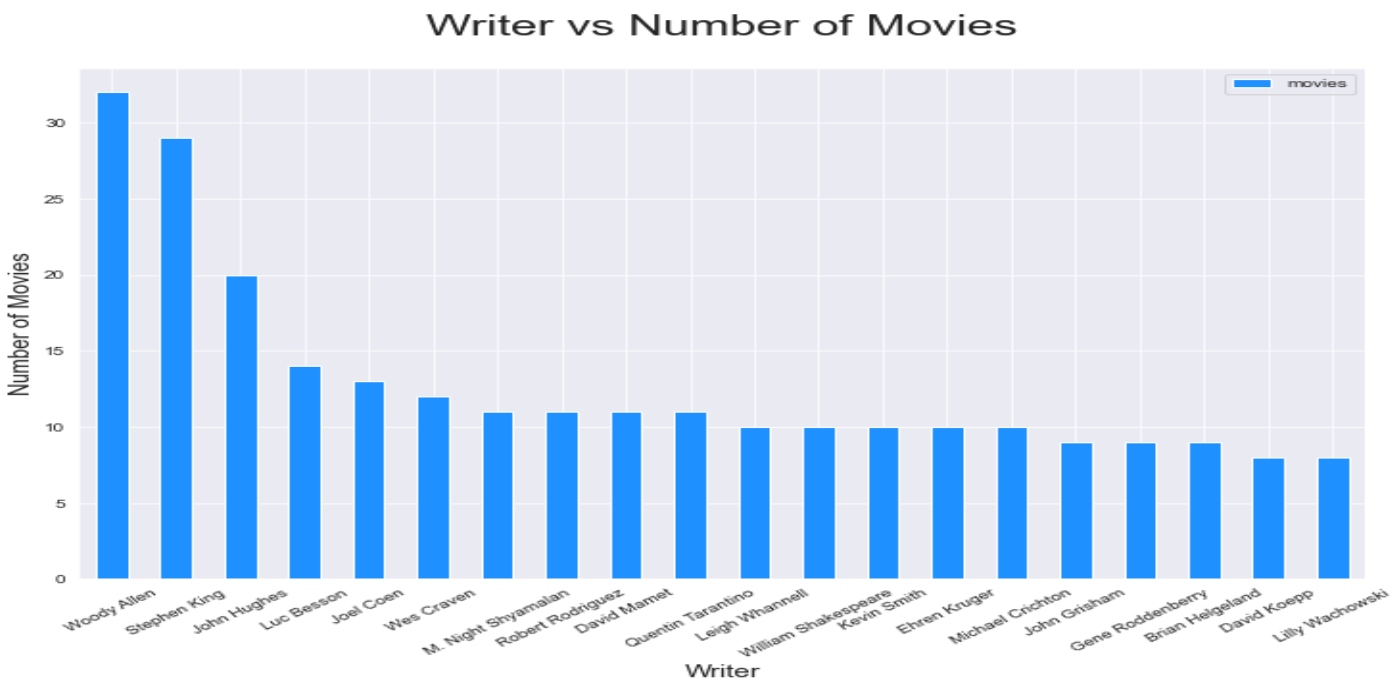
In 40
```
writers = df.groupby(['writer'], as_index=False)['name'].count().sort_values(by='name', ascending=False)
writers.rename(columns={'name':'movies'}, inplace=True)
top_writers = writers.head(20)
top_writers
```

Out 40

	writer	movies
1898	M. Night Shyamalan	11
2601	Robert Rodriguez	11
706	David Mamet	11
2447	Quentin Tarantino	11
1821	Leigh Whannell	10
3175	William Shakespeare	10
1747	Kevin Smith	10
886	Ehren Kruger	10

20 rows × 2 columns Open in new tab

In 41
```
top_writers.plot(kind='bar', x='writer', y='movies', color='dodgerblue', figsize=(12,8))
plt.title('Writer vs Number of Movies', size=25, y=1.05)
plt.xlabel('Writer', size=15)
plt.xticks(rotation=35)
plt.ylabel('Number of Movies', size=15)
plt.show()
```



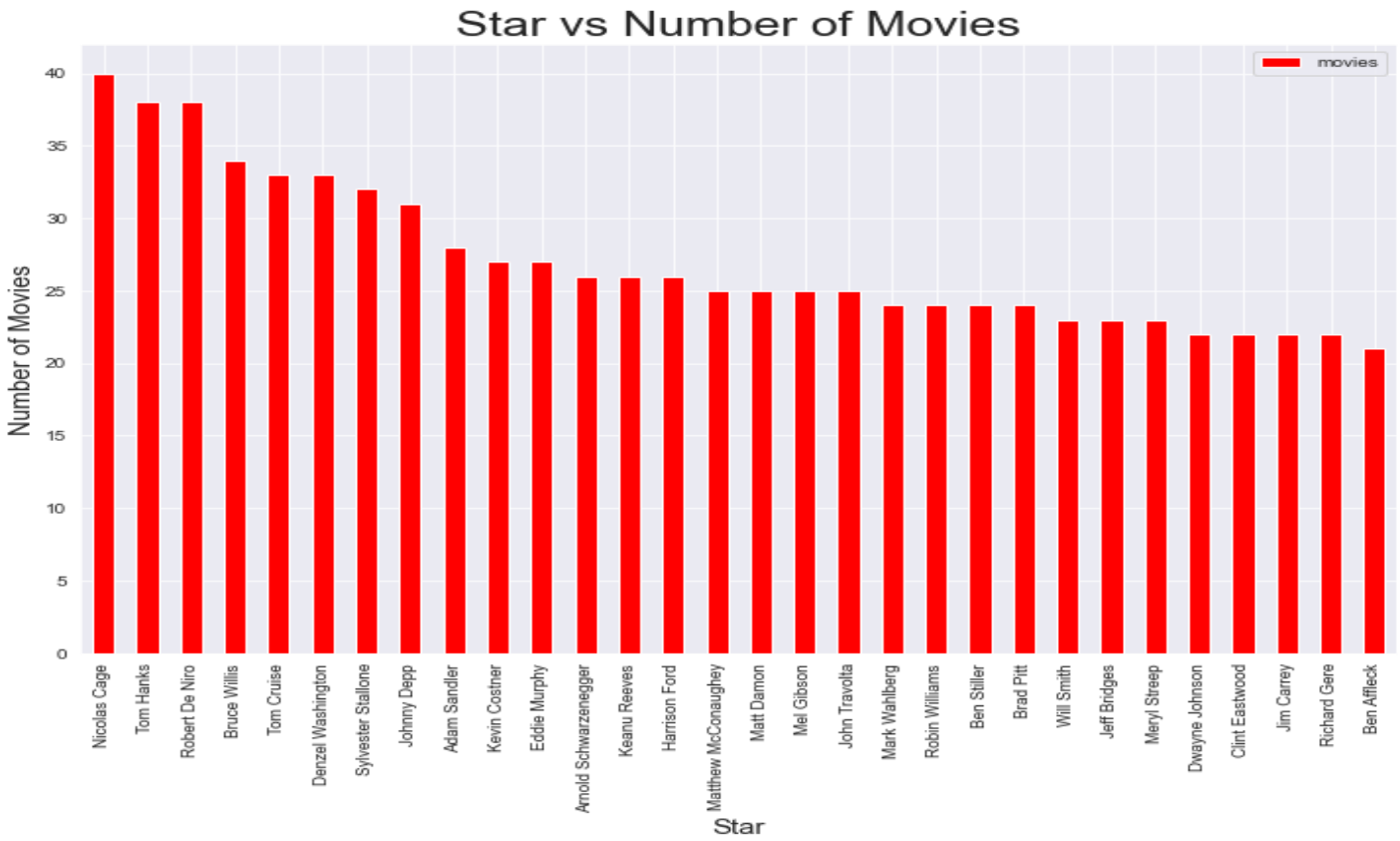
This is graph is generated by group_by method and the sorted by the number of times a particular writer appears in the dataset. We can plot writers on x-axis and the numbers of movies they have written on the y-axis. We can see woody allen has the most number of appearances followed by stephen king etc,.

STARS VS NUMBER OF MOVIES

In 123
```
stars=df.groupby(['star'], as_index=False)['name'].count().sort_values(by='name',ascending=False)
stars.rename(columns={'name':'movies'},inplace=True)
top_stars=stars.head(30)
print(top_stars)
```

```
                    star  movies
1313         Nicolas Cage      40
1735            Tom Hanks      38
1469       Robert De Niro      38
220          Bruce Willis      34
1732          Tom Cruise      33
436     Denzel Washington      33
1665   Sylvester Stallone      32
889           Johnny Depp      31
13            Adam Sandler      28
1011        Kevin Costner      27
```

In 115
```
top_stars.plot(kind='bar', x='star', y='movies', color='red', figsize=(12,8))
plt.title('Star vs Number of Movies', size=25)
plt.xlabel('Star', size=15)
plt.ylabel('Number of Movies', size=15)
plt.show()
```



This is graph is generated by group_by method and the sorted by the number of times a particular star appears in the dataset. We can plot stars on x-axis and the numbers of movies they appeared on the y-axis. We can see nicolas cage has the most number of appearances followed by tom hanks etc,.

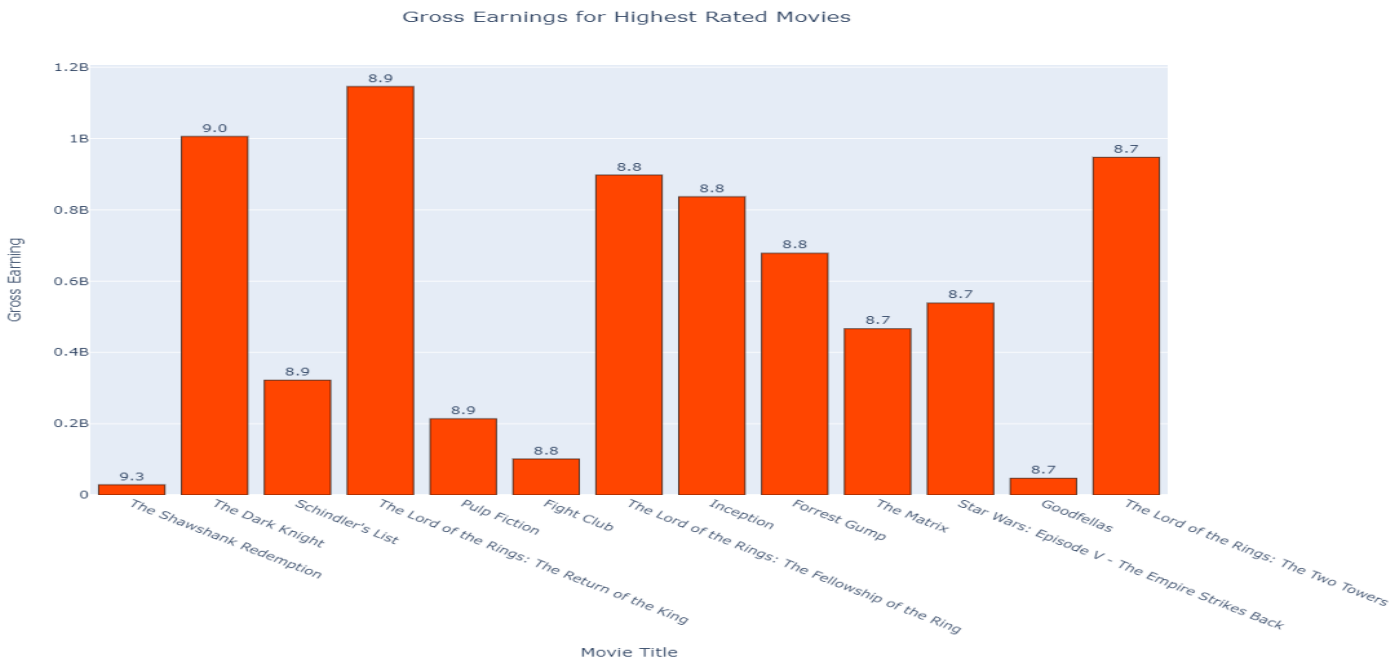
GROSS EARNINS FOR HIGHEST RATED MOVIES

In 130
```
df_scoring=df.sort_values(by='score',ascending=False)
df_scoring=df_scoring[df_scoring['score']>8.6]
print(df_scoring)
```

```
                                          name rating      genre  \
2443                       The Shawshank Redemption      R      Drama
5243                                The Dark Knight  PG-13     Action
2247                                Schindler's List      R  Biography
4245     The Lord of the Rings: The Return of the King  PG-13     Action
2444                                   Pulp Fiction      R      Crime
3444                                     Fight Club      R      Drama
3845  The Lord of the Rings: The Fellowship of the Ring  PG-13     Action
5643                                      Inception  PG-13     Action
2445                                   Forrest Gump  PG-13      Drama
3443                                     The Matrix      R     Action
```

In 144
```
import plotly.express as px

fig = px.bar(df_scoring,
             x='name',
             y='gross',
             title='Gross Earnings for Highest Rated Movies',
             text='score',
             width=1100,
             height=800)
fig.update_layout(xaxis_title='Movie Title', yaxis_title='Gross Earning', title_x = 0.45)
fig.update_traces(texttemplate='%{text:.2s}', textposition='outside', hovertext='score')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
```

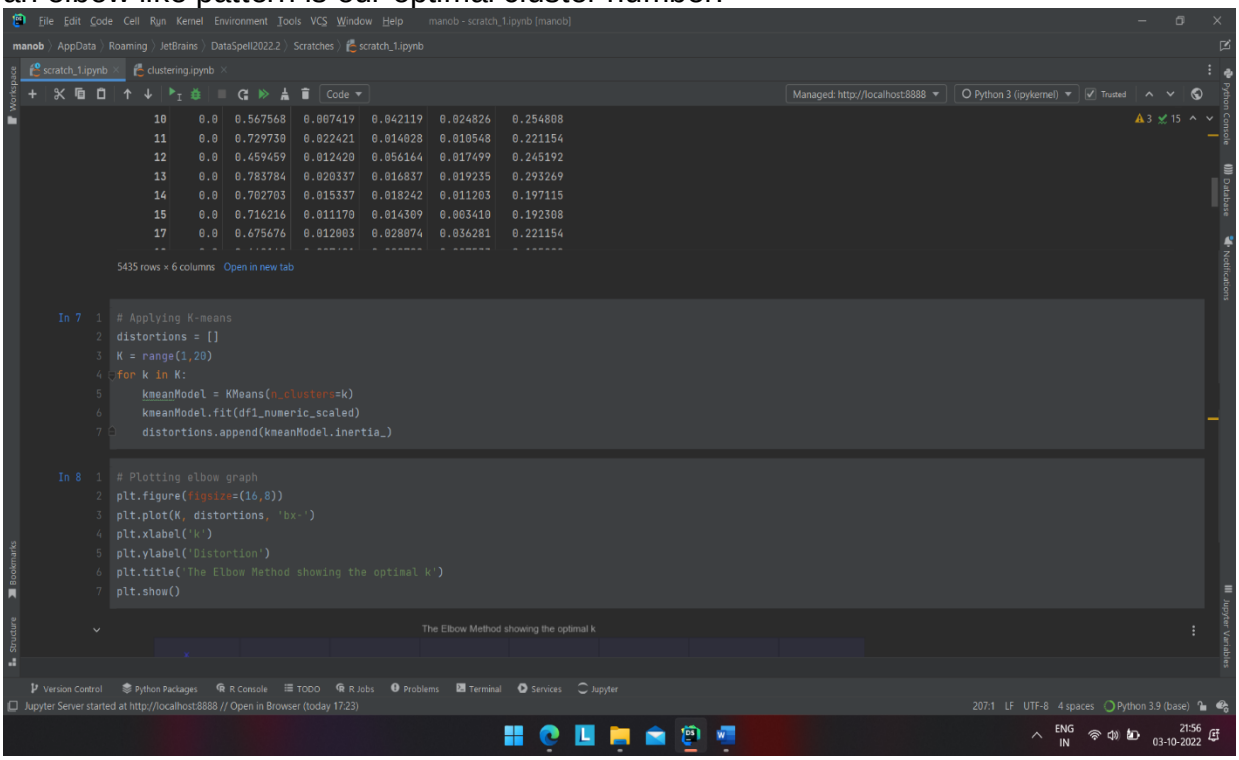


It not assumed that the highest rated movie has the highest gross earnings. We can see that shawshank redemption which is rated a staggering 9.3 has not earned that much compared to the dark knight which has earned around 1.0B. This graphs shows that the higher the rating does not correspond to higher gross.

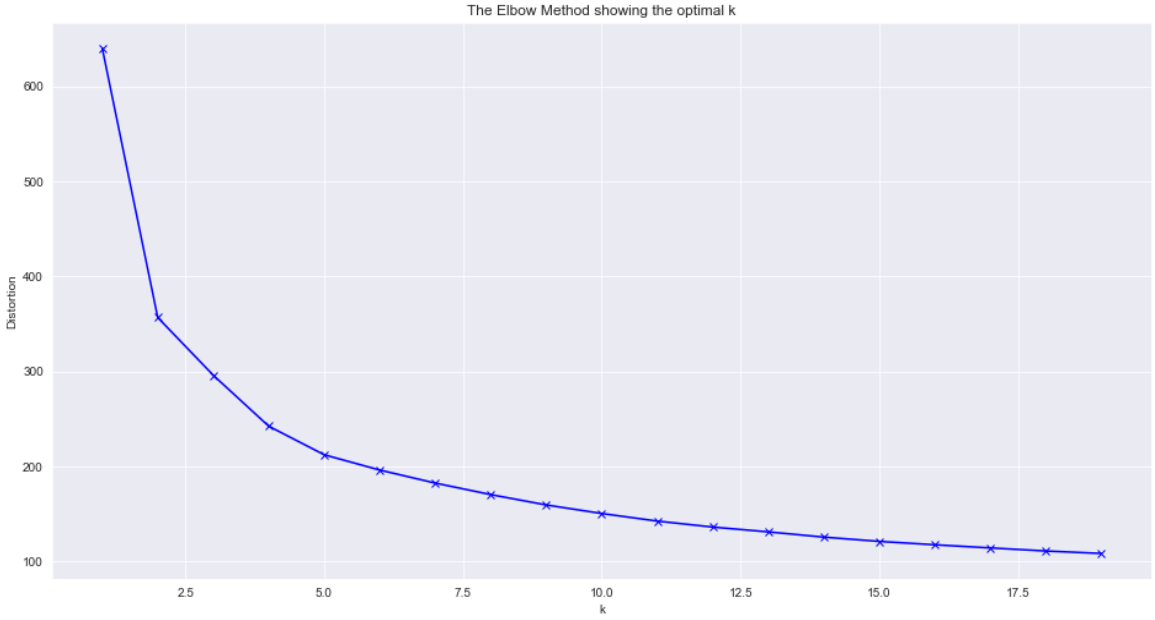
RESULTS: CLUSTERING

K-MEANS CLUSTERING

We can apply the elbow method to properly determine how many clusters are suitable for our data. The elbow method can be applied to the data by considering inertia. The point where we see an elbow like pattern is our optimal cluster number.

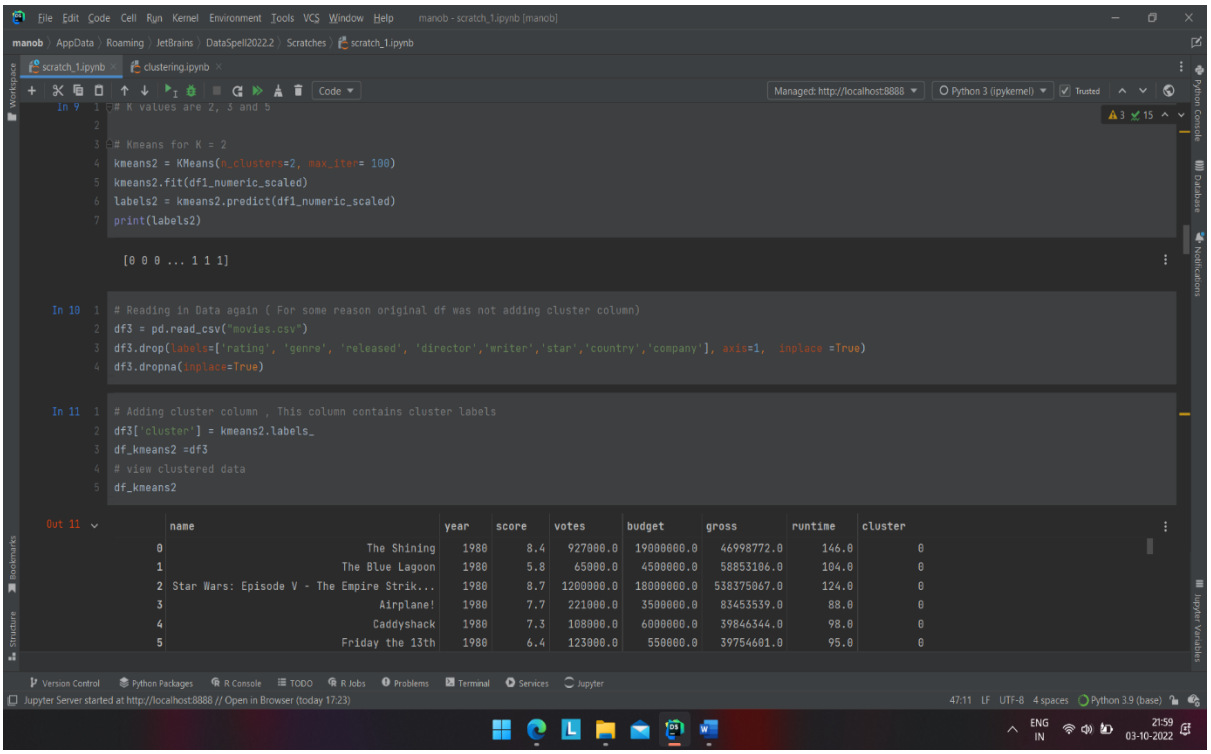


The elbow approach is a heuristic for figuring out how many clusters there are in a data collection. Plotting the explained variation as a function of the number of clusters, the procedure entails choosing the elbow of the curve as the appropriate number of clusters. The number of parameters in other data-driven models, such as the number of primary components to describe a data collection, can be chosen using the same methodology.



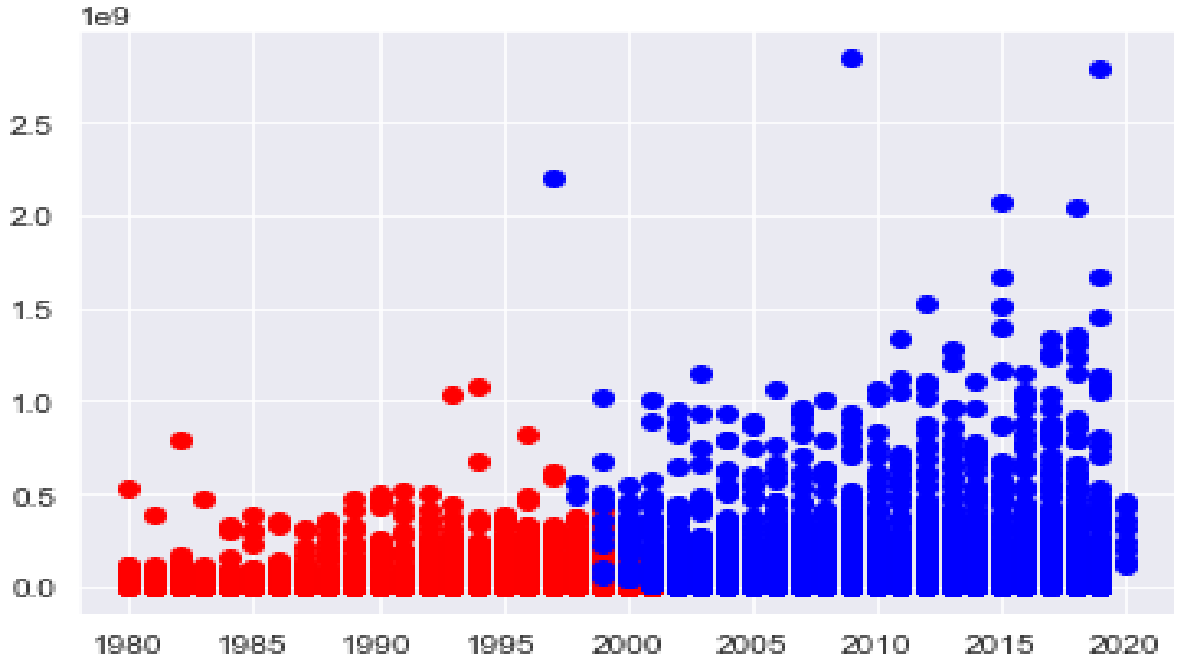
From the graph we can see the k=2 is the best cluster.
Let us just plot clustering plots for K=2, K=3, K=5.

1) K=2



Clustered data for K=2 and added cluster label to data frame for better understanding.

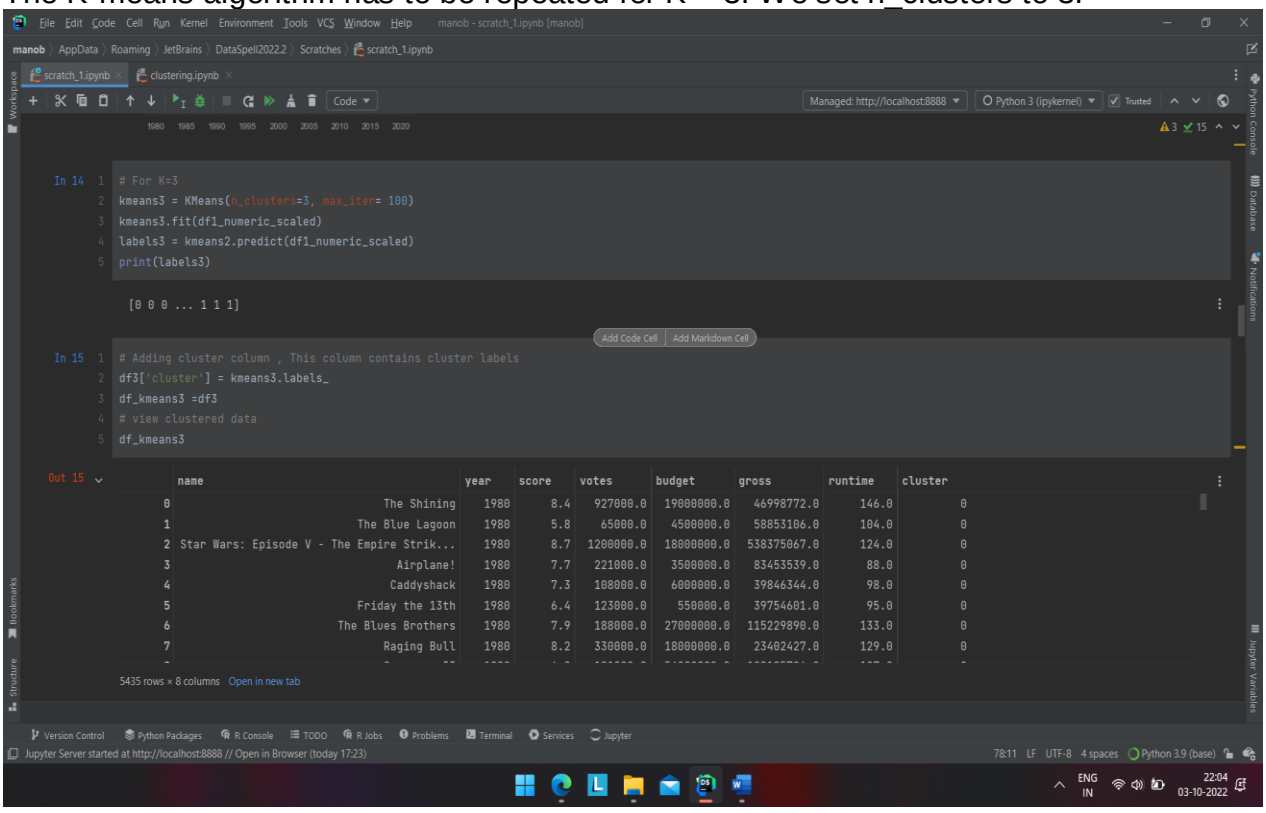
Plotting the clustered data for K=2



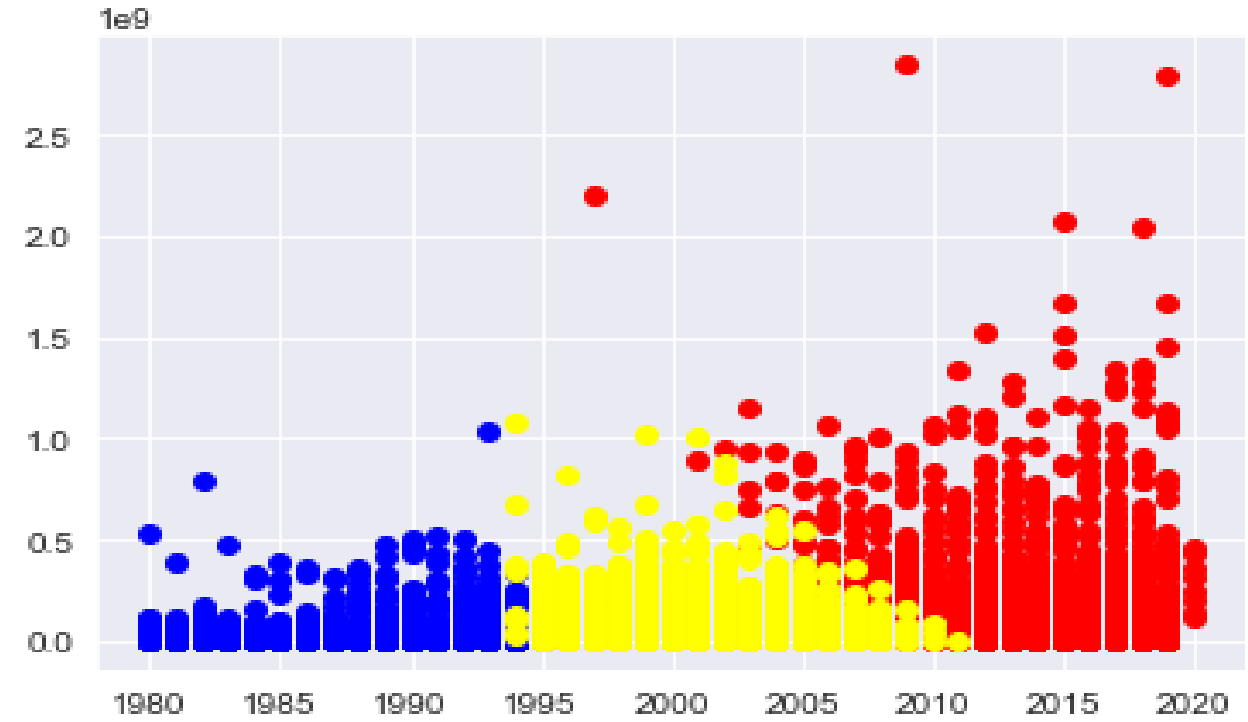
The data has been clustered for K=2.

2) K=3

The K-means algorithm has to be repeated for K = 3. We set n_clusters to 3.



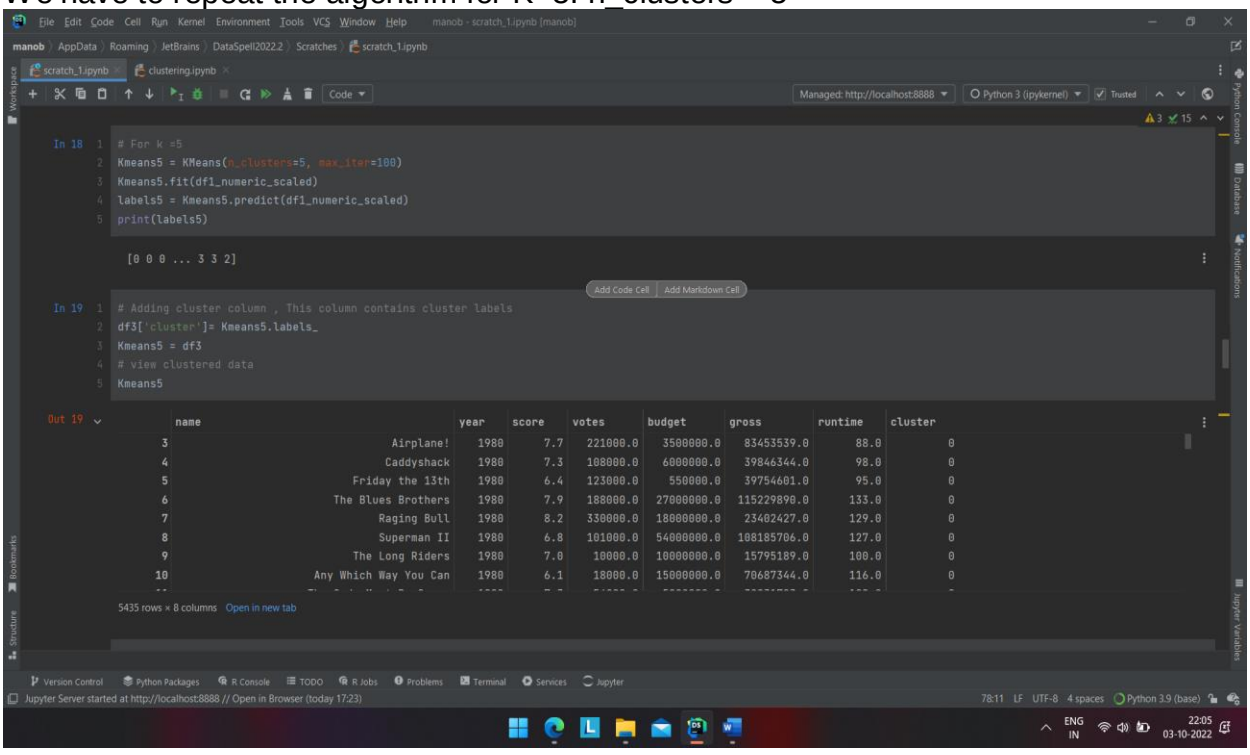
Plotting clustered Data for K=3



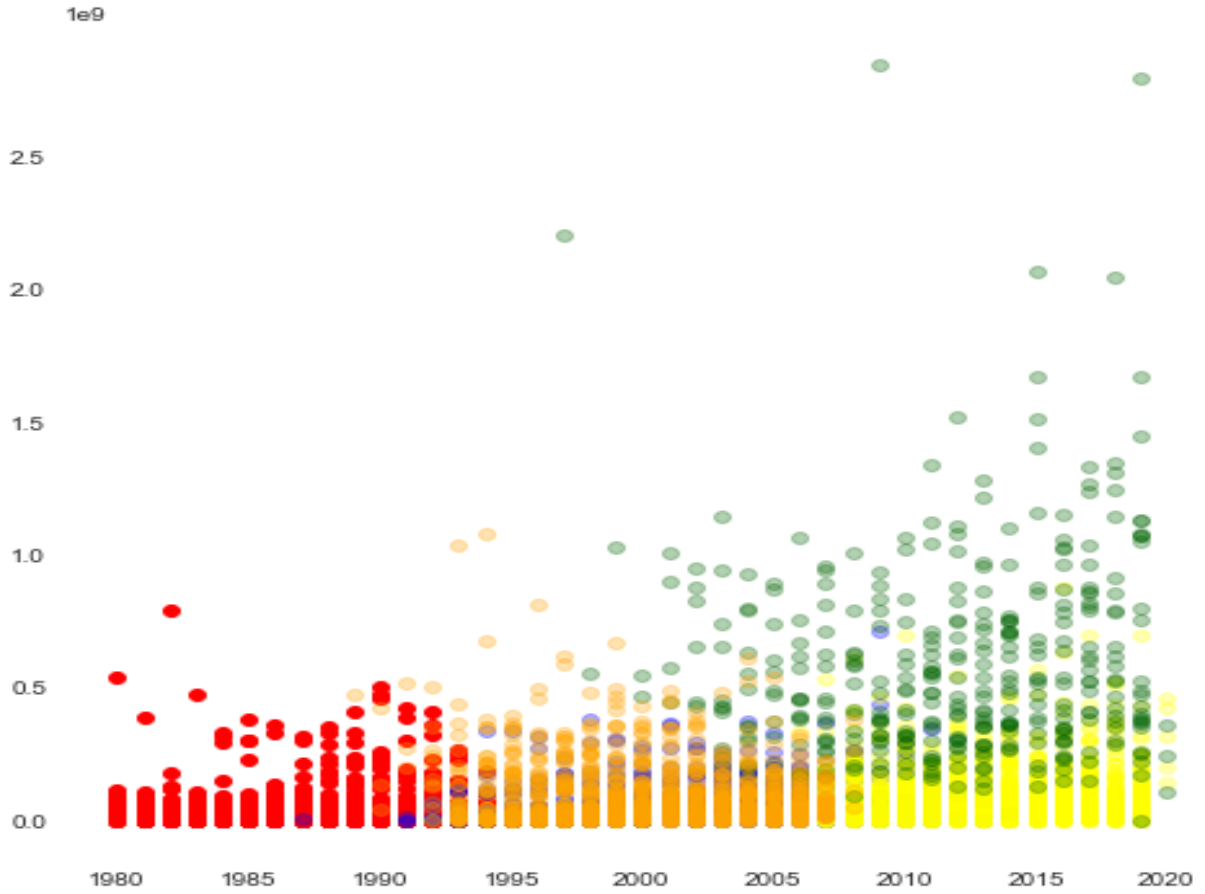
Data has been clustered for K=3. Clustering looks pretty good.

3) K=5

We have to repeat the algorithm for K=5. n_clusters = 5



Plotting Data for K=5



There is a lot of overlap between clusters and the clustering looks bad. It is not optimal. The elbow graph method has already plotted us optimal clusters. The elbow graph method is a fine measure to understand optimal clusters for data.

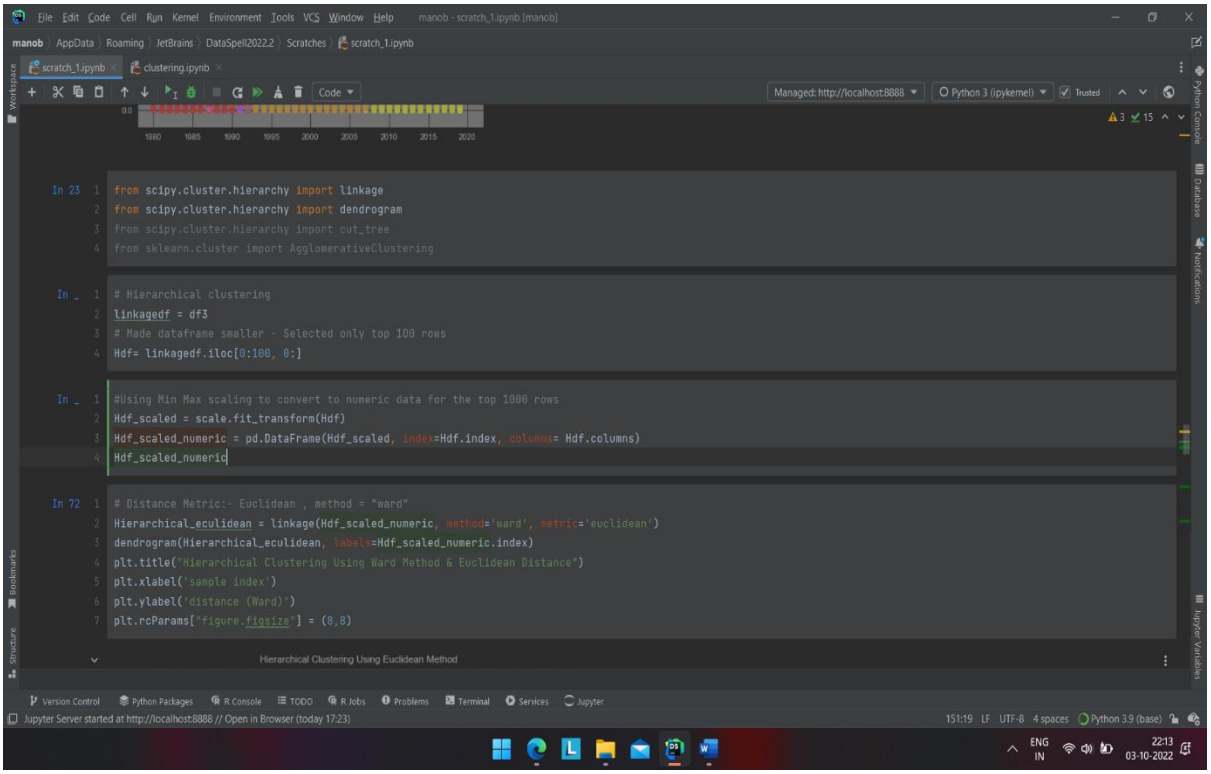
HIERARCHICAL CLUSTERING

For this clustering I reduced my data set from 6000 rows to just 100 rows and rescaled it accordingly. This had to be done as there were too clusters forming, and the clustering was getting complex.

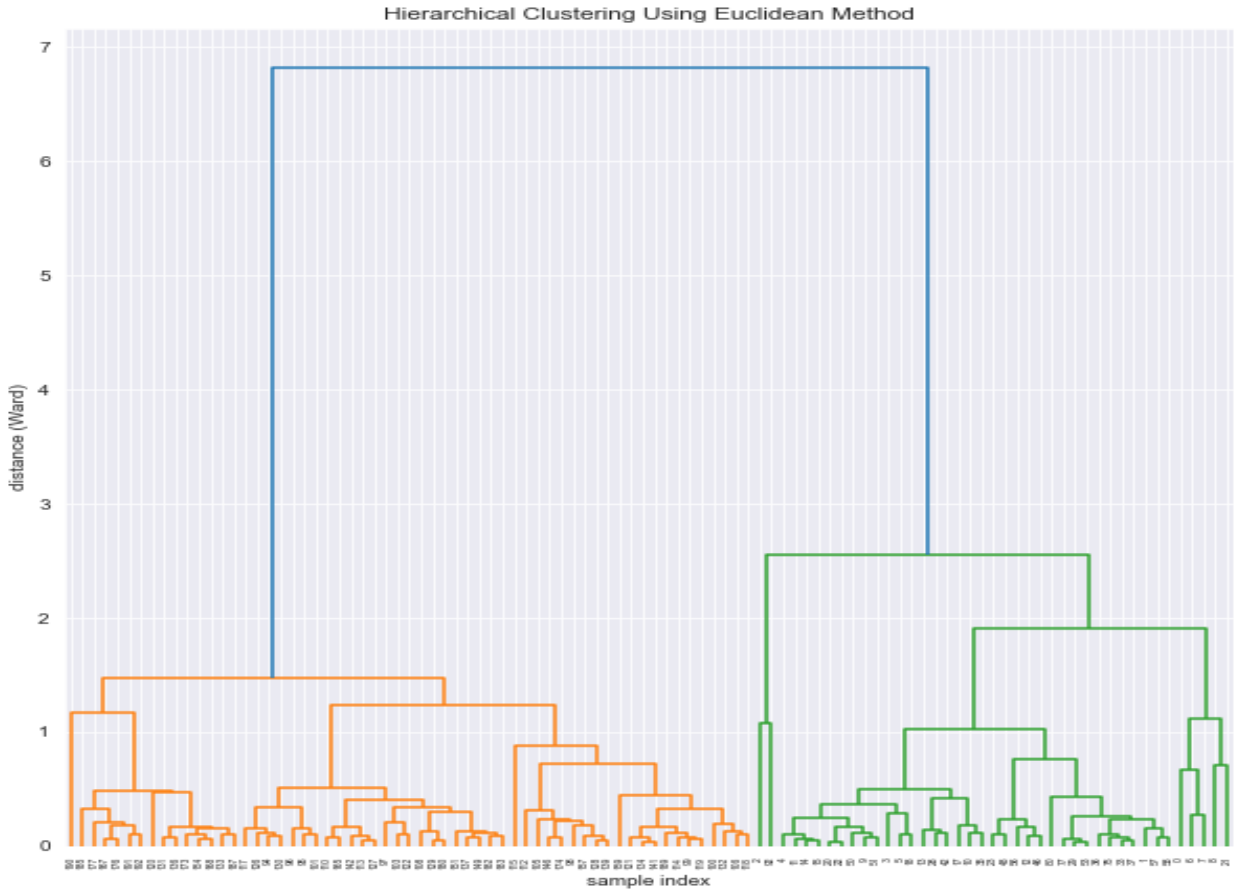
The Data prepping for this clustering method is similar to K-means clustering. We use a different library to implement this clustering method. We use the linkage library and the dendrogram library to successfully implement this clustering method. There are many methods by which hierarchical clustering can be implemented but due to the sake of time and document length only of the methods and distance metrics will be discussed here.

We showcase the ward's , complete and average methods and use Euclidean, City block and cosine sim as distance metrics for our respective methods. The code and the visuals have been pasted below.

I found this clustering method very unique, but I will only use it for specific purposes. Sticking to K-means seems like the best way to go with unsupervised machine learning.



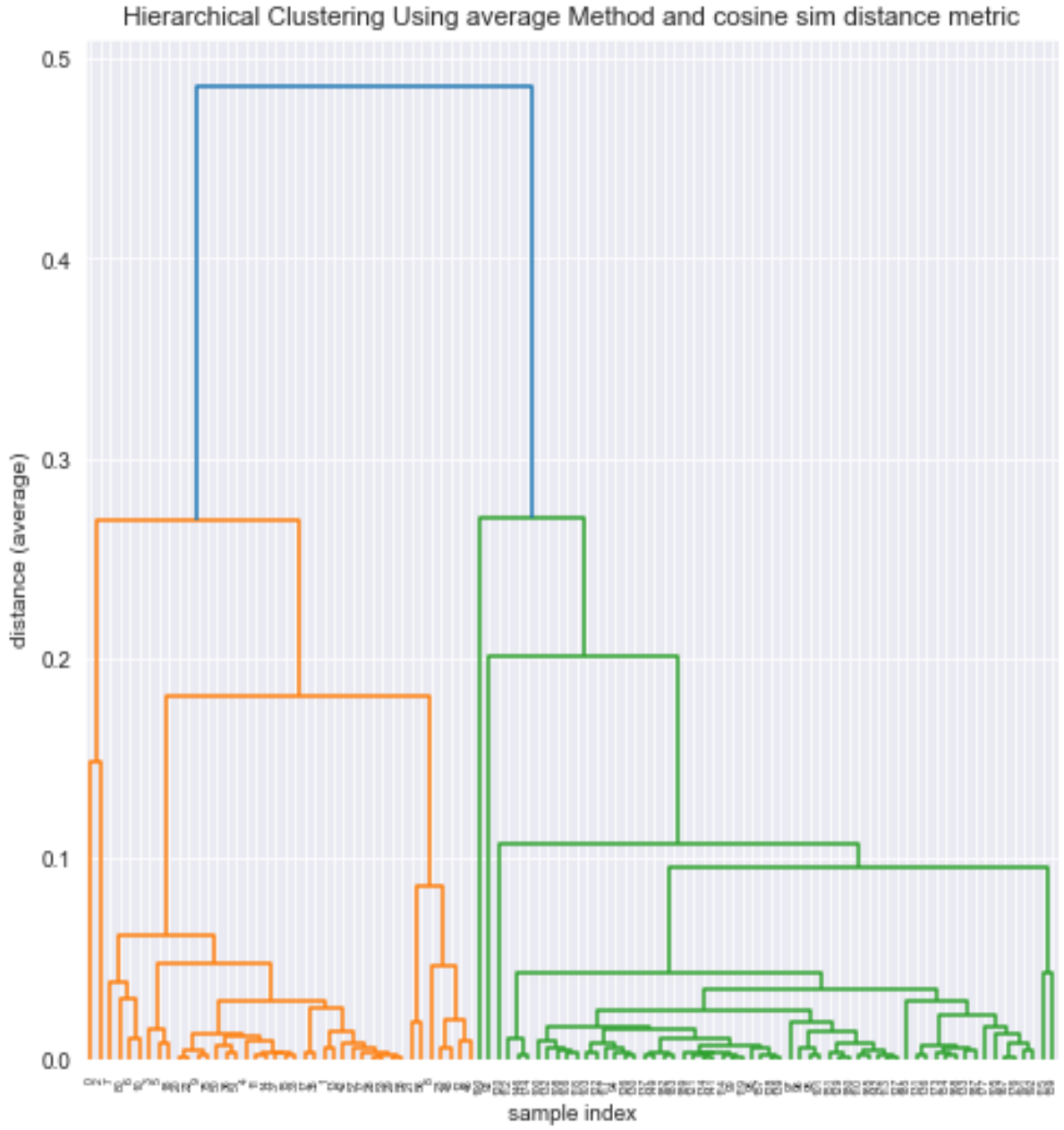
“Euclidean Distance” Clustering with “Ward’s” Method



Hierarchical Clustering using “average” method and “cosine sim” distance metric.

```
# Distance Metric :- Cosine Sim , Method = "average"
# Ward only works with eculidean
Hierarchical_cosine = linkage(Hdf_scaled_numeric, method='average', metric= 'cosine')
dendrogram(Hierarchical_cosine, labels=Hdf_scaled_numeric.index)
plt.title("Hierarchical Clustering Using average Method and cosine sim distance metric")
plt.xlabel('sample index')
plt.ylabel('distance (average)')
plt.rcParams["figure.figsize"] = (8,8)
```

Plotting Clusters

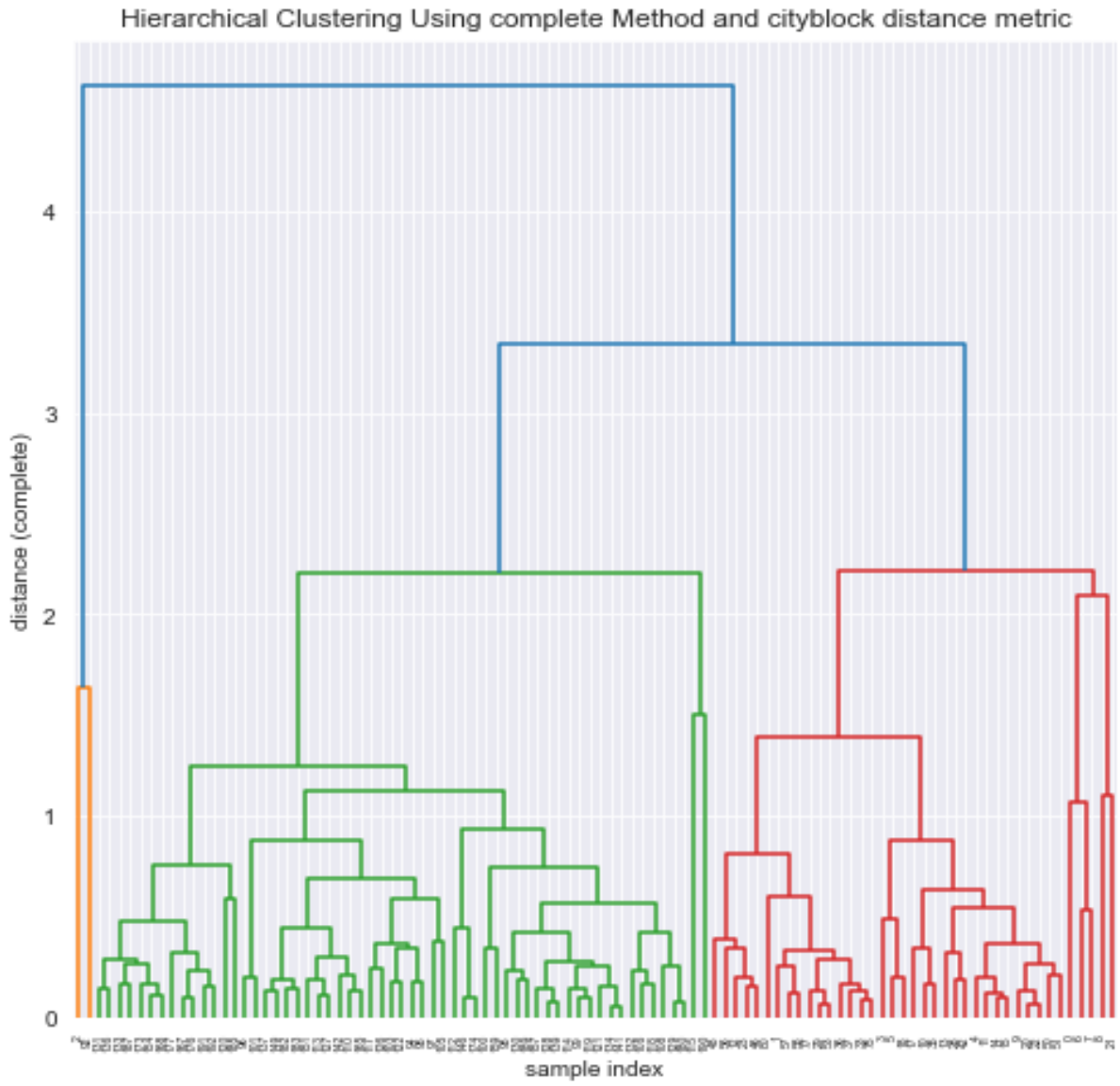


Clustering looks good and sharp.

Hierarchical Clustering using “complete” method and “cityblock” distance metric.

```
# Distance Metric :- cityblock , Method = "complete"
# distance.pdist to check various distance metrics available
Hierarchical_cityblock = linkage(Hdf_scaled_numeric, method='complete', metric= 'cityblock')
dendrogram(Hierarchical_cityblock, labels=Hdf_scaled_numeric.index)
plt.title("Hierarchical Clustering Using complete Method and cityblock distance metric")
plt.xlabel('sample index')
plt.ylabel('distance (complete)')
plt.rcParams["figure.figsize"] = (8,8)
```

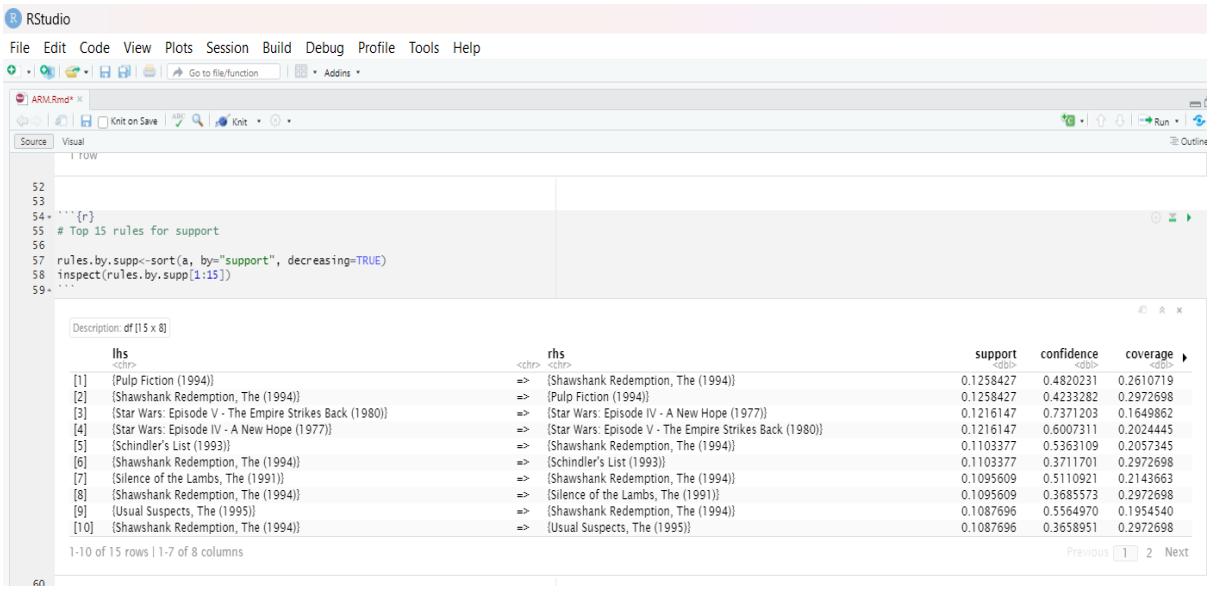
Plotting clustering



RESULTS: ARM

Association rules are helpful in data mining for studying and forecasting consumer behaviour. Customer analytics, market basket analysis, product clustering, catalog design, and retail layout all benefit from their use. Association rules are used by programmers to create machine learning-capable software.
The Support, Confidence and lift for movie data has been coded and displayed below.

Let us view the top 15 rules for Support, Confidence and Lift.

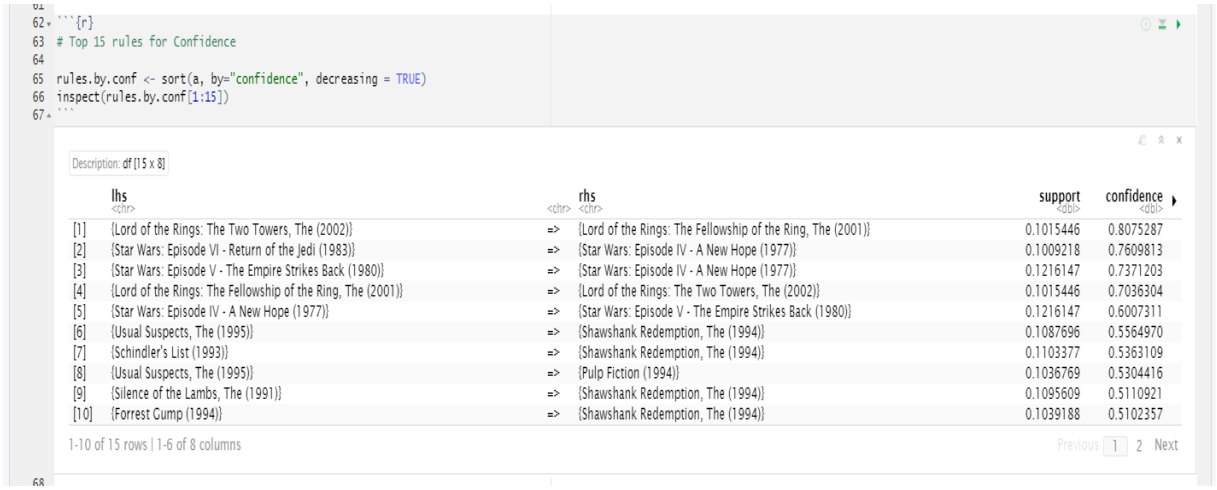


	lhs		rhs	support	confidence	coverage
[1]	{Pulp Fiction (1994)}	=>	{Shawshank Redemption, The (1994)}	0.1258427	0.4820231	0.2610719
[2]	{Shawshank Redemption, The (1994)}	=>	{Pulp Fiction (1994)}	0.1258427	0.4233282	0.2972698
[3]	{Star Wars: Episode V - The Empire Strikes Back (1980)}	=>	{Star Wars: Episode IV - A New Hope (1977)}	0.1216147	0.7371203	0.1649862
[4]	{Star Wars: Episode IV - A New Hope (1977)}	=>	{Star Wars: Episode V - The Empire Strikes Back (1980)}	0.1216147	0.6007311	0.2024445
[5]	{Schindler's List (1993)}	=>	{Shawshank Redemption, The (1994)}	0.1103377	0.5363109	0.2057345
[6]	{Shawshank Redemption, The (1994)}	=>	{Schindler's List (1993)}	0.1103377	0.3711701	0.2972698
[7]	{Silence of the Lambs, The (1991)}	=>	{Shawshank Redemption, The (1994)}	0.1095609	0.5110921	0.2143663
[8]	{Shawshank Redemption, The (1994)}	=>	{Silence of the Lambs, The (1991)}	0.1095609	0.3685573	0.2972698
[9]	{Usual Suspects, The (1995)}	=>	{Shawshank Redemption, The (1994)}	0.1087696	0.5564970	0.1954540
[10]	{Shawshank Redemption, The (1994)}	=>	{Usual Suspects, The (1995)}	0.1087696	0.3658951	0.2972698

1-10 of 15 rows | 1-7 of 8 columns Previous 1 2 Next

The rules with highest support have been displayed. Pulp fiction -> Shawshank redemption has the highest support. It means that this transaction is the most frequently occurring one in our dataset. We can the follow up with Star Wars , silence of the lambs , unusual suspects and so on.

The rules with highest confidence have been displayed below. We can notice that people who watched lord of the rings the two towers are most likely to watch the fellowship of the ring. It makes sense also because you would usually end up watching all the movies in any trilogy to follow the story. The next is Star Wars is also has very good confidence and it also a franchise. We can see that a popular movie has high confidence with other popular movies.



	lhs		rhs	support	confidence
[1]	{Lord of the Rings: The Two Towers, The (2002)}	=>	{Lord of the Rings: The Fellowship of the Ring, The (2001)}	0.1015446	0.8075287
[2]	{Star Wars: Episode VI - Return of the Jedi (1983)}	=>	{Star Wars: Episode IV - A New Hope (1977)}	0.1009218	0.7609813
[3]	{Star Wars: Episode V - The Empire Strikes Back (1980)}	=>	{Star Wars: Episode IV - A New Hope (1977)}	0.1216147	0.7371203
[4]	{Lord of the Rings: The Fellowship of the Ring, The (2001)}	=>	{Lord of the Rings: The Two Towers, The (2002)}	0.1015446	0.7036304
[5]	{Star Wars: Episode IV - A New Hope (1977)}	=>	{Star Wars: Episode V - The Empire Strikes Back (1980)}	0.1216147	0.6007311
[6]	{Usual Suspects, The (1995)}	=>	{Shawshank Redemption, The (1994)}	0.1087696	0.5564970
[7]	{Schindler's List (1993)}	=>	{Shawshank Redemption, The (1994)}	0.1103377	0.5363109
[8]	{Usual Suspects, The (1995)}	=>	{Pulp Fiction (1994)}	0.1036769	0.5304416
[9]	{Silence of the Lambs, The (1991)}	=>	{Shawshank Redemption, The (1994)}	0.1095609	0.5110921
[10]	{Forrest Gump (1994)}	=>	{Shawshank Redemption, The (1994)}	0.1039188	0.5102357

1-10 of 15 rows | 1-6 of 8 columns Previous 1 2 Next

The rules with highest lift have been displayed below. Rules which have lift greater than 1 are significant and are deemed to be interesting. For the parameters we set in our Apriori algorithm the rules lift with highest lift have been displayed below.

```r
# Top 15 rules for lift
rules.by.lift <- sort(a, by="lift", decreasing = TRUE)
inspect(rules.by.lift[1:15])
```

Description: df [15 x 8]

	lhs		rhs	support	confidence	coverage	lift
	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
[1]	{Lord of the Rings: The Fellowship of the Ring, The (2001)}	=>	{Lord of the Rings: The Two Towers, The (2002)}	0.1015446	0.7036304	0.1443153	5.595586
[2]	{Lord of the Rings: The Two Towers, The (2002)}	=>	{Lord of the Rings: The Fellowship of the Ring, The (2001)}	0.1015446	0.8075287	0.1257474	5.595586
[3]	{Star Wars: Episode VI - Return of the Jedi (1983)}	=>	{Star Wars: Episode IV - A New Hope (1977)}	0.1009218	0.7609813	0.1326206	3.758963
[4]	{Star Wars: Episode IV - A New Hope (1977)}	=>	{Star Wars: Episode VI - Return of the Jedi (1983)}	0.1009218	0.4985160	0.2024445	3.758963
[5]	{Star Wars: Episode V - The Empire Strikes Back (1980)}	=>	{Star Wars: Episode IV - A New Hope (1977)}	0.1216147	0.7371203	0.1649862	3.641099
[6]	{Star Wars: Episode IV - A New Hope (1977)}	=>	{Star Wars: Episode V - The Empire Strikes Back (1980)}	0.1216147	0.6007311	0.2024445	3.641099
[7]	{Usual Suspects, The (1995)}	=>	{Pulp Fiction (1994)}	0.1036769	0.5304416	0.1954540	2.031784
[8]	{Pulp Fiction (1994)}	=>	{Usual Suspects, The (1995)}	0.1036769	0.3971203	0.2610719	2.031784
[9]	{Silence of the Lambs, The (1991)}	=>	{Pulp Fiction (1994)}	0.1052743	0.4910955	0.2143663	1.881074
[10]	{Pulp Fiction (1994)}	=>	{Silence of the Lambs, The (1991)}	0.1052743	0.4032389	0.2610719	1.881074

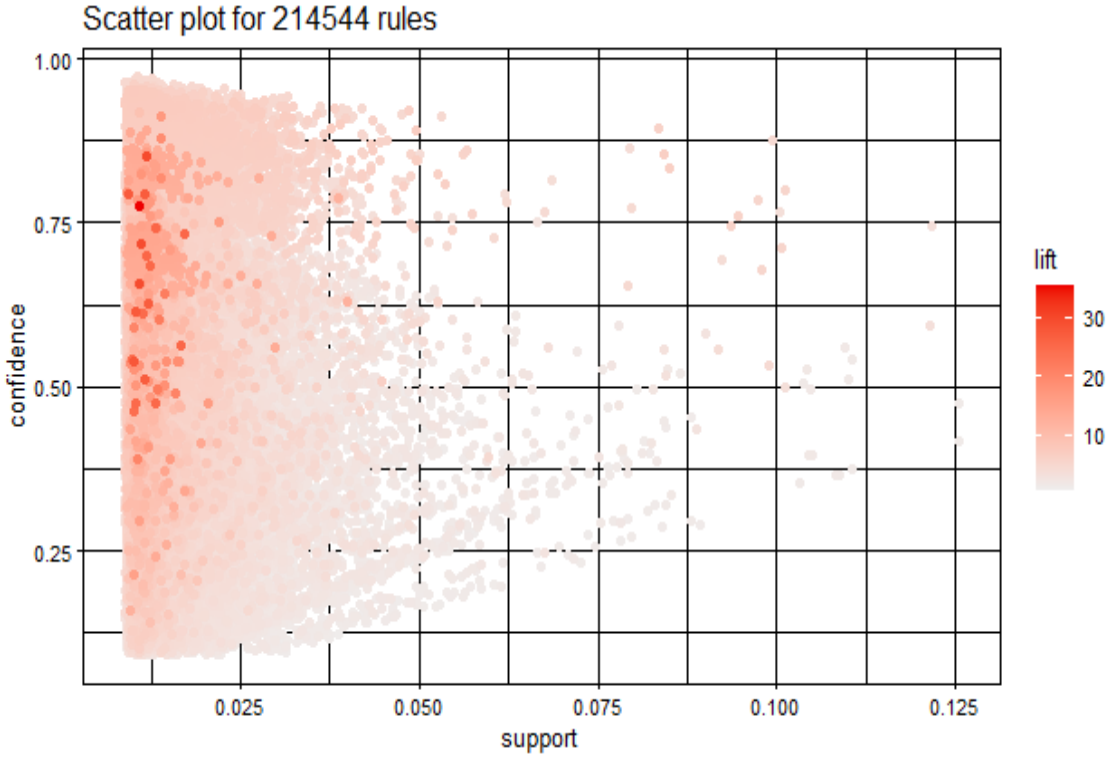
1-10 of 15 rows | 1-8 of 8 columns

Previous 1 2 Next

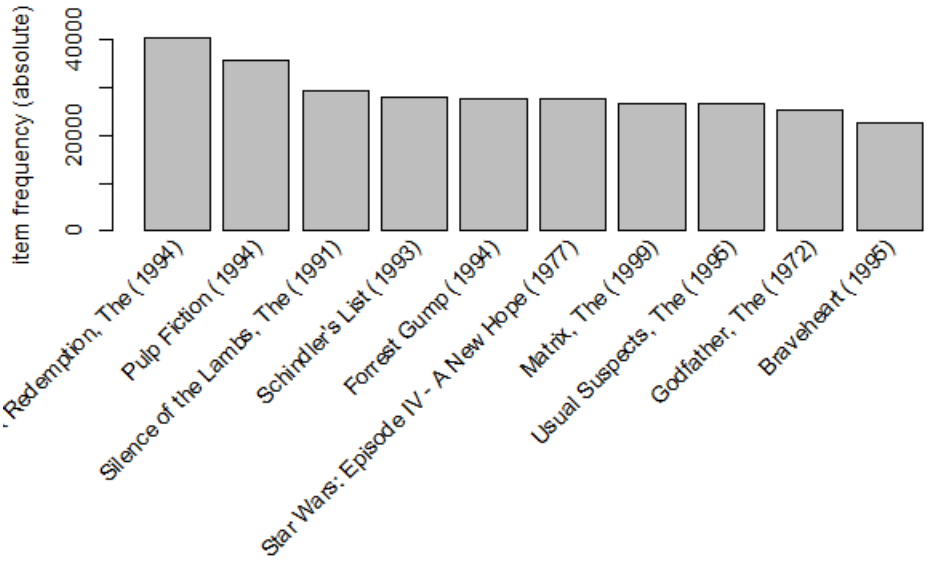
Lord of the rings movies has a very good lift. It means that they have a very good association with each other. Star Wars franchise movies have a very good lift. Popular movies have good lift in general.

VISUALIZATIONS OF THE RULES

Below is a general visualization of support, confidence and lift for our transactions.



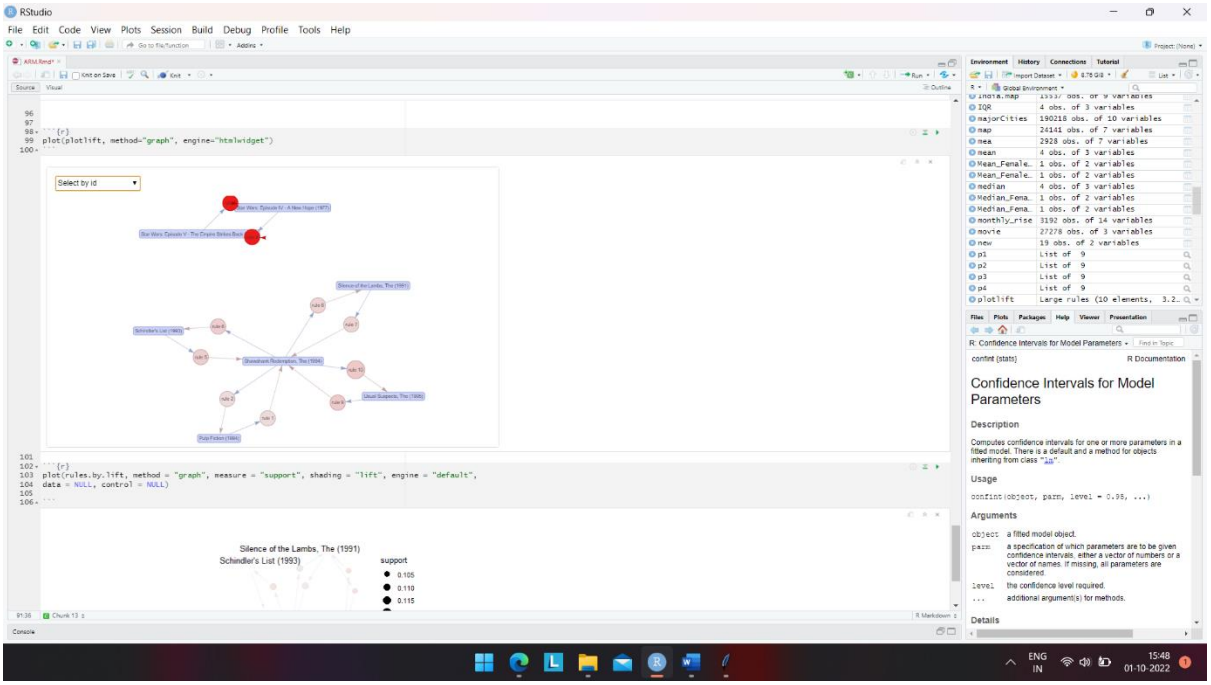
This is a frequency item plot, and we can the movies which have been occurring most in our data.



We can see the redemption , pulp fiction , silence of the lambs and so on have been occurring frequently in the dataset.

Below is a graph of associations which have the highest lift. It is an interactive image, but I couldn't export the interactive image to word.

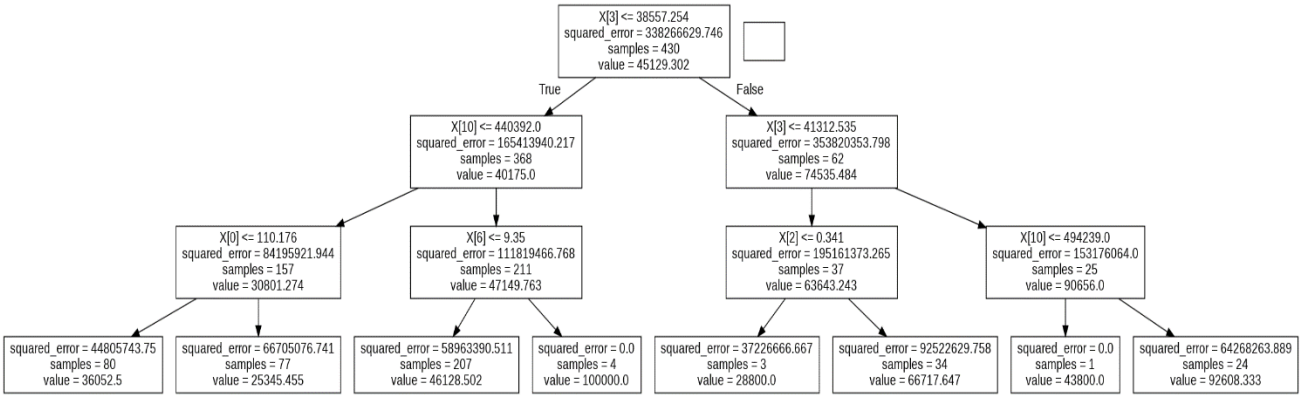
This is an interactive html widget which shows us associations with highest confidence.



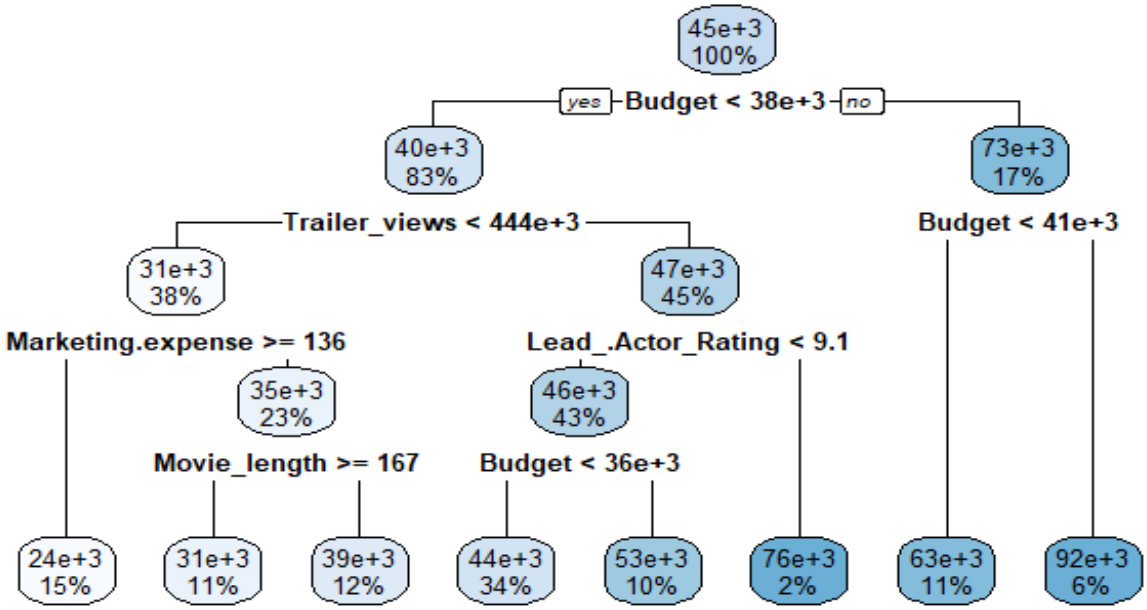
Visualization associated with ARM have been shown above.

RESULTS-DECISION TREES

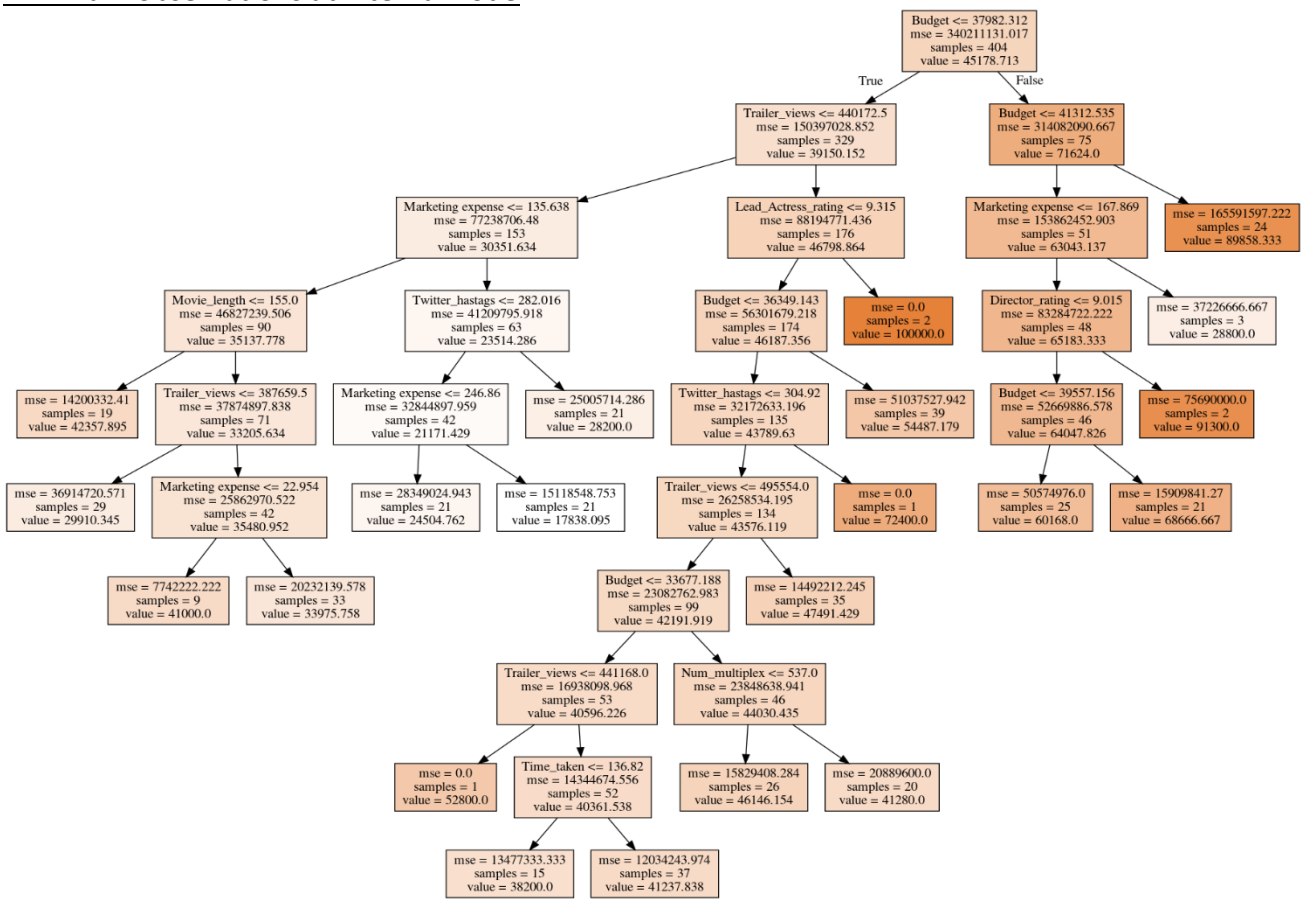
I discretised my data by converting continuous data attribute values into a finite set of intervals and associating with each interval some specific data value. I did this only for 1 column. This increased my modelling accuracy by a 4% to be exact.
Pruning a tree lets us cut off parts of the tree which are not beneficial for us.
Pre-Pruning helps us in controlling tree growth



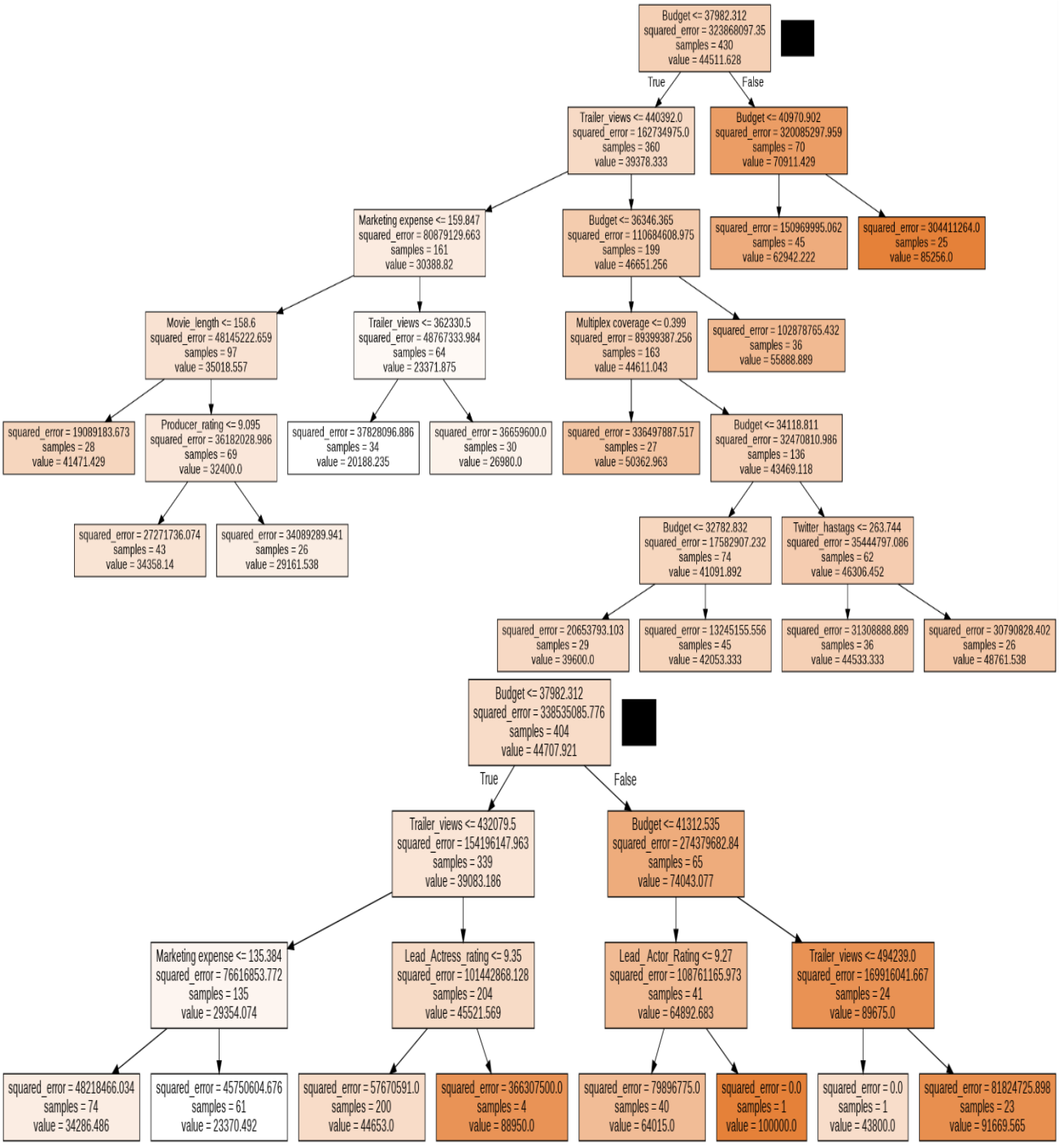
DECISION TREE in R



Minimum observations at internal node



Minimum observations at leaf node



*Confusion matrices for my decision trees are in my code. They were not copying into the document for some reason.

I learnt how to model a decision tree properly both in r and python. I feel that implementing decision trees is python is easier compared to R. I learnt the difference between a decision tree classifier and decision tree regressor.

My model accuracy was **83.01%** which is pretty good. I was predicting collection variable here hence I had to use a regressor instead of a classifier. These are the various trees I obtained by tuning the parameters of my model.

RESULTS-NAÏVE BAYES

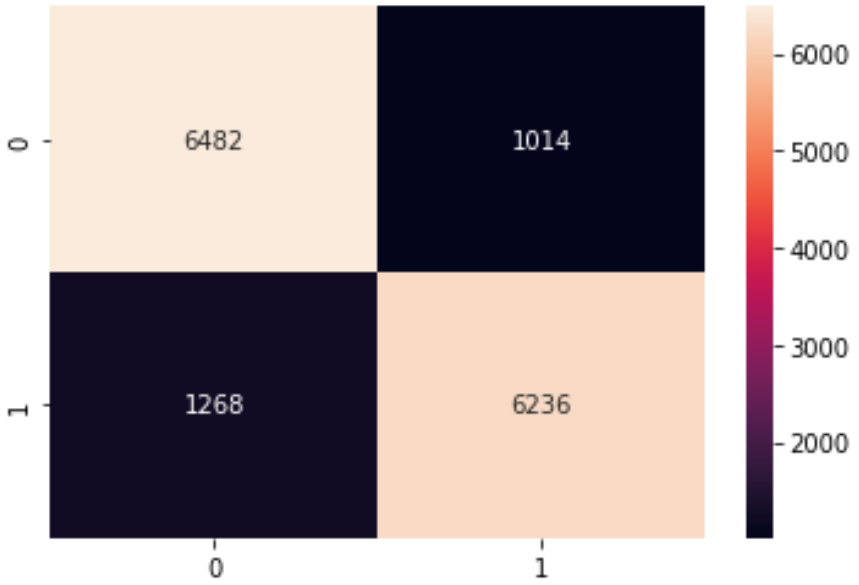
I implemented naïve bayes both in python as well as R.

I had text data, so I had to implement Multinomial Naïve bayes or Bernoulli naïve bayes. I found better accuracy with Multinomial naïve bayes, but there was only a difference of 1%. I have clearly differentiated both in my code and all the cleaning is included as well.

Confusion Matrix

The matrix represents the different combinations of Actual VS Predicted values. There were 15000 labels in total and out of that 6482 were predicted negative and were actually negative and 6236 were predicted positive and they were actually positive. We have 1014 predicted positive but they were negative and 1268 predicted positive but they were negative.

The accuracy of the model was **88.38285714285715%**



Confusion Matrix R

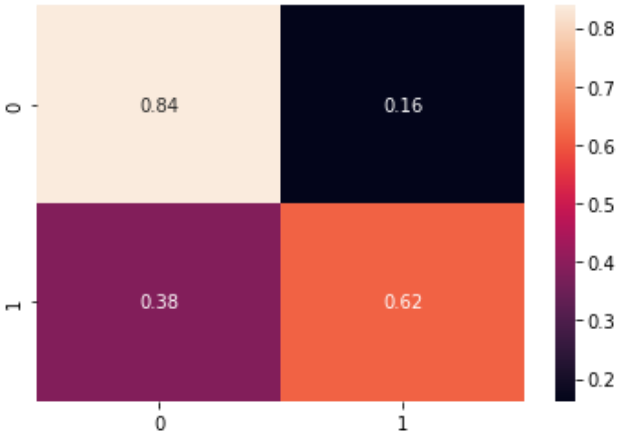
The matrix represents the different combinations of Actual VS Predicted values. There were 2642 labels in total and out of that 1198 were predicted negative and were actually negative and 433 were predicted positive and they were actually positive. We have 370 predicted positive but they were negative and 641 predicted positive but they were negative.

predictnaivetype	0	1
0	1198	370
1	641	433

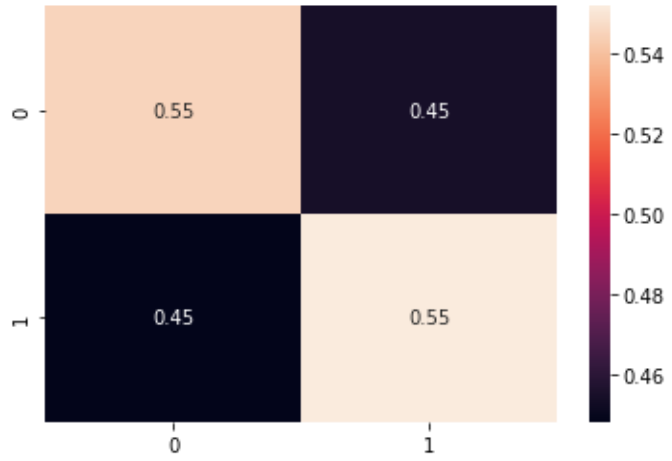
Naïve bayes helped me predict whether a movie review was positive or negative based on the user's description.

RESULTS-SVM

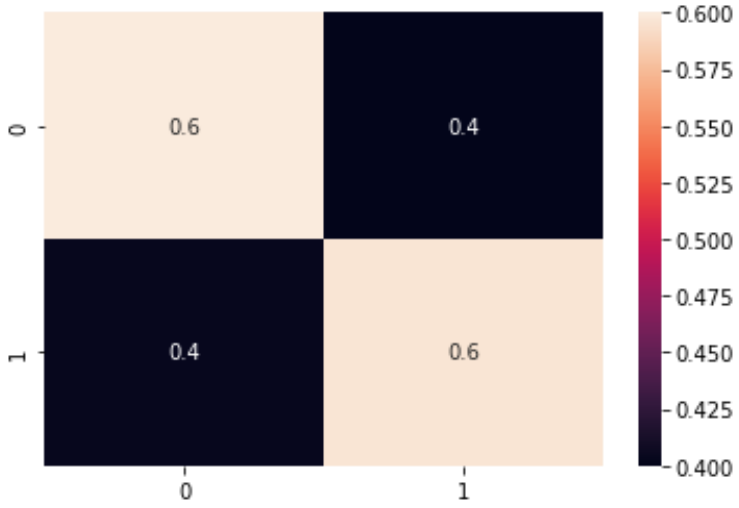
Confusion matrix for linear kernel



Confusion matrix for polynomial kernel



Confusion matrix for Sigmoid Kernel



The linear kernel gave me the best accuracy of 72% compared to the other kernels. I have listed the accuracy score for the other kernels in my code.

RESULTS-GENERAL

Naïve bayes had the best accuracy score for my data followed by decision trees and then followed by support vector machines. I couldn't use decision tree classifier as the accuracy score was low compared to the regressor. The accuracy score for the regressor was good.
Decision tree's offered critical information about the data as I was able to follow how the data was being classified and how the leaf nodes where being separated. Naïve bayes also offered me good information as it was able to predict the tone of the review very accurately and had a accuracy scores.
Support vector machines has the lowest score of all because I guess my data had too many features and it was not able to classify the data properly. I managed to get 70% accuracy which is still pretty good. In defence the dataset for the Oscars had only around 600 rows so that might not have been enough training data to be honest. I feel support vector machines would have done much better if I had a bigger dataset.

INTRODUCTION TO A NEURAL NETWORK

A neural network is made of artificial neurons that receive and process input data. Data is passed through the input layer, the hidden layer, and the output layer. A neural network process starts when input data is fed to it. Data is then processed via its layers to provide the desired output. A neural network learns from structured data and exhibits the output.

Neural networks are comprised of layers of neurons.

These layers consist of the following:

- Input layer
- Multiple hidden layers
- Output layer

The input layer receives data represented by a numeric value. Hidden layers perform the most computations required by the network. Finally, the output layer predicts the output. In a neural network, neurons dominate one another. Each layer is made of neurons. Once the input layer receives data, it is redirected to the hidden layer. Each input is assigned with weights. The weight is a value in a neural network that converts input data within the network's hidden layers. Weights work by input layer, taking input data, and multiplying it by the weight value.

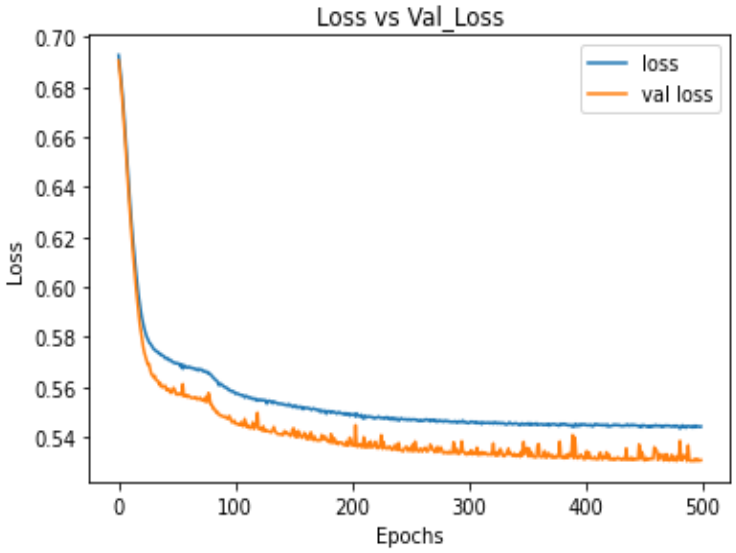
It then initiates a value for the first hidden layer. The hidden layers transform the input data and pass it to the other layer. The output layer produces the desired output. The inputs and weights are multiplied, and their sum is sent to neurons in the hidden layer. Bias is applied to each neuron. Each neuron adds the inputs it receives to get the sum. This value then transits through the activation function. The activation function outcome then decides if a neuron is activated or not. An activated neuron transfers information into the other layers. With this approach, the data gets generated in the network until the neuron reaches the output layer.

APPLYING NN TO MOVIES DATSET

My dataset had a lot of features like name of the actors, directors, movie , release year, gross , budget etc. I had to filter out certain features to ensure I use a new subset of my features to predict my target variable. I had rating as a target variable. Any movie below the rating of 6.5 was given a label of zero and a movie with a rating of 6.5 or greater was given a label of 1.
The label here is the rating of the movie converted to binary values. I plan to predict the rating of the movie based on the three features in the dataset.
The dataset has around 5500 rows and the there is an equal balance between the positive labels and the negative labels.

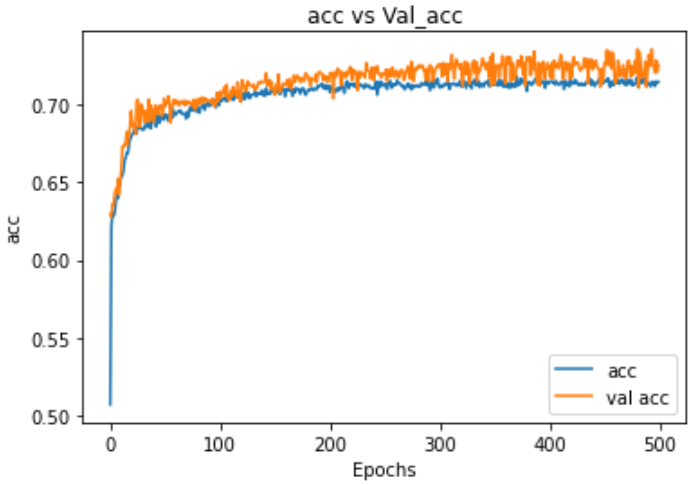
	votes	budget	gross	label
0	927000.0	19000000.0	46998772.0	1
1	65000.0	4500000.0	58853106.0	0
2	1200000.0	18000000.0	538375067.0	1
3	221000.0	3500000.0	83453539.0	1
4	108000.0	6000000.0	39846344.0	1
...	...	...	...	...
7648	140000.0	90000000.0	426505244.0	1
7649	102000.0	85000000.0	319715683.0	1
7650	53000.0	175000000.0	245487753.0	0
7651	42000.0	135000000.0	111105497.0	1
7652	3700.0	80000000.0	461421559.0	1

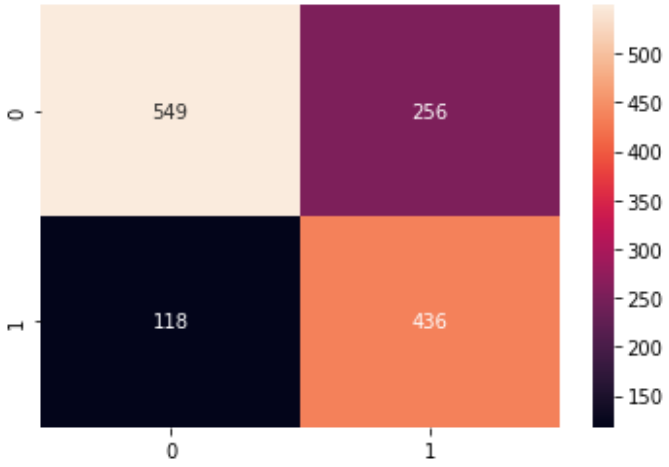
5436 rows × 4 columns



The neural network ran for 500 epochs

```
82/82 [==============================] - 0s 4ms/step - loss: 0.5728 - acc: 0.6875 - mse: 0.1977 - val_loss: 0.5614 - val_acc: 0.6887 - val_mse: 0.1923
Epoch 38/500
82/82 [==============================] - 0s 3ms/step - loss: 0.5729 - acc: 0.6843 - mse: 0.1978 - val_loss: 0.5601 - val_acc: 0.6990 - val_mse: 0.1917
Epoch 39/500
82/82 [==============================] - 0s 4ms/step - loss: 0.5726 - acc: 0.6909 - mse: 0.1976 - val_loss: 0.5601 - val_acc: 0.6968 - val_mse: 0.1917
Epoch 40/500
82/82 [==============================] - 0s 3ms/step - loss: 0.5721 - acc: 0.6902 - mse: 0.1974 - val_loss: 0.5603 - val_acc: 0.6924 - val_mse: 0.1919
Epoch 41/500
82/82 [==============================] - 0s 3ms/step - loss: 0.5715 - acc: 0.6902 - mse: 0.1972 - val_loss: 0.5605 - val_acc: 0.6880 - val_mse: 0.1921
Epoch 42/500
82/82 [==============================] - 0s 3ms/step - loss: 0.5717 - acc: 0.6865 - mse: 0.1973 - val_loss: 0.5588 - val_acc: 0.6990 - val_mse: 0.1912
Epoch 43/500
82/82 [==============================] - 0s 3ms/step - loss: 0.5714 - acc: 0.6853 - mse: 0.1972 - val_loss: 0.5590 - val_acc: 0.6968 - val_mse: 0.1914
Epoch 44/500
82/82 [==============================] - 0s 4ms/step - loss: 0.5708 - acc: 0.6929 - mse: 0.1969 - val_loss: 0.5593 - val_acc: 0.6932 - val_mse: 0.1916
Epoch 45/500
```



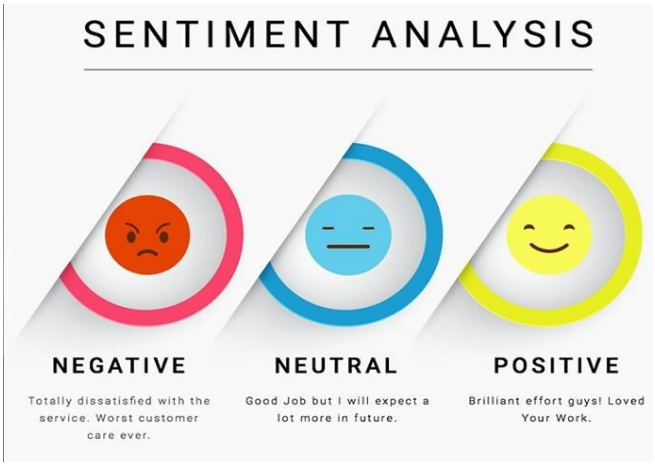


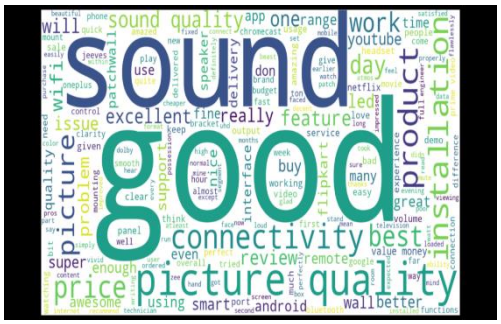
RESULTS- NEURAL NETWORK

I used the sigmoid as my activation function and a sequential model to classify ratings of the movies. The accuracy was 73% which is decent. I added 2 dense layers of 6 neurons each with the relu activation and a final dense layer of 1 neuron with the sigmoid activation function. The network was able to classify the movies based on the features given and I noticed the loss go down with the increase in the number of epochs. After 500 epochs the loss stabilised so that means that it is at its minimum value.

CONCLUSIONS

This conclusion is basically divided into two major parts. One of which focuses on Movie Recommendation system and the other on the Sentiment analysis. The study discusses both the systems and has come to some important conclusions.

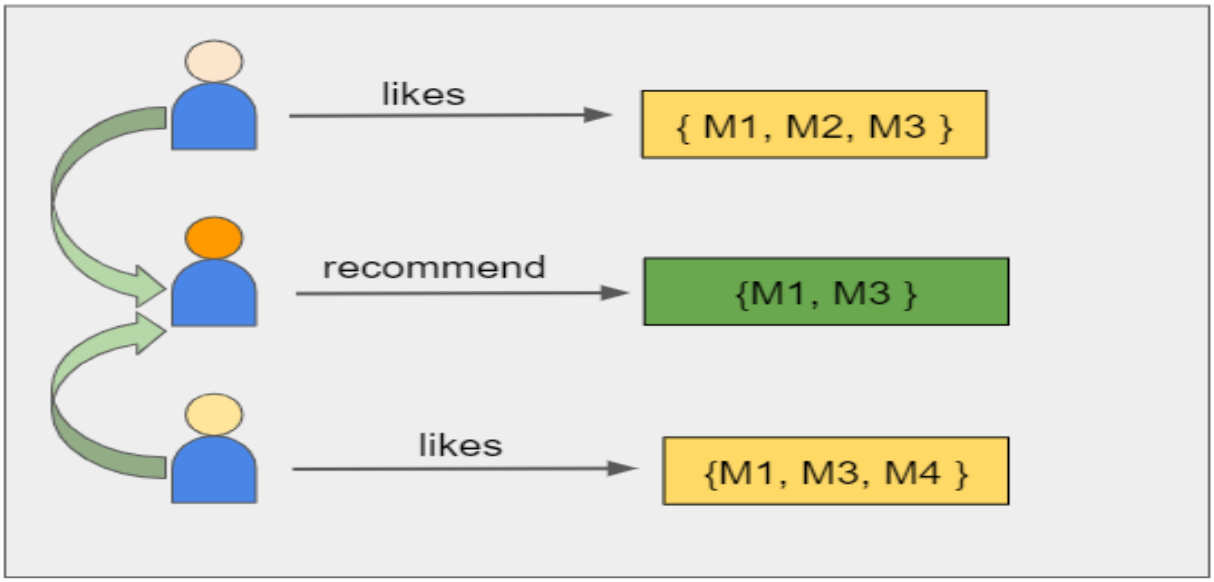




Sentiment analysis also plays an important role in this study. It basically aims to classify the reviews into positive or negative. Machine learning algorithms have been used for the same. The main reason behind using more than one algorithm is to find out what which is the best algorithm to classify the reviews because the reviews have huge diversity in them, so it is very important to choose the right algorithm for classification. Some prospects of this study have been mentioned below:

1) Increasing the Accuracy of both Sentiment Analysis for better classification of sarcastic or ironic reviews.
2) Sentiment Analysis of the reviews in different languages other than English.
3) Movie recommendation according to users' preference (cast, genre, year of release, etc.).

Although the system is accurate, it does have some limitations. One of which is, if the movie entered by the user isn't present in the dataset or if the user does not enter the name of the movie in the similar manner as that of in the dataset, then the system fails to classify movies. One more limitation is the linguistic barrier while doing the sentimental analysis. As of now only reviews written in English can be analysed. The Sentimental analysis also gives wrong classification if the reviews are sarcastic or ironic.



In this study, an efficient movie recommender system has been designed using the association rules mining technique and collaborative filter technique. Applying ML techniques gives more realistic movie lists for the user to choose. The results were evaluated in term of the important degree. The proposed system improves the important degree and gives better accuracy than the existing techniques used. Apriori algorithm improved the lists of user-recommended movies that are close to their liking, depending on which movie the user selects the first time. In the future, the proposed system can be more improved using big datasets. In addition, new directions for improvement could be using deep learning techniques which may enhance the efficiency of the movie recommendation system, in that case the model can be tuned to train more situations.