

## SKRIPSI

ANALISIS KESUKSESAN FILM DENGAN DATA MINING



Teuku Hashrul

NPM: 2016730067

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2020



**UNDERGRADUATE THESIS**

**MOVIE PROFIT ANALYSIS USING DATA MINING**



**Teuku Hashrul**

**NPM: 2016730067**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2020**



## **LEMBAR PENGESAHAN**

### **ANALISIS KESUKSESAN FILM DENGAN DATA MINING**

**Teuku Hashrul**

**NPM: 2016730067**

**Bandung, 19 Juni 2020**

**Menyetujui,**

**Pembimbing**

**Kristopher David Harjono M.T.**

**Ketua Tim Penguji**

**Anggota Tim Penguji**

**Dr.rer.nat. Cecilia Esti Nugraheni**

**Luciana Abednego, M.T.**

**Mengetahui,**

**Ketua Program Studi**

**Mariskha Tri Adithia, P.D.Eng**



## **PERNYATAAN**

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### **ANALISIS KESUKSESAN FILM DENGAN DATA MINING**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal 19 Juni 2020

Meterai Rp. 6000
---------------------

Teuku Hashrul  
NPM: 2016730067



## ABSTRAK

Film merupakan media komunikasi yang bersifat audio visual untuk menyampaikan suatu pesan atau cerita kepada penontonnya dan dijadikan sebagai media hiburan. Film yang dibuat ada karena kumpulan orang dibalik layar. Perusahaan film berlomba-lomba untuk membuat film yang memperoleh keuntungan maksimum. Terdapat banyak kemungkinan faktor yang dapat dijadikan film dapat memperoleh keuntungan maksimum. Penelitian ini adalah analisis kesuksesan film dengan *data mining* untuk memperoleh faktor-faktor yang dapat memprediksi kesuksesan sebuah film. Penelitian ini merupakan eksperimen untuk membandingkan beberapa metode *machine learning* seperti regresi dalam memprediksi kesuksesan sebuah film. Penelitian ini menggunakan bahasa pemrograman Python dan memanfaatkan beberapa *library* untuk melakukan *data mining*.

*Data mining* adalah proses menemukan suatu pola dari kumpulan data yang besar. Dengan *data mining*, manusia dapat menemukan sebuah informasi / pemahaman baru dari data. Kumpulan proses *data mining* adalah *Data cleaning* untuk menghilangkan *noise* dan data yang tidak konsisten. *Data integration* adalah proses menggabungkan data dari beberapa sumber. *Data transformation* adalah mengubah bentuk data menjadi lebih mudah dan relevan untuk kebutuhan analisis. *Data Selection* adalah proses memilih data yang relevan untuk kebutuhan analisis. *Data Mining* adalah tahap untuk menggunakan metode *Machine Learning* untuk menemukan pola dari sebuah data. *Pattern Evaluation* adalah tahap untuk memeriksa pola yang dihasilkan apakah menghasilkan kebenaran.

Penelitian ini menghasilkan informasi berupa hasil visualisasi data dari *dataset* film yang dianalisis. Perangkat lunak membaca *dataset* yang berupa data film dari tahun 2006 sampai 2016 lalu melakukan sekumpulan proses *data mining* seperti membersihkan data dengan menghilangkan *noise*, melakukan pengumpulan data tambahan media sosial seperti Youtube, melakukan integrasi data dengan menggabungkan *dataset* dengan data tambahan, melakukan pemilihan fitur, melakukan prediksi keuntungan menggunakan fitur yang sudah dipilih sebelumnya dan melakukan evaluasi terhadap model prediksi yang dibuat. Selain itu, perangkat lunak melakukan *clustering* untuk mengelompokkan data film pada *dataset* berdasarkan aktor dan *genre*.

Berdasarkan penelitian dan hasil evaluasi data yang dilakukan, dapat disimpulkan bahwa faktor yang dapat berpengaruh dalam kesuksesan film adalah jumlah penonton yang menyukai film tersebut (*votes*), besar biaya yang dikeluarkan untuk membuat film (*budget*) dan jumlah penonton *trailer* film pada situs Youtube. Selain itu, dapat disimpulkan bahwa selera penonton (*votes*) lebih berpengaruh dalam memperoleh kesuksesan film dibanding dengan selera kritis (*review*).

**Kata-kata kunci:** *Data Mining* , *Machine Learning*, Regresi, *Clustering*, *Dataset*



## ABSTRACT

Movie is an audio visual media to communicate and deliver some story to its viewer. People tend to watch movie as an entertainment purpose. There are movie makers in movie production house who wants to make their movies gain maximum profit. There are many possible factor that affects movie to gain maximum profit. This research is about movie profit analysis using data mining to acquire insights about movie industry and factor that will be used as a predictor to predict profit of a movie. There are several programs that will be made to collect, clean, select, analyze and evaluate movie data. This research also will do experiment to compare Machine Learning techniques such as regression to predict profit of a movie. This research will use Python as a programming language and several libraries to help develop Data Mining script.

Data Mining is a process of discovering interesting patterns and knowledge from large amounts of data. Data mining can help human to obtain new knowledge and information from data. There are several process in data mining such as data cleaning to remove noise data. Data integration is a process to combine data from multiple sources. Data transformation is a process to transform data to become more readable and more relevant for analysis. Data selection is a process to choose feature that correlates and more relevant to the model. Data mining is a process to use Machine Learning methods to find interesting patterns from a data. Pattern evaluation is a process to validate the Machine Learning model.

This research will produce interesting insights and data visualization from the chosen dataset. the program will read the dataset of a popular movie from 2006 to 2016. This research also will implementing data mining process such as remove noise from dataset, collect additional social media data features such as Youtube, integrate the additional data and the original dataset, select the most relevant data feature, do some predictive analysis to predict movie profit and evaluate the model. This dataset also will be clustered by actor and genre that similar.

According to the research and the data analysis that has been done, it could be concluded that the most important factor to increase the possibility to gain maximum profit from a movie are how many viewer that liked the movie (votes), how much the production cost (budget) and how many movie trailer views gained in Youtube. This research also conclude that how many viewer that liked the movie (votes) is more important factor that acquiring a good grade from professional reviewer.

**Keywords:** Data Mining, Machine Learning, Regression, Clustering, Dataset



*Skripsi ini saya persembahkan kepada ibunda dan almarhum  
ayahanda . . .*



## KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan yang Maha Esa karena atas berkat dan rahmat-Nya penulis berhasil menyelesaikan penyusunan skripsi ini yang berjudul "Analisis Kesuksesan Film Menggunakan Dengan Data Mining". Penulis menyadari bahwa penyusunan skripsi ini tidak akan berhasil tanpa dukungan doa dari berbagai pihak, oleh karena itu penulis ingin mengucapkan terima kasih kepada:

- Orang tua penulis yang telah bekerja keras mendoakan, mendukung dan memenuhi kebutuhan penulis selama proses penyusunan skripsi.
- Kedua kakak penulis yang selalu mendoakan dan mendukung penulis.
- Paman dan Tante yang selalu mendorong penulis untuk berkembang dan belajar.
- Bapak Kristopher David Harjono M.T.yang telah memberikan bimbingan dan arahan selama proses penyusunan skripsi.
- Sahabat semasa kuliah Giovanni, Timothy, Rashif, Naofal, Alif dan Shafira
- Sahabat semasa SMA Evander, Dewi, Reky, Cory, Annissa, Gintar dan Agung.
- Teman-teman Himpunan Mahasiswa Program Studi Teknik Informatika (HMPSTIF) sebagai tempat pengembangan diri.

Penulis berharap semoga skripsi ini dapat berguna bagi segenap pihak yang berkepentingan. Akhir kata, penulis memohon maaf apabila terdapat kekurangan dalam hasil penyusunan skripsi ini.

Bandung, Juni 2020

Penulis



## DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xxi</b>
<b>DAFTAR TABEL</b>	<b>xxv</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	1
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi . . . . .	2
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Penelitian Kaggle . . . . .	5
2.2 Measuring the Central Tendency [1] . . . . .	6
2.2.1 Mean . . . . .	7
2.2.2 Median . . . . .	7
2.2.3 Modus . . . . .	7
2.3 Data Mining . . . . .	7
2.3.1 Data Cleaning . . . . .	9
2.3.2 Data Integration . . . . .	9
2.3.3 Data Reduction . . . . .	9
2.3.4 Data Transformation . . . . .	10
2.3.5 Data Selection . . . . .	10
2.4 Machine Learning . . . . .	11
2.4.1 Regression . . . . .	11
2.4.2 Classification . . . . .	13
2.4.3 Clustering . . . . .	17
2.5 Data Visualization . . . . .	19
2.5.1 Boxplot . . . . .	20
2.5.2 Histogram . . . . .	21
2.5.3 Scatter Plot . . . . .	21
2.5.4 Pie Chart . . . . .	22
2.6 One Hot Encoding . . . . .	23
2.7 Web Scraping . . . . .	23
2.8 Istilah Dalam Bisnis Film . . . . .	24
<b>3 ANALISIS</b>	<b>25</b>
3.1 Analisis Masalah . . . . .	25

<b>3.2</b>	Tahap Penggerjaan Penelitian . . . . .	26
3.2.1	Analisis Data Utama . . . . .	26
3.2.2	Analisis Data Tambahan IMDB . . . . .	26
3.2.3	Analisis Data Sosial Media . . . . .	26
<b>3.3</b>	Analisis Penerapan Data Cleaning . . . . .	27
<b>3.4</b>	Analisis Penerapan Data Transformation . . . . .	27
3.4.1	Attribute Construction . . . . .	28
3.4.2	Normalization . . . . .	28
<b>3.5</b>	Analisis Penerapan Data Integration . . . . .	29
<b>3.6</b>	Analisis Penerapan Data Selection . . . . .	30
<b>3.7</b>	Analisis Penerapan Linear Regression . . . . .	31
<b>3.8</b>	Analisis Penerapan Polynomial Regression . . . . .	33
<b>3.9</b>	Analisis Penerapan Evaluasi Regresi . . . . .	36
<b>3.10</b>	Analisis Penerapan K-NN . . . . .	37
<b>3.11</b>	Analisis Penerapan Evaluasi Klasifikasi . . . . .	38
<b>3.12</b>	Analisis Penerapan K-Means . . . . .	39
<b>3.13</b>	Analisis Penerapan Agglomerative . . . . .	40
<b>3.14</b>	Analisis Penerapan Evaluasi Clustering . . . . .	42
<b>3.15</b>	Analisis Visualisasi Data . . . . .	43
<b>3.16</b>	Analisis Web Scrapping . . . . .	45
<b>4</b>	<b>IMPLEMENTASI</b> . . . . .	47
<b>4.1</b>	Lingkungan Perangkat Keras . . . . .	47
<b>4.2</b>	Lingkungan Perangkat Lunak . . . . .	47
<b>4.3</b>	Deskripsi Dataset . . . . .	47
<b>4.4</b>	Proses Analisis Data Utama . . . . .	49
4.4.1	Data Cleaning . . . . .	50
4.4.2	Analisis Data Utama Menggunakan Visualisasi . . . . .	51
4.4.3	Data Selection . . . . .	61
4.4.4	Percobaan Prediksi Fitur Data Utama . . . . .	64
<b>4.5</b>	Proses Analisis Data Tambahan . . . . .	69
4.5.1	Data Collection . . . . .	70
4.5.2	Data Integration . . . . .	74
4.5.3	Data Transformation . . . . .	74
4.5.4	Analisis Data Tambahan Menggunakan Visualisasi . . . . .	76
4.5.5	Data Selection . . . . .	81
4.5.6	Percobaan Prediksi Fitur Data Tambahan . . . . .	87
<b>4.6</b>	Proses Analisis Data Sosial Media . . . . .	88
4.6.1	Data Collection . . . . .	89
4.6.2	Data Integration . . . . .	95
4.6.3	Data Transformation . . . . .	96
4.6.4	Analisis Data Media Sosial Menggunakan Visualisasi . . . . .	97
4.6.5	Data Selection . . . . .	100
4.6.6	Percobaan Prediksi Fitur Data Sosial Media . . . . .	104
<b>4.7</b>	Analisis Clustering . . . . .	110
4.7.1	Clustering Berdasarkan Aktor . . . . .	110
4.7.2	Clustering Berdasarkan Genre . . . . .	117
<b>4.8</b>	Prediksi Berdasarkan Cluster . . . . .	122
<b>5</b>	<b>KESIMPULAN DAN SARAN</b> . . . . .	127
<b>5.1</b>	Kesimpulan . . . . .	127
<b>5.2</b>	Saran . . . . .	128

<b>DAFTAR REFERENSI</b>	<b>129</b>
<b>A KODE PROGRAM</b>	<b>131</b>
<b>B HASIL EKSPERIMEN</b>	<b>167</b>



## DAFTAR GAMBAR

2.1 Proses Data Mining . . . . .	8
2.2 Proses Klasifikasi . . . . .	14
2.3 Ilustrasi Decision Tree . . . . .	15
2.4 Boxplot dataset iris . . . . .	20
2.5 Histogram dataset iris . . . . .	21
2.6 Scatter plot dengan korelasi positif (a) dan negatif (b) . . . . .	22
2.7 Scatter plot tanpa korelasi . . . . .	22
2.8 Pie chart dataset iris . . . . .	23
3.1 Contoh dataset yang memiliki data kotor . . . . .	27
3.2 Kode analisis data cleaning . . . . .	27
3.3 Kode contoh Attribute Construction . . . . .	28
3.4 Kode contoh <i>Normalization</i> . . . . .	29
3.5 Kode contoh <i>integration</i> . . . . .	30
3.6 Kode contoh <i>pearson correlation</i> . . . . .	31
3.7 Kode contoh <i>linear regression</i> . . . . .	33
3.8 Kode contoh <i>polynomial regression</i> . . . . .	35
3.9 Perbandingan kurva linear dan polinom . . . . .	35
3.10 Kode contoh <i>evaluasi regresi</i> . . . . .	37
3.11 Kode contoh <i>K-NN</i> . . . . .	38
3.12 Kode contoh evaluasi klasifikasi . . . . .	39
3.13 Kode contoh <i>clustering K-Means</i> . . . . .	40
3.14 Dendogram example . . . . .	42
3.15 Kode contoh <i>agglomerative clustering</i> . . . . .	42
3.16 Kode contoh evaluasi clustering . . . . .	43
3.17 Contoh Analisis Histogram . . . . .	44
3.18 Contoh Analisis Boxplot . . . . .	44
3.19 Kode contoh visualisasi . . . . .	45
3.20 Flowchart Octoparse . . . . .	45
3.21 Tampilan utama Octoparse . . . . .	46
3.22 Tampilan scrapping pada Octoparse . . . . .	46
4.1 Ilustrasi sebuah baris film pada dataset . . . . .	48
4.2 Barchart deteksi jumlah baris yang kolomnya terdapat null . . . . .	50
4.3 Wordcloud kolom genre pada dataset . . . . .	51
4.4 Piechart persebaran film berdasarkan genre . . . . .	52
4.5 Top 10 Kombinasi genre dengan pendapatan kotor (Revenue) Terbaik . . . . .	53
4.6 Hubungan korelasi selera penonton dengan <i>review</i> . . . . .	54
4.7 Distribusi nilai kolom metascore . . . . .	54
4.8 Distribusi nilai kolom rating . . . . .	55
4.9 Perbandingan Distribusi nilai kolom review . . . . .	55
4.10 Histogram kolom metascore dan rating . . . . .	56
4.11 Distribusi nilai kolom runtime . . . . .	56

4.12 Distribusi kelas nilai runtime . . . . .	57
4.13 Hubungan korelasi runtime dan votes . . . . .	57
4.14 Distribusi nilai kolom votes . . . . .	58
4.15 Histogram distribusi kolom votes . . . . .	58
4.16 Distribusi nilai kolom year . . . . .	59
4.17 Histogram distribusi kolom year . . . . .	60
4.18 Line plot Jumlah Revenue dari tahun ke tahun . . . . .	60
4.19 Histogram Revenue Tahun 2011 . . . . .	61
4.20 Analisis Korelasi antara situs review dengan revenue . . . . .	62
4.21 Analisis Korelasi runtime year dan revenue . . . . .	62
4.22 Scatter plot votes dan revenue . . . . .	63
4.23 Pengujian perbandingan korelasi tiap fitur dengan revenue menggunakan pearson correlation . . . . .	63
4.24 Plot linear regression prediksi revenue dengan votes . . . . .	65
4.25 Distribusi nilai <i>squared error</i> . . . . .	65
4.26 Plot polynominal regression prediksi revenue dengan votes . . . . .	66
4.27 Distribusi nilai squared error prediksi revenue menggunakan <i>polynomial regression</i> . . . . .	67
4.28 Distribusi nilai squared error prediksi revenue dan perbandingan tiap algoritma regresi yang digunakan . . . . .	67
4.29 Fitur advance search pada situs IMDB . . . . .	70
4.30 Pengaturan octoparse pada hasil pencarian advanced search . . . . .	71
4.31 Detail page sebuah film pada situs IMDB . . . . .	71
4.32 Flowchart cara kerja scraping budget IMDB menggunakan Octoparse . . . . .	72
4.33 Flowchart cara kerja scraping budget IMDB menggunakan library IMDBpy . . . . .	73
4.34 Distribusi nilai kolom budget . . . . .	76
4.35 Distribusi nilai kolom profit . . . . .	76
4.36 Perbandingan film yang rugi dan untung pada dataset . . . . .	77
4.37 Distribusi nilai kolom ROI . . . . .	77
4.38 Histogram kolom Budget . . . . .	78
4.39 Histogram kolom Profit . . . . .	78
4.40 Histogram kolom ROI . . . . .	79
4.41 Line plot akumulasi profit dan budget semua film tiap tahun . . . . .	79
4.42 Top 10 Kombinasi genre dengan votes terbaik . . . . .	80
4.43 10 kombinasi genre dengan perbandingan budget dan revenue tertinggi (profit) . . . . .	81
4.44 Analisis korelasi kolom Budget dan Revenue . . . . .	82
4.45 Pengujian perbandingan korelasi tiap fitur dengan revenue menggunakan pearson correlation ke-2 . . . . .	83
4.46 Analisis korelasi situs review dan profit . . . . .	83
4.47 Analisis korelasi runtime votes dan profit . . . . .	84
4.48 Analisis korelasi budget dan profit . . . . .	84
4.49 Pengujian perbandingan korelasi tiap fitur dengan profit menggunakan pearson correlation ke-2 . . . . .	85
4.50 Analisis korelasi situs review dan ROI . . . . .	85
4.51 Analisis korelasi runtime votes dan roi . . . . .	86
4.52 Analisis korelasi budget dan profit . . . . .	86
4.53 Pengujian perbandingan korelasi tiap fitur dengan ROI menggunakan pearson correlation ke-2 . . . . .	87
4.54 Contoh JSON hasil request search youtube menggunakan API . . . . .	90
4.55 Contoh JSON hasil request seach youtube menggunakan API . . . . .	91
4.56 Contoh JSON Error Youtube Quota Exceeded . . . . .	91
4.57 Tampilan octoparse untuk scraping youtube view . . . . .	92

4.58	Contoh fitur Hashtag Instagram . . . . .	93
4.59	Boxplot view count . . . . .	97
4.60	Hashtag Instagram title boxplot . . . . .	98
4.61	Hashtag Instagram actor boxplot . . . . .	99
4.62	Analisis tren <i>youtube view trailer</i> . . . . .	100
4.63	Analisis tren <i>Instagram Hashtag</i> . . . . .	100
4.64	Pearson correlation fitur prediktor terhadap <i>revenue</i> . . . . .	101
4.65	Pearson correlation fitur prediktor terhadap <i>revenue</i> . . . . .	102
4.66	Pearson correlation fitur prediktor terhadap <i>profit</i> . . . . .	103
4.67	Pearson correlation fitur prediktor terhadap <i>profit</i> . . . . .	104
4.68	Distribusi nilai <i>squared errors</i> ( $(\text{ypred} - \text{ytest})^2$ ) <i>Linear regression</i> pada prediksi <i>revenue</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	105
4.69	Distribusi nilai <i>squared errors</i> ( $(\text{ypred} - \text{ytest})^2$ ) <i>Polynomial regression</i> pada prediksi <i>revenue</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	106
4.70	Distribusi nilai <i>squared errors</i> ( $(\text{ypred} - \text{ytest})^2$ ) <i>Linear regression</i> pada prediksi <i>profit</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	108
4.71	Distribusi nilai <i>squared errors</i> ( $(\text{ypred} - \text{ytest})^2$ ) <i>Polynomial regression</i> pada prediksi <i>profit</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	109
4.72	Kualitas Clustering Aktor berdasarkan jumlah menggunakan <i>Silhouette Score</i> . . . . .	111
4.73	Distribusi nilai <i>intraclass</i> pada beberapa percobaan clustering <i>actor</i> . . . . .	112
4.74	Distribusi jumlah anggota tiap <i>cluster</i> aktor . . . . .	113
4.75	Pengujian sebuah cluster merepresentasikan satu aktor dominan . . . . .	114
4.76	Wordcloud distribusi aktor cluster 13 . . . . .	115
4.77	Wordcloud distribusi aktor cluster 28 . . . . .	115
4.78	Wordcloud distribusi aktor cluster 161 . . . . .	115
4.79	Pengujian aktor dominan memiliki <i>genre</i> favorit pada tiap <i>cluster</i> . . . . .	116
4.80	Pengujian <i>genre</i> favorit pada tiap <i>cluster</i> juga berkontribusi tinggi untuk keuntungan . . . . .	117
4.81	Kualitas Clustering <i>Genre</i> berdasarkan jumlah menggunakan <i>Silhouette Score</i> . . . . .	118
4.82	Distribusi nilai <i>intraclass</i> pada beberapa percobaan clustering <i>genre</i> . . . . .	119
4.83	Perbandingan jumlah <i>cluster</i> yang memiliki aktor utama dan tidak . . . . .	121
4.84	Perbandingan jumlah <i>cluster</i> yang aktor utama berkontribusi berdasarkan <i>revenue</i> pada kombinasi <i>genre</i> . . . . .	122
4.85	Segmented Cluster Flowchart . . . . .	123
4.86	Distribusi R2 hasil prediksi revenue berdasarkan <i>cluster genre</i> . . . . .	124
4.87	Distribusi R2 hasil prediksi revenue berdasarkan <i>cluster aktor</i> . . . . .	124
B.1	20 Kombinasi <i>genre</i> dengan akumulasi <i>revenue</i> tertinggi . . . . .	167
B.2	20 Kombinasi <i>genre</i> yang paling banyak dibuat . . . . .	168
B.3	Distribusi <i>revenue</i> semua kombinasi <i>genre</i> . . . . .	169
B.4	Distribusi <i>votes</i> semua kombinasi <i>votes</i> . . . . .	170
B.5	Distribusi Jumlah anggota hasil cluster aktor . . . . .	171
B.6	Jumlah perbandingan kelompok aktor yang <i>genre</i> favoritnya berkontribusi tertinggi pada <i>profit</i> . . . . .	172
B.7	Jumlah perbandingan kelompok aktor yang <i>genre</i> favoritnya berkontribusi tertinggi pada <i>ROI</i> . . . . .	172



## DAFTAR TABEL

3.1	Tabel contoh yang perlu diubah . . . . .	28
3.2	Tabel dengan kolom baru setelah <i>Data Transformation</i> . . . . .	28
3.3	Tabel Dataset yang ingin dinormalisasi . . . . .	29
3.4	Tabel dataset setelah dinormalisasi . . . . .	29
3.5	Tabel kumpulan buku . . . . .	29
3.6	Tabel pengarang . . . . .	30
3.7	Tabel buku dan pengarang sudah tergabung . . . . .	30
3.8	Tabel data performa belajar siswa . . . . .	30
3.9	Tabel hasil perhitungan pearson . . . . .	31
3.10	Tabel dataset regresi . . . . .	32
3.11	Perhitungan komponen konstanta dan koefisien . . . . .	32
3.12	Tabel perbandingan volume penjualan prediksi dan volume penjualan asli . . . . .	33
3.13	Tabel dataset polinomial . . . . .	33
3.14	perhitungan komponen matriks hubungan polinom orde 2 . . . . .	34
3.15	Perbandingan prediksi volume penjualan (Y) polynomial regression . . . . .	35
3.16	Tabel perhitungan R2 berdasarkan contoh prediksi Linear Regression . . . . .	36
3.17	Tabel dataset k-nearest neighbor . . . . .	37
3.18	Tabel perhitungan euclidean distance dengan data baru . . . . .	37
3.19	Tabel 3 tetangga terdekat berdasarkan perhitungan euclidean distance . . . . .	38
3.20	Tabel dataset evaluasi klasifikasi . . . . .	38
3.21	Tabel dataset k-Means . . . . .	39
3.22	Tabel perhitungan k-Means iterasi ke-1 . . . . .	39
3.23	Tabel perubahan centroid iterasi 2 . . . . .	40
3.24	Tabel perhitungan k-Means iterasi ke-2 . . . . .	40
3.25	Tabel dataset perhitungan agglomerative . . . . .	40
3.26	Tabel perhitungan euclidean distance iterasi pertama . . . . .	41
3.27	Tabel perubahan nilai cluster iterasi 1 . . . . .	41
3.28	Tabel perhitungan euclidean distance iterasi kedua . . . . .	41
3.29	Tabel perubahan nilai cluster iterasi 2 . . . . .	41
3.30	Tabel dataset contoh yang akan di evaluasi . . . . .	42
3.31	Tabel dataset analisis visualisasi . . . . .	43
4.1	Deskripsi dataset . . . . .	49
4.2	5 Contoh Data pada dataset . . . . .	49
4.3	Tabel sebelum <i>split</i> . . . . .	51
4.4	Tabel setelah <i>split</i> . . . . .	51
4.5	Film <i>outlier</i> berdasarkan <i>votes</i> . . . . .	59
4.6	Tabel Percobaan Regresi Linear untuk prediksi revenue menggunakan <i>votes</i> . . . . .	64
4.7	Tabel Percobaan Regresi Polinom untuk prediksi revenue menggunakan <i>votes</i> . . . . .	66
4.8	Tabel perbandingan skor akurasi r2 menggunakan regresi . . . . .	67
4.9	Sampel contoh data hasil scraping budget IMDB menggunakan Octoparse . . . . .	72
4.10	Tabel 10 sampel hasil data scrapping menggunakan IMDBpy . . . . .	74

4.11 Tabel sampel data integrasi antara dataset dengan budget . . . . .	74
4.12 Tabel sampel beberapa konversi budget pada dataset . . . . .	75
4.13 Tabel 5 sampel pembuatan kolom profit pada dataset . . . . .	75
4.14 Tabel 5 sampel pembuatan kolom roi pada dataset . . . . .	75
4.15 Tabel Skor R2 Revenue . . . . .	87
4.16 Tabel sampel percobaan prediksi Revenue menggunakan regresi fitur votes dan budget	88
4.17 Tabel skor R2 prediksi profit . . . . .	88
4.18 Tabel sampel percobaan prediksi profit menggunakan budget dan votes dan perbandingannya dengan profit asli . . . . .	88
4.19 10 sampel url yang degenerate . . . . .	92
4.20 10 sampel hasil <i>scrapping youtube view</i> menggunakan <i>octoparse</i> . . . . .	93
4.21 10 Sampel URL Instagram Hashtag dengan Menggunakan Judul Film . . . . .	94
4.22 10 Sampel hasil <i>scraping</i> data hashtag instagram menggunakan <i>octoparse</i> . . . . .	94
4.23 10 Sampel hasil pengumpulan data <i>hashtag</i> aktor utama setiap film pada dataset . . . . .	95
4.24 10 Sampel hasil <i>scraping</i> data <i>hashtag</i> aktor dengan <i>Octoparse</i> . . . . .	95
4.25 Sampel data integrasi antara <i>dataset</i> dengan data sosial media . . . . .	96
4.26 Operasi dan contoh data <i>transformation</i> pada data media sosial . . . . .	96
4.27 5 Sampel <i>dataset</i> setelah konversi <i>data transformation</i> . . . . .	96
4.28 5 Film <i>outlier</i> dengan <i>View Count</i> Terbanyak . . . . .	98
4.29 5 Film <i>outlier</i> berdasarkan Hashtag judul terbanyak . . . . .	98
4.30 5 Film <i>outlier</i> berdasarkan <i>Actor Hashtag</i> terbanyak . . . . .	99
4.31 Tabel percobaan <i>Linear Regression</i> untuk prediksi <i>revenue</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	105
4.32 Hasil nilai R2 <i>linear regression</i> pada prediksi <i>revenue</i> menggunakan data sosial media	105
4.33 Tabel percobaan <i>Polynomial Regression</i> untuk prediksi <i>revenue</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	106
4.34 Hasil nilai R2 <i>polynomial regression</i> pada prediksi <i>revenue</i> menggunakan data sosial media . . . . .	107
4.35 Tabel percobaan <i>Linear Regression</i> untuk prediksi <i>profit</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	107
4.36 Hasil nilai R2 <i>linear regression</i> pada prediksi <i>profit</i> menggunakan data sosial media	108
4.37 Tabel percobaan <i>Polynomial Regression</i> untuk prediksi <i>profit</i> menggunakan fitur <i>votes,budget</i> dan <i>viewCount</i> . . . . .	109
4.38 Contoh Aktor pada <i>Dataset</i> . . . . .	110
4.39 Contoh Hasil One Hot Encoding Aktor . . . . .	110
4.40 Perbandingan waktu <i>clustering</i> menggunakan KMeans dan Agglomeratives . . . . .	112
4.41 Contoh Anggota <i>Cluster</i> 13,28 dan 161 hasil <i>cluster Actor</i> . . . . .	113
4.42 Contoh jumlah kontribusi aktor utama pada <i>cluster</i> 13 , 28 dan 161 . . . . .	117
4.43 Contoh Genre pada <i>Dataset</i> . . . . .	118
4.44 Contoh Hasil One Hot Encoding Genre . . . . .	118
4.45 Contoh Anggota <i>Cluster</i> 13,28 dan 62 hasil <i>cluster Genre</i> . . . . .	120
4.46 Hasil evaluasi prediksi Revenue menggunakan <i>Votes</i> dengan R2 Clustering <i>Genre</i> .	123
4.47 Hasil evaluasi prediksi Revenue menggunakan <i>Votes</i> dengan R2 Clustering <i>Actor</i> .	124

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Film merupakan media komunikasi yang bersifat audio visual untuk menyampaikan suatu pesan kepada penontonnya. Keberadaan film membuat masyarakat menjadikan film sebagai media hiburan. Beragam cara dapat dilakukan untuk menikmati sebuah film yaitu datang ke bioskop, membeli kaset DVD dan *streaming* menggunakan aplikasi *desktop* dan *smartphone*.

Film yang dibuat ada karena ada kumpulan orang di balik layar yang bekerja untuk membuatnya. Terdapat beragam perusahaan produksi film yang berlomba-lomba untuk membuat film yang dapat memperoleh keuntungan maksimum. Dengan menciptakan film yang sesuai dengan keinginan penontonnya, maka peluang keuntungan yang diperoleh pun akan semakin meningkat dan dapat menutup *budget* yang digunakan sebelumnya untuk biaya produksi.

Berdasarkan penelitian analisis data film yang ada sebelumnya pada *kaggle*, data film memiliki beberapa atribut umum yaitu judul, *genre*, *rating*, keuntungan, *budget*, nilai *review* dari situs internet dan aktor yang terlibat dan lama tayang. Data film yang ada digunakan untuk membantu menganalisis sifat dari film. Penelitian yang dilakukan adalah analisis prediksi nilai IMDB *score*, yaitu situs yang berisi data film.

*Genre* adalah sebutan untuk membedakan berbagai jenis film. Sebuah film memiliki satu atau beragam *genre*. Terdapat banyak film yang menggunakan kombinasi dari beberapa *genre*. *Genre* yang ada berupa *action*, *adventure*, *animation*, *drama*, *comedy*, *horror*, *romance* dan lain-lain.

Terdapat banyak kemungkinan faktor yang dapat dijadikan sebuah film dapat memperoleh keuntungan maksimum. Faktor kesuksesan film berupa faktor *rating* dari situs *review* film, nama aktor yang terlibat, nama sutradara yang terlibat dan jumlah *budget* yang dikeluarkan. Salah satu faktor dan kombinasi beberapa faktor dapat memengaruhi kesuksesan film.

Berdasarkan uraian diatas, akan dilakukan sebuah penelitian mengenai data film. Penelitian ini adalah analisis kesuksesan film dengan *data mining* untuk memperoleh faktor-faktor yang ada dapat memprediksi kesuksesan sebuah film. Dari faktor yang diperoleh, maka akan diprediksi *revenue*/pendapatan sebuah film berdasarkan data film yang sudah ada sebelumnya.

Pada penelitian ini dibuat sekumpulan perangkat lunak yang digunakan untuk mengumpulkan, membersihkan data, analisis, pembuatan model, evaluasi kerja model dan visualisasi data. Perangkat lunak yang dibuat akan membantu menganalisis data film yang digunakan. Pembuatan perangkat lunak akan menggunakan bahasa pemrograman *Python* dan memanfaatkan beberapa *library* dari *Python*. *Pandas* digunakan untuk integrasi data. *Sci-kit learn* digunakan untuk implementasi regresi, teknik *clustering* dan *classification* untuk memprediksi keuntungan sebuah film. Penelitian ini akan melakukan eksperimen untuk membandingkan beberapa metode *machine learning* dalam memprediksi kesuksesan sebuah film.

### 1.2 Rumusan Masalah

Berkaitan dengan identifikasi masalah yang ada pada deskripsi di atas, masalah-masalah yang ada dapat dirumuskan sebagai berikut.

- Apa saja faktor yang dapat digunakan untuk menentukan kesuksesan sebuah film ?
- Bagaimana langkah dalam melakukan analisis kesuksesan film dengan *data mining* ?
- Bagaimana hasil pengujian pada penelitian ini ?

### 1.3 Tujuan

Tujuan yang ingin dicapai dari penelitian ini adalah:

- Mengeksplorasi data yang dikumpulkan
- Membuat perangkat lunak yang dapat menggunakan metode *data mining* untuk melakukan analisis faktor-faktor yang dapat berpengaruh pada kesuksesan film
- Menguji metode-metode yang digunakan pada penelitian ini

### 1.4 Batasan Masalah

Pelaksanaan penelitian ini permasalahannya dibatasi pada:

1. Dataset yang digunakan pada penelitian berasal dari situs penyedia dataset seperti *Kaggle*
2. Data yang digunakan merupakan data IMDB dari tahun 2006 sampai 2016
3. Penelitian ini akan membuat mengimplementasikan tahapan *data mining* dengan memanfaatkan *library* dari *Python*
4. Penelitian ini tidak membuat antarmuka perangkat lunak dalam memprediksi kesuksesan film sehingga penjabaran dalam penelitian ini menggunakan visualisasi data

### 1.5 Metodologi

Langkah-langkah yang akan dilakukan dalam melakukan penelitian ini, yaitu:

1. Melakukan studi literatur dengan mencari jurnal, *paper* mengenai penelitian sejenis dari berbagai sumber untuk membantu penulis dalam menulis
2. Melakukan studi literatur langkah-langkah teknik *data mining* untuk memahami konsep
3. Melakukan studi literatur mengenai metode-metode *machine learning* yaitu regresi, *clustering* dan *classification* yang relevan
4. Melakukan studi literatur mengenai teori dan implementasi visualisasi data seperti *histogram*, *scatter plot*, *box plot* untuk membantu mengetahui sifat data yang dikumpulkan menggunakan *matplotlib*
5. Melakukan penelitian sejenis mengenai industri perfilman untuk mengetahui relevansi antar faktor yang ada
6. Mempelajari bahasa pemrograman *Python* dan beberapa *library* dari *Python* seperti *Pandas*, *Sci-Kit learn* dan *matplotlib*
7. Mencari sumber data yang relevan untuk melakukan pengumpulan data dari situs *review* film dan media sosial

8. Melakukan integrasi data dari sumber yang digunakan
9. Melakukan eksplorasi untuk menemukan sifat data
10. Melakukan analisis data secara statistik dengan teknik visualisasi data
11. Menerapkan metode-metode *machine learning* regresi, *clustering* dan *classification*
12. Melakukan pengujian dan eksperimen
13. Menulis dokumen skripsi

## 1.6 Sistematika Pembahasan

Sistematika penulisan ini berguna untuk memberikan gambaran secara umum mengenai penelitian yang akan dibuat. Berikut ini adalah uraian dari sistematika pembahasan :

- Bab 1. Pendahuluan, membahas tentang latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, metode penelitian dan sistematika pembahasan mengenai skripsi
- Bab 2. Landasan Teori, membahas pengertian *data mining*, langkah-langkah *data mining*, algoritma *Machine Learning*, visualisasi data dan *web scraping*
- Bab 3. Analisis, membahas tentang bagaimana implementasi teori yang dijelaskan pada bab sebelumnya. Pada bab ini juga menjelaskan implementasi menggunakan *library Python*.
- Bab 4. Implementasi, membahas tentang deskripsi *dataset*, hasil analisis data, pengujian prediksi dan interpretasi pola menarik dari visualisasi.
- Bab 5. Kesimpulan dan Saran membahas kesimpulan yang diperoleh setelah melakukan analisis data serta saran yang dapat diberikan untuk pengembangan lebih lanjut tentang analisis data film



## BAB 2

### LANDASAN TEORI

Bab Landasan Teori ini berisi teori yang menjadi dasar eksperimen penelitian ini. *Data Mining* sebagai teori utama yang mendasari penelitian ini. Akan dijelaskan tahap-tahap dari *Data Mining* dan dasar dari *Machine Learning* yang akan digunakan untuk melakukan analisis data terhadap kesuksesan film. Beberapa teknik *preprocessing* juga akan dijelaskan untuk digunakan saat memproses data.

#### 2.1 Penelitian Kaggle

Sebagai referensi utama dalam penelitian, penulis mempelajari *data mining* dari artikel situs *kaggle*<sup>1</sup> yang berjudul "*Analyze IMDB score with data mining algorithms*". Artikel ini membahas penerapan *data mining* pada data film untuk memprediksi nilai IMDB untuk sebuah film. Terdapat 3 metode yang dibandingkan menggunakan *classification*. Bahasa pemrograman yang digunakan untuk eksperimen adalah R.

Di dalam artikel tersebut, terdapat langkah-langkah *data mining* yang diterapkan. Penelitian dimulai dengan *Data Exploration* untuk memahami data. *Data cleaning* untuk membersihkan data agar dapat digunakan saat *Data mining*. *Data visualization* untuk memvisualisasikan data sehingga mempermudah dalam melihat korelasi dan hubungan tiap data. *Data preprocessing* untuk mengubah data sesuai kebutuhan. *Implement algorithm* untuk memasukkan data yang sudah diproses ke algoritma yang relevan yaitu *classification*. *Classification* merupakan salah satu algoritma *supervised machine learning* yang digunakan untuk memprediksi nilai sebuah kelas.

*Dataset* yang digunakan pada artikel *kaggle* memiliki 5043 data film dengan 28 variabel. Data film berasal dari 100 negara. *Dataset* memiliki 2399 nama sutradara. Nama-nama variabel beserta deskripsinya adalah sebagai berikut :

- **movie \_title** : judul film
- **duration** : durasi film
- **director \_name**: nama sutradara
- **director \_facebook \_likes** : jumlah *likes* pada *facebook page* sutradara
- **actor \_1 \_name** : nama pemeran utama
- **actor \_1 \_facebook \_likes** : jumlah *likes* pada *facebook page* artis
- **actor \_2 \_name** : nama pemeran pendukung pertama
- **actor \_2 \_facebook \_likes** : jumlah *likes* pada *facebook page* artis
- **actor \_3 \_name** : nama pemeran pendukung kedua

---

<sup>1</sup>Yueming, (2018) "*Analyze IMDB score with data mining algorithms*" <https://www.kaggle.com/carolzhangdc/analyze-imdb-score-with-data-mining-algorithms>. 8 April 2020.

- **actor\_3\_facebook\_likes** : jumlah *likes* pada *facebook page* artis
- **num\_user\_for\_reviews** : jumlah *user* yang memberikan *review*
- **num\_critic\_for\_reviews** : jumlah *user* yang memberikan *critical review*
- **num\_voted\_user** : jumlah *user* yang mendukung film
- **cast\_total\_facebook\_likes** : jumlah *like* dari setiap pemeran film di *facebook*
- **movie\_facebook\_likes** : jumlah *likes* film di *facebook*
- **plot\_keywords** : kata kunci yang mendeskripsikan film
- **facenumber\_in\_poster** : jumlah wajah pemain di poster film
- **color** : jenis warna film ('Black and white' atau 'Color')
- **genres** : kategori film
- **title\_year** : tahun rilis film (dari tahun 1916 sampai 2016)
- **language** : bahasa film
- **country** : negara asal film
- **content\_rating** : *nilai* konten dalam film
- **aspect\_ratio** : perbandingan panjang dan lebar layar resolusi film
- **movie\_imdb\_link** : tautan film pada *imdb*
- **gross** : pendapatan kotor
- **budget** : biaya produksi film
- **imdb\_score** : nilai film yang diberikan oleh *imdb*

Pada tahap eksplorasi data, beberapa hal dilakukan untuk melakukan *data cleaning* seperti menghilangkan film yang memiliki *missing value*, menghilangkan film yang duplikat, melakukan normalisasi pada judul film. Tahap eksplorasi dilakukan untuk memahami data dan mengubah data ke bentuk yang lebih relevan untuk tahap pengujian model

Pada tahap visualisasi data, terdapat beberapa teknik yang dilakukan untuk membantu menganalisis data menggunakan grafik seperti memunculkan histogram jumlah film setiap tahun dari 1916 sampai 2016. Pengurutan data dilakukan untuk memunculkan informasi-informasi yang dibutuhkan seperti 20 besar film berdasarkan keuntungan dan 20 besar nama sutradara yang menghasilkan film dengan nilai IMDB tertinggi

Eksperimen yang dilakukan pada artikel akan memproses *dataset* yang sudah dibersihkan untuk dimasukkan ke beberapa algoritma *classification* yaitu *Decision Tree*, *K-nearest neighbors* dan *Random Forest*. Berdasarkan hasil pengujian ternyata *Random Forest* memiliki akurasi yang paling tinggi dibanding algoritma lain yaitu 0,76 atau 76 persen. Sehingga, *model* yang dibuat dapat dipercaya untuk memprediksi seberapa bagus film berdasarkan skor IMDB.

## 2.2 Measuring the Central Tendency [1]

*Central Tendency* adalah cara untuk mengukur persebaran tiap nilai pada kumpulan data. Kumpulan data yang besar sulit untuk dibaca sehingga membutuhkan suatu cara untuk memahaminya. *Central Tendency* dapat menghitung sebuah nilai yang dapat merepresentasikan kumpulan data. Terdapat beberapa cara untuk mengukur persebaran data yaitu menggunakan *mean* / rata-rata, *median* (Q2) dan *modus*.

### 2.2.1 Mean

*Mean* / rata-rata adalah bilangan yang mewakili sekumpulan data. Rata-rata dapat dihitung dengan menjumlahkan setiap data dibagi dengan jumlah elemen pada data tersebut. Sebuah kumpulan data X dengan elemen  $x_1, x_2, \dots, x_N$  memiliki  $N$  elemen. Berikut adalah rumus *mean* yaitu :

$$\bar{x} = \frac{\sum_i^N x_i}{N} = \frac{x_1 + x_2 + \dots + x_N}{N} \quad (2.1)$$

X pada Persamaan 2.1 merupakan kumpulan data.  $x_1$  sampai  $x_N$  merepresentasikan tiap nilai dari kumpulan data. Penjumlahan dari setiap elemen dapat dibagi dengan jumlah elemen untuk mendapatkan rata-rata dari data tersebut. Untuk mempermudah penjelasan, diberikan contoh yaitu sebuah kumpulan data nilai ujian suatu kelas yaitu 90,85,75 dan 80. Jumlah elemen pada data nilai yaitu 4. Menggunakan *mean* pada 2.1, didapatkan :

$$\bar{x} = \frac{90 + 85 + 75 + 80}{4} = \frac{330}{4} = 82.5$$

### 2.2.2 Median

*Median* adalah nilai tengah yang dapatkan dari sebuah data yang terurut. Jika banyaknya data genap, maka rumus mediannya adalah :

$$Me(Q2) = \frac{(X_{n/2} + X_{(n/2)+1})}{2} \quad (2.2)$$

$X_n$  dan  $X_{n+1}$  pada Persamaan 2.3 merupakan data urutan ke  $n/2$  yang sudah terurut menaik. Jika banyaknya data ganjil, maka rumus median adalah :

$$Me(Q2) = \frac{X_{(n+1)/2}}{2} \quad (2.3)$$

$X_{(n+1)/2}$  pada Persamaan 2.3 merupakan data urutan ke  $n/2$  yang sudah terurut menaik. Untuk mempermudah penjelasan, diberikan sebuah contoh yaitu kumpulan data nilai ujian yang sudah terurut menaik yaitu 75,80,85 dan 90. Maka didapatkan perhitungan *median* yaitu :

$$Me(Q2) = \frac{X_{n/2} + X_{n/2} + 1}{2} = \frac{X_2 + X_3}{2} = \frac{80 + 85}{2} = 82.5$$

### 2.2.3 Modus

*Modus* adalah elemen pada kumpulan data yang paling sering muncul. Terdapat beberapa jenis *modus* yaitu *unimodal*, *bimodal* dan *trimodal*. *Modus unimodal* yaitu elemen dengan frekuensi terbanyak berjumlah 1. Berikut adalah contoh kumpulan data nilai ujian sekolah yang terurut adalah 75,80,85,85 dan 90. *Modus* pada kumpulan data ini adalah 85 karena frekuensi kemunculan elemen 85 adalah 2 kali.

## 2.3 Data Mining [1]

*Data Mining / Knowledge Discovery Process* (KDD) adalah proses menemukan suatu pola dari kumpulan data yang besar. Dengan *data mining*, manusia dapat menemukan sebuah informasi / pemahaman baru dari data. Sumber data objek yang dapat diproses untuk *data mining* adalah dari *database*, *data warehouse* atau data yang didapatkan dari sebuah proses.

Suatu data objek yang digunakan di *data mining* merupakan sebuah entitas. Kumpulan dari data objek disebut *data set*. Contoh nama lain *data set* adalah *data points* dan *objects*. Contoh

data objek adalah data pelanggan di salon, transaksi pembelian di supermarket, data pasien di rumah sakit, data mahasiswa di kampus dan lain-lain.

Atribut adalah sebuah karakteristik dan kondisi dari data objek. Data objek dapat memiliki satu atau lebih atribut. Atribut dapat disebut juga sebagai dimensi, fitur atau variabel. Contoh atribut adalah sebuah data objek transaksi pembelian di supermarket memiliki kumpulan atribut yaitu *customer\_ID*, *name* dan *address*. Sebuah atribut memiliki beberapa jenis yaitu nominal, biner, ordinal, atau numerik.

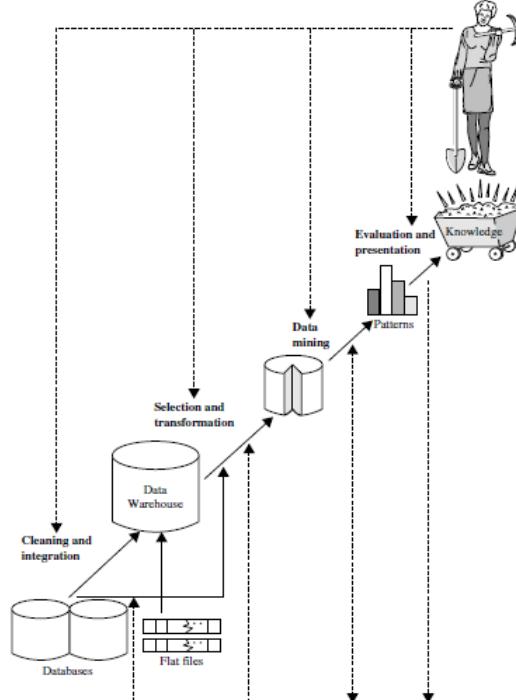
Atribut numerik adalah jenis nilai kuantitatif. Atribut numerik dapat diukur kuantitasnya. Nilai dari atribut numerik dapat berupa bilangan bulat (*integer*) atau bilangan *real*. Atribut numerik dapat dibagi lagi menjadi interval dan rasio.

Atribut nominal adalah tipe atribut yang tiap nilainya merupakan sebuah kategori/kondisi/kode. Atribut nominal memiliki isi berupa sebuah nama, simbol dari objek yang direpresentasikan. Nilai nominal tidak dapat dipengaruhi oleh perhitungan karena setiap angka/kode menunjukkan kondisi. Contoh atribut nominal adalah sebuah objek data pelanggan supermarket memiliki atribut jenis kelamin yang bernilai 'Pria' atau 'Wanita'.

Atribut biner adalah tipe atribut yang serupa dengan nominal tetapi hanya memiliki 2 jenis nilai. Biasanya, nilai biner dikodekan menjadi 0 dan 1. Contoh dari nilai biner adalah atribut pembayaran pada data objek transaksi memiliki nilai *true* atau *false* yang menyatakan pernyataan pembayaran sudah diselesaikan.

Atribut ordinal adalah tipe atribut yang memiliki hubungan keterurutan dari setiap nilainya. Atribut ordinal dapat dihasilkan dari mengubah nilai numerik yang disebut *discretization*. Contoh dari atribut ordinal adalah nilai survei kepuasan penduduk yang awalnya memiliki rentang nilai 0 sampai 10 lalu dikonversikan menjadi ordinal berupa buruk, puas dan sangat puas.

Berikut adalah ilustrasi gambar proses *Data Mining* yaitu :



Gambar 2.1: Proses Data Mining

berdasarkan Gambar 2.1 di atas, berikut adalah penjelasan dari masing-masing proses yaitu :

- ***Data cleaning*** : menghilangkan noise dan data yang tidak konsisten
- ***Data integration*** : menggabungkan data dari beberapa sumber jika ada

- **Data transformation** : mengubah bentuk data menjadi lebih mudah dan relevan untuk kebutuhan analisis
- **Data selection** : memilih data yang relevan untuk melakukan analisis
- **Data mining** : proses menggunakan metode *machine learning* untuk menemukan pola dari sebuah data
- **Pattern evaluation** : untuk memeriksa dari pola yang dihasilkan apakah dapat menghasilkan kebenaran mengenai pola yang ditemukan

### 2.3.1 Data Cleaning

*Data Cleaning* merupakan salah satu tahap *data preprocessing*. *Data cleaning* adalah kegiatan untuk membersihkan data. Data kotor adalah data dengan yang memiliki *missing value*, mengubah atau menghilangkan *noisy data*. *Noisy data* adalah data yang seharusnya tidak berada dikumpulan *dataset*. *Noisy data* dapat muncul dikarenakan beberapa hal seperti kesalahan proses saat perangkat keras membaca data, salah input, *human error* atau kesalahan saat diproses menggunakan perangkat lunak. *Noisy data* harus dihilangkan dikarenakan dapat memengaruhi hasil proses *data mining*.

Proses *Data Cleaning* membantu analisis data yang lebih valid. Terdapat beberapa cara yang dapat dilakukan jika pada data terdapat nilai yang hilang / *missing value* seperti :

- **Mengabaikan bagian data tersebut:** metode ini dapat dilakukan dengan cara mengabaikan *missing value*. *Missing value* dapat diabaikan jika data yang *missing value* tidak terlalu banyak. Sedikit data yang diabaikan tidak akan mempengaruhi hasil *data mining*
- **Mengisi nilai yang hilang secara manual:** metode ini tidak dapat direkomendasikan karena prosesnya yang memakan banyak waktu.
- **Menggunakan nilai konstan untuk mengisi nilai yang hilang:** *missing value* dapat diubah dengan membuat nilai konstan. Contoh yang dapat dipahami adalah mengubah *value* yang kosong menjadi '*unknown*' pada suatu atribut. Data yang sudah ditandai '*unknown*' akan membantu komputer mengidentifikasinya.
- **Menggunakan nilai tengah:** nilai yang hilang dapat diubah dengan nilai tengah yang tidak akan mempengaruhi distribusi data. Cara yang dapat digunakan adalah mengubah data tersebut menjadi *mean* /rata-rata dari *data set*.

### 2.3.2 Data Integration

*Data integration* adalah proses menggabungkan data dari beberapa sumber data. *Data integration* dapat menghasilkan data baru yang akan membantu proses *data mining*. Kebutuhan baru akan tambahan deskripsi data membuat proses *data integration* perlu dilakukan. Cara yang dapat dilakukan pada tahap *Data integration* adalah menggabungkan beberapa tabel dari basisdata, *web crawling* yaitu mengekstrak data dari *web pages*.

### 2.3.3 Data Reduction

*Data reduction* adalah salah satu *data preprocessing* yang dilakukan untuk mengurangi *data set* menjadi jumlah yang lebih sedikit. *Data reduction* diterapkan untuk mempercepat proses *data mining* dan meningkatkan akurasi dalam proses *data mining*. Metode *Data reduction* dibagi menjadi dua yaitu *dimensionality reduction* dan  *numerosity reduction*.

**Dimensionality Reduction** adalah proses memilih atribut tertentu dan menghilangkan atribut data yang tidak dibutuhkan. Teknik yang dapat dilakukan adalah menggunakan *feature selection*.

*Feature selection* adalah teknik untuk memilih atribut yang relevan untuk *predictive analysis* menggunakan *machine learning*.

**Numerosity Reduction** adalah proses memilih baris data tertentu dan menghilangkan sisa baris yang tidak digunakan. Cara yang dapat dilakukan adalah *sampling* yaitu mengambil *subset* dari data secara *random*. *Sampling* akan mengambil sebagian data yang mewakili kumpulan dari *dataset*.

### 2.3.4 Data Transformation

*Data Transformation* adalah proses mengubah data yang lebih sesuai saat digunakan saat proses *data mining*. Terdapat beberapa metode yang dapat digunakan untuk *data transformation* yaitu :

- **Smoothing** : *smoothing* adalah kegiatan untuk memproses *noisy data*. *Noisy data* dapat diproses dengan berbagai cara seperti mengubah menjadi *mean* data tersebut dan mengabaikannya.
- **Attribute Construction** : menambah atribut baru berdasarkan atribut yang sudah ada demi membantu proses *data mining*.
- **Aggregation**: menghitung dan menyimpan data laporan seperti nilai maksimum, minimum dan rata-rata dari *data set*
- **Normalization**: atribut data tertentu diskalakan pada rentang nilai tertentu yang lebih distandardkan. Normalisasi akan mengubah skala sebuah kumpulan nilai yang ada menjadi rentang 0 sampai 0.1. Salah satu caranya adalah dengan menggunakan *Minmax Normalization*
- **Discretization**: proses mengubah atribut numerik menjadi diskret. Contohnya adalah mengubah atribut 'umur'. Umur dapat diubah menjadi kelompok nilai yang memiliki jarak berdasarkan jenis. Contohnya adalah 'balita' direntang 2-5 tahun, anak kecil direntang 5-12 dan 'remaja' rentang 12-17 tahun.

### 2.3.5 Data Selection

*Data Selection* adalah proses memilih data yang tepat untuk melakukan analisis dengan *machine learning*. Sebuah *data set* memiliki dua atau lebih atribut. Data perlu dipilih dengan mengambil atribut yang relevan dan mengabaikan atribut yang tidak relevan. *Machine Learning* akan melakukan prediksi nilai atribut yang disebut *response / dependen* berdasarkan atribut prediktor/independen. Pemilihan atribut yang tepat akan meningkatkan akurasi dalam pembuatan *model*. Terdapat beberapa teknik yang dapat digunakan untuk memilih fitur/prediktor yang tepat.

#### Pearson Correlation

*Pearson Correlation* adalah teknik yang dapat digunakan untuk memeriksa korelasi antara 2 atribut numerik yaitu A dan B. Berikut adalah rumus dari penghitungan koefisien korelasi :

$$r_{A,B} = \frac{\sum_{i=1}^n (ai - \bar{A})(bi - \bar{B})}{n\sigma_A\sigma_B} \quad (2.4)$$

Persamaan 2.4 adalah rumus *pearson*.  $n$  adalah jumlah baris,  $ai$  dan  $bi$  adalah nilai A baris i dan nilai B baris i,  $\bar{A}$  dan  $\bar{B}$  adalah nilai rata-rata dari A dan B,  $\sigma_A$  dan  $\sigma_B$  adalah nilai standar deviasi dari A dan B. Jika hasil perhitungan koefisien korelasi lebih dari 0, maka A dan B memiliki hubungan **korelasi positif**. Hubungan korelasi positif adalah kondisi ketika naiknya nilai A maka nilai B juga akan naik.

Jika hasil perhitungan koefisien korelasi adalah 0, maka A dan B adalah atribut **independen** dan tidak memiliki korelasi. Jika hasil perhitungan koefisien korelasi adalah lebih kecil dari 0, maka

A dan B memiliki hubungan **korelasi negatif**. Korelasi negatif adalah hubungan dimana ketika satu atribut nilainya semakin bertambah, maka atribut lain akan berkurang. *Scatter plot* juga dapat digunakan untuk melihat korelasi antara 2 atribut.

### Chi Square ( $X^2$ )

*Chi Square* adalah teknik yang dapat digunakan untuk memeriksa korelasi antara 2 atribut kategori. Berikut adalah rumus dari perhitungan *chi square* yaitu :

$$X_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (2.5)$$

Persamaan 2.5 adalah rumus untuk menghitung *chi Square*. C adalah tiap kejadian atribut respon dan prediktor.  $O_i$  adalah jumlah *observed value* yaitu jumlah kemunculan prediktor i pada respon i. *Expected value* dapat dihitung dengan :

$$E1 = n * p \quad (2.6)$$

$$p = P(predictor_i) * P(response_i) \quad (2.7)$$

Persamaan 2.7 adalah perhitungan untuk mendapatkan nilai *Expected*. Penjumlahan dari setiap perhitungan kemungkinan pasangan atribut prediktor dan respon akan menjadi *chi square*. Semakin tinggi nilai *chi square*, maka semakin relevan sebuah pasangan atribut prediktor dan respon digunakan.

## 2.4 Machine Learning [2]

*Machine Learning* adalah metode yang dapat dilakukan komputer untuk belajar berdasarkan data. Dengan *Machine Learning*, komputer dapat mengambil sebuah keputusan atau memprediksi. Komputer akan mencari pola dari kumpulan sampel data yang disebut dengan *training data*. *Machine learning* juga disebut sebagai *predictive analysis* karena dapat membantu komputer untuk mengambil sebuah keputusan. *Machine Learning* dapat dibedakan berdasarkan jenis *input* dan *output* yang dihasilkan. Jenis-jenis kategori *Machine Learning* yaitu :

- **Supervised Learning** : algoritma yang menerima kumpulan sampel data yang dijadikan *training data* dan menghasilkan *output* berupa jenis kelas dari data tersebut. *Supervised learning* menerima data yang sudah memiliki label agar dapat memprediksi data baru berdasarkan *training data*. Algoritma *supervised learning* antara lain adalah *Regression* dan *Classification*. *Supervised learning* disebut sebagai *predictive analysis* karena kemampuannya untuk memprediksi nilai
- **Unsupervised Learning** : algoritma yang menerima kumpulan sampel data *training* yang belum memiliki label. *Unsupervised learning* dapat digunakan untuk mengelompokkan kumpulan data berdasarkan nilai dan kesamaan. Algoritma *unsupervised learning* adalah *clustering*. *Unsupervised Learning* disebut sebagai *descriptive analysis* karena kemampuannya untuk mengelompokkan data sesuai kemiripan dan mendeskripsikannya.

### 2.4.1 Regression [3]

*Regression* adalah teknik *supervised learning* yang digunakan untuk memprediksi nilai kontinu. Regresi menerima sampel data numerik sebagai input untuk menghasilkan sebuah persamaan yang dapat digunakan untuk memprediksi nilai yang dibutuhkan. Regresi memprediksi nilai variabel bergantung (*response*) berdasarkan nilai variabel independen. Terdapat beberapa jenis *regression* yaitu *linear regression* dan *polynomial regression*.

## Linear Regression

Linear Regression adalah algoritma regresi yang digunakan untuk menghasilkan persamaan linear. *Linear regression* juga dapat digunakan untuk menguji sejauh mana hubungan sebab akibat antara variabel dependen dengan variabel independen. *Linear Regression* dapat digunakan untuk memprediksi nilai kontinu. Persamaan *linear regression* yaitu :

$$Y = a + bX \quad (2.8)$$

Nilai Y pada persamaan (2.8) merupakan atribut variabel dependen (*response*). X merupakan atribut variabel independen (*predictor*). a merupakan konstanta dan b merupakan koefisien regresi (kemiringan). Koefisien regresi merupakan besaran *response* yang ditimbulkan oleh *predictor*. Nilai-nilai a dan b dapat dihitung dengan menggunakan rumus yaitu :

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2} \quad (2.9)$$

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \quad (2.10)$$

Berdasarkan uraian rumus (2.10) dan rumus (2.9), seberapa kuatnya pengaruh variabel atribut independen terhadap variabel atribut dependen dapat dihitung menggunakan koefisien determinasi ( $R^2$ ).

## Polynomial Regression

*Polynomial Regression* adalah algoritma *regression* yang digunakan untuk menghasilkan persamaan polinomial (suku banyak) berdasarkan data yang diinput. Persamaan akan berubah sesuai dengan *degree* / orde yang ditentukan. Persamaan yang dihasilkan dapat digunakan untuk menghitung nilai variabel bergantung menggunakan nilai variabel independen dan koefisien yang ditemukan. Persamaan *polynomial regression* adalah :

$$y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n \quad (2.11)$$

Nilai Y pada persamaan 2.11 merupakan variabel dependen yang ingin diprediksi. Nilai  $a_0$  sampai  $a_n$  adalah nilai koefisien. Nilai X adalah atribut variabel independen. Sebelum menggunakan persamaan di atas, kita perlu untuk mencari koefisien dari setiap suku perpangkatan. Untuk persamaan polinomial orde 2 didapatkan hubungan yaitu :

$$\begin{cases} na_0 + (\sum_{i=1}^n xi)a_1 + (\sum_{i=1}^n xi^2)a_2 &= \sum_{i=1}^n yi \\ (\sum_{i=1}^n xi)a_0 + (\sum_{i=1}^n xi^2)a_1 + (\sum_{i=1}^n xi^3)a_2 &= \sum_{i=1}^n (xiyi) \\ (\sum_{i=1}^n xi^2)a_0 + (\sum_{i=1}^n xi^3)a_1 + (\sum_{i=1}^n xi^4)a_2 &= \sum_{i=1}^n (x_i^2 y_i) \end{cases} \quad (2.12)$$

Berdasarkan persamaan 2.12, dapat diubah dalam bentuk matriks persamaan untuk mendapatkan koefisien yaitu :

$$\begin{bmatrix} n & \sum_{i=1}^n xi & \sum_{i=1}^n xi^2 \\ \sum_{i=1}^n xi & \sum_{i=1}^n xi^2 & \sum_{i=1}^n xi^3 \\ \sum_{i=1}^n xi^2 & \sum_{i=1}^n xi^3 & \sum_{i=1}^n xi^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n yi \\ \sum_{i=1}^n (xiyi) \\ \sum_{i=1}^n (x_i^2 y_i) \end{bmatrix} \quad (2.13)$$

Persamaan (2.13) dapat menemukan masing-masing koefisien sehingga dapat dimasukan pada persamaan *Polynomial Regression*. Hasil perhitungan koefisien dapat diaplikasikan pada rumus persamaan sehingga bisa menghitung prediksi nilai atribut respon (dependen).

## Evaluasi Regresi

Sebuah model yang dibuat untuk melakukan prediksi dengan regresi dapat diukur estimasi keakuratan prediksinya. Metode yang dilakukan untuk mengukurnya yaitu nilai *error* dan akurasi. Metode yang dapat digunakan untuk menguji seberapa baik hasil prediksi atribut respon yang ingin diperoleh adalah dengan membandingkan selisih antara atribut respon pada *test set* (*y true*) dengan atribut respon yang diprediksi (*y pred*) menggunakan model ini

### **Mean Squared Error**

Selisih antara prediksi dan yang asli disebut dengan *error*. *Mean Squared Error* (MSE) adalah rata-rata dari nilai *error* setiap data objek. Rumus dari MSE (2.14) adalah :

$$\frac{1}{N} \sum_{i=1}^n (y_{truei} - y_{predi})^2 \quad (2.14)$$

*Root Mean Squared Error* (RMSE) adalah akar pangkat dari MSE yang akan memberikan nilai yang lebih dapat dinormalkan. Berikut adalah rumus dari RMSE (2.15) yaitu :

$$\sqrt{\frac{1}{N} \sum_{i=1}^n (y_{truei} - y_{predi})^2} \quad (2.15)$$

### **Coefficient of Determination (R2)**

**Koefisien determinasi** adalah nilai yang menunjukkan kuat/tidaknya hubungan antara dua variabel. Berikut adalah cara menghitung koefisien determinasi :

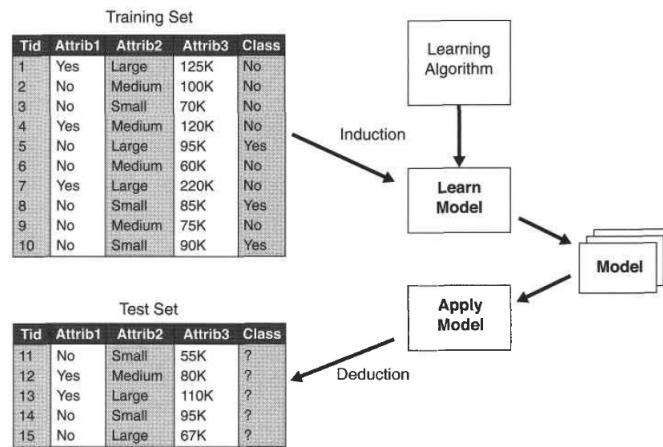
$$R^2 = 1 - \frac{\sum (y - \hat{y})^2}{\sum (y - \bar{y})^2} \quad (2.16)$$

Uraian rumus (2.16) di atas merupakan cara untuk menghitung koefisien determinasi. Y adalah nilai *response* / nilai asli yang ingin diprediksi.  $\hat{y}$  adalah hasil prediksi yang dihasilkan oleh model.  $\bar{y}$  adalah *mean* dari kumpulan data.

Terdapat 3 kemungkinan dalam hasil perhitungan koefisien determinasi. Jika nilai  $R^2 > 0$ , maka kedua atribut memiliki korelasi positif. Korelasi positif terjadi saat suatu nilai atribut meningkat (X), maka atribut lainnya (Y) juga meningkat. Jika nilai  $R^2 < 0$ , maka kedua atribut memiliki korelasi negatif. Korelasi negatif terjadi saat nilai atribut meningkat, maka atribut lainnya (Y) juga akan menurun. Jika nilai  $R^2 = 0$ , maka kedua atribut tidak memiliki korelasi sama sekali. Koefisien determinasi juga dapat digunakan untuk mengukur seberapa baik sebuah model / prediksi yang dihasilkan. Kemungkinan skor terbaik untuk  $R^2$  adalah 1.0

## 2.4.2 Classification

*Classification* merupakan metode *supervised learning* selain regresi. *Classification* dilakukan untuk menetapkan label data / kategori pada sebuah *data object*. *Classification* merupakan proses untuk memetakan variabel prediktor / independen (X) terhadap variabel respon / dependen (Y). Terdapat beberapa algoritma *classification* yang dapat digunakan untuk memprediksi nilai label pada *data set*.



Gambar 2.2: Proses Klasifikasi

Gambar 2.2 menunjukkan sebuah proses dalam melakukan *classification*. *Classifier / classification technique* adalah sebuah cara untuk membuat model dari *input data set*. Pada awalnya, *dataset* dapat dibagi menjadi 2 bagian yaitu *training set* dan *test set*. Terdapat *training set* yaitu kumpulan data / *record* yang sudah memiliki label data akan *ditrain* modelnya dengan menggunakan algoritma *classification*. Model yang sudah dibuat akan digunakan untuk memprediksi nilai label dari data yang ingin diprediksi yaitu *test set* untuk diuji keakuratannya. Hasil nilai label yang sudah diprediksi dapat dinilai dengan cara membandingkan hasil prediksi label dengan label asli pada *test set*.

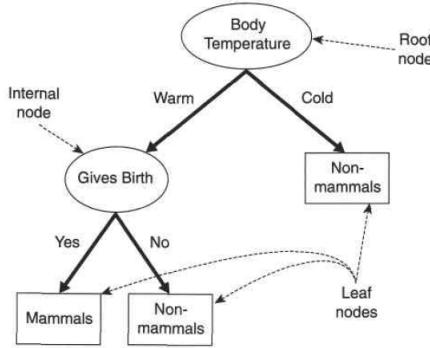
Dalam memanfaatkan *classification* untuk memprediksi nilai label, terdapat beberapa algoritma yang dapat dimanfaatkan untuk melakukan *classification* yaitu *Decision Tree*, *k Nearest Neighbors* (*kNN*), dan *Naive Bayes Classifier*.

### Decision Tree

*Decision tree* adalah teknik *classifier* dapat digunakan untuk menentukan label data pada *test set*. *Decision tree* akan membentuk sebuah pohon keputusan berdasarkan atribut prediktor yang akan digunakan untuk membuat model. Pohon keputusan yang dihasilkan akan menjadi aturan bagaimana menentukan label dari *data set*. Membuat *Decision tree* akan memanfaatkan *training set* untuk menentukan berdasarkan apa pohon dapat *displit*.

Struktur sebuah pohon keputusan memiliki 3 *node* yaitu :

- **Root node** : tidak memiliki *incoming edge* dan tidak atau banyak *edges*
- **Internal nodes** : memiliki satu *incoming edge* dan banyak *outgoing edges*
- **Leaf / terminal nodes** memiliki satu *incoming edge* dan tidak ada *outgoing edges*



Gambar 2.3: Ilustrasi Decision Tree

Gambar 2.3 menunjukkan sebuah contoh bagaimana penerapan pohon keputusan pada sebuah *dataset* binatang. Tujuan dari pohon keputusan adalah untuk menentukan aturan apakah sebuah binatang termasuk *mammals* atau *non-mammals*. Atribut prediktor yang terdapat ada *root node* dan *internal nodes* seperti *Body temperature* dan *Given Birth* digunakan untuk memecah *data* agar dapat menentukan label.

Teknik ini dapat menghasilkan berbagai pohon keputusan berdasarkan attribut yang dipilih sebagai atribut pembagi. Untuk menghasilkan *model* yang akurat, dibutuhkan cara untuk menentukan atribut yang dapat membagi pohon. Cara memilih atribut adalah dengan mengukur tingkat *impurity* / ketidakmurnian dari *node*. Tujuan dari pembuatan pohon keputusan ini adalah untuk membuat semua *node pure* / murni. Sebuah *node* dapat dikatakan murni jika sudah memiliki label data.

*Entropy* adalah ukuran *randomness* / ketidakpastian sebuah data. Semakin rendah nilai *entropy*, maka semakin murni *node* tersebut. Rumus dari entropi adalah yaitu :

$$\text{Entropy}(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2(i|t) \quad (2.17)$$

Berdasarkan uraian rumus (2.17), perhitungan entropi merupakan penjumlahan peluang tiap kelas  $i$  terhadap semua jumlah data  $t$ . Untuk menentukan atribut mana yang dapat digunakan sebagai pemecah, kita butuh untuk membandingkan tingkat ketidakmurnian *parent node* (sebelum dipecah) dengan tingkat ketidakmurnian dari *child node* (setelah dipecah). **Information Gain** ( $\Delta \text{ info}$ ) adalah sebuah kriteria yang dapat digunakan untuk menentukan atribut pembagi dalam pembentukan pohon. Rumus dari *information gain* yaitu :

$$\Delta \text{info} = I(\text{parent}) - \sum_{j=1}^k \frac{N(vj)}{N} I(jv) \quad (2.18)$$

Rumus (2.18)  $I$  adalah nilai ketidakmurnian yang dapat diperoleh menggunakan *entropy*.  $N$  adalah total observasi data / jumlah *record* dari *parent node*,  $k$  adalah jumlah nilai dari suatu atribut, dan  $vj$  adalah jumlah *record* pada nilai atribut  $j$ . *Decision tree* akan memilih atribut prediktor mana dengan memilih nilai  $\Delta$  ( $\text{info}$ ) yang maksimum.

## K-Nearest Neighbor Classifiers

*K-Nearest Neighbor* adalah algoritma *classification* untuk menentukan label sebuah data. *K-Nearest Neighbor* akan menentukan label data baru berdasarkan *dataset* yang sudah memiliki label pada *training set*. Menentukan label untuk data baru dapat ditentukan dengan mencari *training data* yang terdekat dengan data baru. Label pada  $k$ -jarak terdekat menjadi label bagi data baru. Setiap data pada *dataset* dapat dianalogikan sebagai titik yang akan dibandingkan jaraknya. Menghitung

jarak antara titik data baru dan setiap titik pada *dataset* dapat menggunakan *euclidean distance*. Menghitung *Euclidean distance* adalah dengan rumus :

$$dist(X, Y) = \sqrt{\sum_n^{i=1} (xi - yi)^2} \quad (2.19)$$

Uraian rumus (2.19) di atas merupakan cara menghitung jarak antara titik baru dengan suatu titik pada *training set*. X merupakan data baru dan Y merupakan suatu titik pada *training set*. Xi merupakan setiap atribut variabel pada data baru dan Yi merupakan setiap atribut variabel pada titik *training set*. Algoritma *k-Nearest Neighbor* akan menghitung *euclidean distance* sebanyak N yaitu jumlah *row* pada *training set* untuk membandingkan jarak data baru dengan setiap data pada *training set*. Semakin kecil nilai *euclidean distance* maka 2 titik yang dibandingkan akan semakin dekat.

Selain *euclidean distance*, *Cosine Similarity* juga dapat digunakan untuk mengukur kemiripan 2 vektor. Menghitung *Cosine Distance* adalah dengan rumus :

$$sim(x, y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (2.20)$$

Uraian rumus (2.20) adalah cara lain untuk mengukur seberapa dekat / mirip 2 vektor.  $x \cdot y$  dibagian atas adalah merupakan *Dot Product* dari 2 vektor. *Dot Product* dapat dihitung dengan cara mengalikan tiap elemen diurutan yang sama lalu dijumlahkan. Bagian bawah merupakan perkalian dari panjang vektor x dan y. Rumus panjang vektor adalah berdasarkan rumus (2.21) dibawah yaitu penjumlahan setiap elemen vektor x yang dikuadratkan lalu diakar :

$$\vec{x} = \sqrt{\sum i^n (x_i)^2} \quad (2.21)$$

Menggunakan *euclidean distance* dapat diterapkan untuk atribut dengan tipe data numerik. Tetapi, untuk atribut nominal seperti jenis warna (biru,merah, hijau dan lain-lain) membutuhkan mekanisme berbeda untuk menentukan jarak terdekat. Menentukan angkanya adalah jika nilai atribut data baru sama dengan data pada *training set* maka jaraknya adalah 0. Jika nilai atribut data baru tidak sama dengan sebuah data pada *training set*, maka jaraknya adalah 1.

Tahap pertama dalam mencari label untuk data baru menggunakan *k-Nearest Neighbor* adalah dengan menentukan k tetangga terdekat. Cara menentukan k terbaik adalah dengan mencoba menghitung jarak pada setiap k. K terbaik dapat ditentukan dengan mencari akurasi tertinggi dari setiap percobaan k.

## Evaluasi Classification

Cara yang dapat digunakan untuk melakukan evaluasi terhadap model yang sudah dibuat adalah dengan menghitung jumlah perbandingan data yang benar dan salah. *Accuracy* adalah sebuah metode *performance metric* yang dapat digunakan untuk menghitung performa model. Cara menghitung akurasi yaitu :

$$Accuracy = \frac{Jumlah\ prediksi\ yang\ benar}{Jumlah\ semua\ prediksi}$$

*Error rate* adalah cara lain yang dapat digunakan sebagai menilai performa model berdasarkan nilai *error* yang dihasilkan. Cara menghitung *error rate* adalah sebagai berikut :

$$Errorrate = \frac{Jumlah\ prediksi\ yang\ salah}{Jumlah\ semua\ prediksi}$$

### 2.4.3 Clustering

*Clustering* merupakan teknik *unsupervised learning*. *Clustering* dapat mengelompokkan tiap data objek pada *dataset* menjadi beberapa kelompok. *Clustering* mengelompokkan data objek yang memiliki kemiripan menjadi satu kelompok dan data objek yang tidak memiliki kemiripan menjadi kelompok yang berbeda. Kemiripan dan ketidakmiripan dari data objek dapat ditentukan menggunakan atribut prediktor pada *dataset*. Ukuran yang dapat ditentukan untuk kemiripan tiap data objek adalah berdasarkan perhitungan jarak. Terdapat beberapa metode *clustering* yaitu :

- **Partitioning methods** : metode *clustering* dengan membuat k partisi dimana jumlah partisi ( $k$ )  $\leq N$  (jumlah data objek). Tiap partisi harus minimal berisi satu data objek. Contoh algoritma *partitioning* adalah *k-Means*
- **Hierarchical methods** : metode *clustering* dengan cara melakukan proses dekomposisi menjadi hirarki. Hirarki yang dihasilkan adalah tingkatan jumlah kelompok yang dihasilkan. Terdapat dua jenis *Hierarchical methods* yaitu *agglomerative* dan *divisive*. *Agglomerative* menerapkan *bottom-up approach*. *Agglomerative* berawal dari tiap data objek pada *dataset* menjadi data yang terpisah. Masing-masing data objek digabung satu per satu sehingga membentuk satu kelompok yang besar. *Divisive* menerapkan *top-bottom approach* yaitu setiap data objek yang dijadikan satu kelompok besar. Setiap iterasi akan dibagi menjadi kelompok-kelompok kecil sampai tiap data objek merupakan kelompok yang terpisah.
- **Density-based methods** : metode *clustering* dengan cara membagi kumpulan data objek menjadi bentuk kelompok yang lebih kompleks. Contoh algoritma *density* adalah *Density Based Spatial Clustering (DBSCAN)*. Cara kerjanya adalah setiap data objek yang merupakan titik secara acak diambil untuk ditentukan jenisnya. Suatu titik merupakan *core points* jika masih memiliki tetangga yang jaraknya lebih kecil dari epsilon. Jika tidak terdapat titik tetangga terdekat, titik dijadikan sebuah *border points*. Setelah itu berpindah ke titik tetangga *core points* terdekat lalu diulangi lagi sampai semua menjadi mempunyai *cluster* atau *outlier*

#### K-Means

*K-Means* adalah algoritma *clustering* jenis *partitioning*. Algoritma ini mengelompokkan data objek dengan memaksimalkan kemiripan. Ukuran kemiripan antar data objek dapat diukur menggunakan jarak. Jumlah kelompok akan ditentukan berdasarkan nilai  $k$  yang ditentukan. Setelah nilai  $k$  telah ditentukan, maka akan dibuat data objek secara *random* atau mengambil salah satu data pada *dataset* yang disebut sebagai *centroid*. *Centroid* merupakan nilai tengah yang merepresentasikan tiap *cluster*. Tiap data objek akan dihitung jaraknya dengan setiap *centroid* yang diinisialisasi. Data objek akan dimasukkan ke kelompok *cluster* yang memiliki jarak terkecil dengan *centroid*. Jarak data objek dengan *centroid* dapat dihitung dengan menggunakan *euclidean distance*:

$$dist(X, Y) = \sqrt{\sum_n^{i=1} (xi - yi)^2} \quad (2.22)$$

Persamaan (2.22) menghitung jarak antara 2 data objek.  $X$  adalah data objek pertama dan  $Y$  adalah data objek kedua.  $\sum$  adalah melakukan iterasi dan penjumlahan setiap atribut. Setiap atribut masing-masing data objek akan dikurang lalu dipangkatkan. Hasil perpangkatan selisih setiap atribut akan diakar pangkat dua sehingga menjadi jarak dari kedua data objek

Langkah-langkah dalam melakukan *k-Means clustering* adalah :

1. Tentukan jumlah  $K$  (jumlah *cluster* / kelompok)
2. Untuk tiap *cluster*, buat satu data objek sebagai titik tengah yang disebut sebagai *centroid*. Inisialisasi *centroid* dapat ditentukan dengan *random* / mengambil salah satu titik dari *data objek*

3. Untuk setiap data objek, hitung jarak menggunakan *euclidean distance* dengan setiap *centroid*. Data objek akan masuk ke kelompok yang memiliki *centroid* dengan jarak terpendek
4. Atur ulang posisi *centroid* dengan menghitung rata-rata tiap atribut yang termasuk dalam *cluster*
5. Ulangi tahap tiga sampai tiap data objek tidak mengalami perubahan dalam menetapkan *cluster*

### Agglomerative Nesting (AGNES)

*Agglomerative clustering* adalah algoritma *clustering* jenis *hierarchical*. Algoritma ini merupakan metode *bottom-up* yaitu setiap data objek sebagai masing-masing *cluster*. Secara iteratif menghitung jarak tiap data objek. Dua data objek dengan jarak terkecil akan digabung menjadi sebuah *cluster*. Proses ini diulangi sampai semua data objek tergabung menjadi satu *cluster*. Jarak tiap titik dapat dihitung menggunakan *euclidean distance* pada Persamaan (2.22) atau menggunakan *cosine distance* pada Persamaan (2.20).

## Evaluasi Clustering

Pengelompokan data dapat dilakukan menggunakan *clustering*, tetapi pengaruh pemilihan jumlah K / berapa kelompok pada algoritma dan metrik jarak yang digunakan dapat berpengaruh terhadap seberapa baik / seberapa mirip anggota dari *cluster* yang dihasilkan. Terdapat beberapa cara untuk mengukur apakah *clustering* yang dilakukan sesuai kebutuhan / akurat. Terdapat 2 cara dalam mengukur *clustering* yaitu *intracluster* dan *silhouette plot*

### Intracluster

*Intracluster* adalah nilai yang merepresentasikan apakah sebuah *cluster* dikelompokkan dengan baik. *Cluster* yang baik adalah *cluster* yang jarak tiap anggotanya dekat / mirip. Cara menghitung *intracluster* sebuah titik adalah dengan menghitung jarak titik tersebut dengan titik representatif dari *cluster*. Titik representatif adalah *centroid* yaitu rata-rata dari tiap anggota pada *cluster* yang sama. Berikut adalah rumus cara menghitung *intracluster* yaitu :

$$intracluster = \frac{\sum_{i=1}^n (JarakTitik(i) dengan Centroid)}{JumlahAnggotapadaClustertersebut} \quad (2.23)$$

Berdasarkan persamaan (2.23) di atas, untuk menghitung *intracluster* sebuah kelompok adalah dengan menghitung rata-rata jarak tiap titik yang merupakan kelompok tersebut dengan *centroid*. Perhitungan jarak dapat menggunakan *euclidean distance* atau *cosine distance*.

### Silhouette Plot

*Silhouette Plot* merupakan teknik yang digunakan untuk mengukur kualitas sebuah *cluster*. Teknik ini menggabungkan kualitas internal (Intracluster) dan eksternal (Intercluster). Menghitung *silhouette plot* adalah dengan rumus berikut yaitu :

$$S(i) = \frac{(b(i) - a(i))}{\max(a(i), b(i))} \quad (2.24)$$

Persamaan (2.24) di atas adalah cara menghitung kualitas sebuah titik pada *cluster*.  $S(i)$  adalah nilai *silhouette* pada sebuah titik pada *cluster*.  $b(i)$  adalah nilai rata-rata jarak titik tersebut dengan titik lain dalam sebuah *cluster* yang sama.  $a(i)$  adalah nilai rata-rata jarak titik tersebut dengan titik tetangga. Titik tetangga adalah tiap titik terdekat dengan titik yang sedang dihitung tetapi masing-masing berasal dari *cluster* yang berbeda.

Untuk mendapatkan kualitas *cluster* secara keseluruhan adalah dengan menghitung rata-rata dari nilai *silhouette plot* dari tiap titik dataset. Nilai *silhouette plot* bisa berada diantara 0.0 sampai 1.0. Semakin besar nilai *cluster* maka semakin baik *clustering* yang dilakukan.

## 2.5 Data Visualization [1]

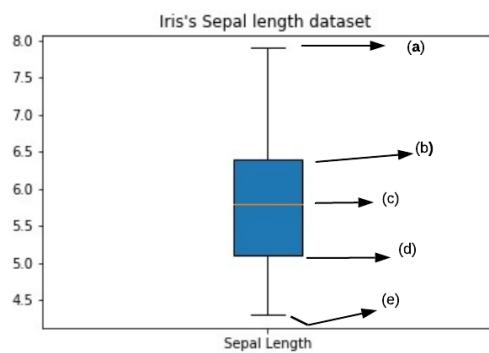
Visualisasi data adalah suatu metode untuk merepresentasikan data. Visualisasi data menggunakan dengan jumlah yang banyak untuk ditampilkan secara grafis. Visualisasi data akan membantu *user* untuk membaca data lebih mudah. Dengan visualisasi, maka kita juga akan mendapatkan informasi mengenai hubungan antara data. Visualisasi data akan membantu meringkas data dan memberikan pengetahuan baru.

Penggunaan visualisasi data dapat digunakan pada saat *data preprocessing* maupun setelahnya. Pada tahap *data preprocessing*, visualisasi data dapat dimanfaatkan untuk membantu proses memeriksa data untuk memastikan kebersihannya. Visualisasi data tidak hanya dilakukan sekali tetapi membutuhkan beberapa kali visualisasi untuk membantu menganalisis tren dari perubahan data itu sendiri.

Visualisasi data dapat dibedakan berdasarkan tipe data yang dianalisis, jumlah atribut yang digunakan, dan bentuk data yang dibutuhkan. Terdapat beberapa teknik yang dilakukan untuk melakukan visualisasi data yaitu *Box plot*, *Histogram*, *Scatter plot* dan *Pie chart*

### 2.5.1 Boxplot

*Boxplot* adalah teknik *data visualization* yang dapat digunakan untuk menampilkan persebaran nilai satu atribut. Masukan data dari grafik ini adalah kumpulan atribut numerik. Teknik ini dapat membantu memberikan informasi mengenai nilai-nilai yang dapat digunakan untuk analisis data. Suatu *Boxplot* akan menampilkan rata-rata(*mean*), median (*Q2*), nilai maksimum, nilai minimum, kuartil bawah(*Q1*)dan kuartil atas(*Q3*). Berikut adalah contoh *box plot* yang dihasilkan berdasarkan *dataset* iris pada kolom *sepal length* yaitu :



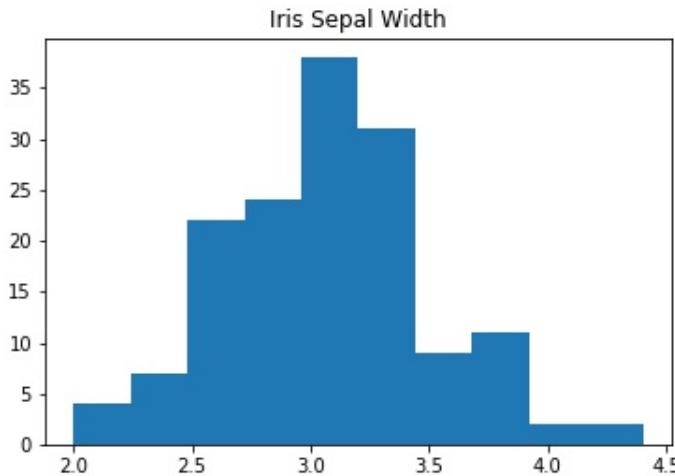
Gambar 2.4: Boxplot dataset iris

Gambar 2.4 menunjukkan kesimpulan informasi dari persebaran *dataset* iris pada kolom *sepal length*. Berikut adalah deskripsi dari tiap huruf :

- (a) nilai maksimum
- (b) kuartil atas (Q3)
- (c) median (Q2)
- (d) kuartil bawah (Q1)
- (e) nilai minimum

### 2.5.2 Histogram

*Histogram* adalah teknik *data visualization* yang digunakan untuk melihat persebaran data suatu atribut. Masukan dari Data yang ditampilkan akan dibagi dalam sebuah *bin* / interval kelas. Teknik ini akan mengelompokkan jumlah *data object* berdasarkan nilai interval yang ditentukan dan sudah terurut. Berikut adalah contoh *histogram* dari *dataset iris sepal width* yaitu :



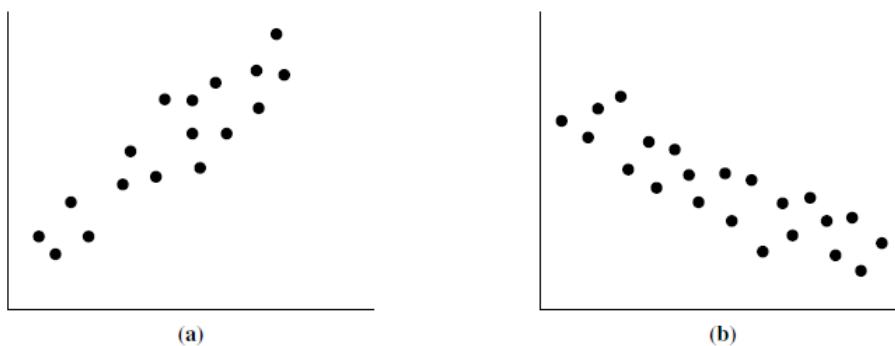
Gambar 2.5: Histogram dataset iris

Histogram yang ditampilkan pada gambar diatas adalah berdasarkan *dataset iris*. *Dataset iris* merupakan data spesies bunga yang paling sering digunakan. Berdasarkan gambar 2.5, histogram satu variabel akan menunjukkan grafik dengan koordinat 2 dimensi yaitu koordinat x dan y. Nilai pada koordinat x menunjukkan tiap interval kelas. Nilai pada koordinat y menunjukkan frekuensi tiap interval data. Tiap batang pada histogram menampilkan tiap interval kelas. Tinggi batang menunjukkan frekuensi pada tiap kelas.

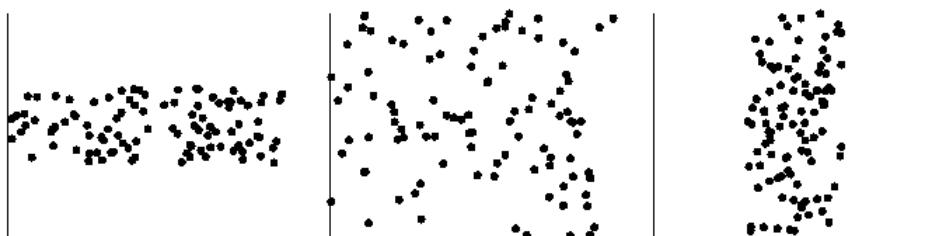
Pada contoh gambar 2.5, terdapat 10 *bins* / batang yang masing-masingnya memiliki rentang sekitar nilai interval 0,25. Pada *bin* pertama, terdapat jarak nilai dari 2.0 sampai kurang lebih 2,25. Aturan juga diterapkan pada *bin* 2 sampai 10 yaitu menyerupai *bin* pertama.

### 2.5.3 Scatter Plot

*Scatter plot* adalah teknik *data visualization* yang dapat digunakan untuk melihat korelasi / keterhubungan antara 2 variabel data. *Scatter plot* akan menggambarkan atribut prediktor (sebab) pada koordinat X dan menggambarkan atribut respon (akibat) pada koordinat Y. Hasil *scatter plot* akan memberi penjelasan apakah atribut prediktor memiliki hubungan dengan atribut respon. Terdapat beberapa jenis *Scatter plot* berdasarkan hasil korelasinya. Terdapat *Scatter plot* yang memiliki korelasi positif, korelasi negatif dan tidak memiliki korelasi. Berikut adalah contoh gambar *scatter plot* yaitu



Gambar 2.6: Scatter plot dengan korelasi positif (a) dan negatif (b)

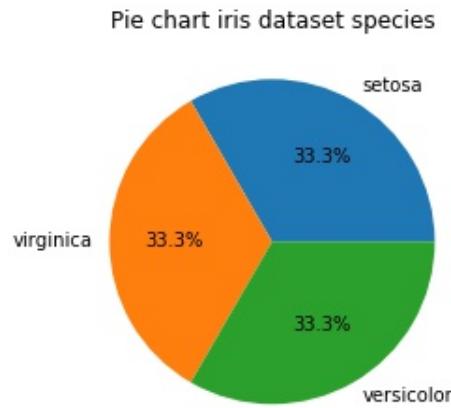


Gambar 2.7: Scatter plot tanpa korelasi

Berdasarkan Gambar 2.6, *scatter plot* dengan korelasi terbentuk jika perubahan atribut pada koordinat X memengaruhi koordinat Y. Korelasi positif terjadi ketika semakin meningkatnya nilai pada koordinat X, maka koordinat Y akan meningkat juga. Korelasi negatif terjadi ketika perubahan koordinat X akan membuat nilai pada koordinat Y berubah berbanding terbalik. Gambar 2.7 menggambarkan atribut koordinat X dan koordinat Y tidak memiliki korelasi.

#### 2.5.4 Pie Chart

*Pie Chart* adalah teknik *data visualization* untuk melihat frekuensi persebaran *categorial data*. *Pie chart* menggunakan bentuk lingkaran sebagai keseluruhan semua kumpulan data. Masing-masing potongan bagian pada *pie chart* merupakan persentase banyak masing-masing nilai kategori. Berikut adalah gambaran *pie chart* menggunakan *dataset iris*.



Gambar 2.8: Pie chart dataset iris

Berdasarkan Gambar 2.8, data yang digunakan untuk divisualisasikan adalah *atribut species* dataset iris. Pada *dataset* terdapat atribut *species* yang memiliki kemungkinan nilai yaitu *setosa*, *virginica*, *versicolor*. Tiap warna pada *pie chart* merepresentasikan masing-masing kategori *species*. Jumlah data pada atribut *species* adalah 150, jumlah masing-masing *species* adalah 50 sehingga presentasenya sama.

## 2.6 One Hot Encoding [2]

*One hot encoding* adalah metode yang dapat digunakan untuk mengubah data kategorial menjadi numerik. Nilai label yang dimiliki tipe data kategorial tidak bisa dihitung / kuantifikasi jika data tersebut digunakan pada *machine learning*. *Machine learning* sering memanfaatkan perhitungan jarak pada data numerik untuk menghitung kemiripan 2 data objek. *One hot encoding* tidak dapat digunakan pada data ordinal. Data ordinal yaitu tipe data kategorial yang tiap nilainya memiliki sifat keterurutan. *One hot encoding* cocok untuk nilai kategorial yang tidak memiliki unsur keterurutan / sederajat.

## 2.7 Web Scraping [4]

*Web Scraping* adalah metode untuk mengambil data dari sebuah *web page*. *Scraping* dapat dilakukan untuk mengambil elemen yang terdapat pada halaman web. *Scraping* dapat dilakukan dengan *web scraper* dan *web crawler*. *Web scraping* akan membantu dalam mengumpulkan data yang banyak dari kumpulan *website*. Dasar perintah dalam *scraping* adalah :

1. Mengambil URL sebuah halaman web
2. Melakukan Parsing pada sebuah halaman
3. Melakukan ekstraksi data URL
4. Memasukan URL pada *queue*

Berdasarkan tahap yang dijelaskan sebelumnya, proses *web scraping* ini akan diulang selama masih ada halaman / URL pada *queue* yang terhubung dari URL sebelumnya dan mengambil data pada halaman.

## 2.8 Istilah Dalam Bisnis Film

Terdapat beberapa istilah yang perlu dipahami mengenai bisnis industri film yaitu :

- Budget : biaya yang dikeluarkan untuk memproduksi film
- Revenue / pendapatan kotor : pemasukan yang diterima dari hasil penjualan film
- Profit / pendapatan bersih : hasil pengurangan *revenue* dan *budget*
- *Return Of Investment (ROI)* : rasio yang mengukur efisiensi sebuah investasi dengan membandingkan *profit* dengan *budget* yang dikeluarkan

## BAB 3

# ANALISIS

Bab Analisis berisi pemahaman lebih lanjut berdasarkan teori-teori yang sudah dijelaskan dari Bab Landasan Teori. Pada bagian ini akan dijelaskan analisis lebih lanjut terhadap masalah yang sudah dijelaskan pada latar belakang. Teori yang sudah dijelaskan sebelumnya pada Landasan Teori akan diberi contoh permasalahan dan cara penyelesaiannya.

### 3.1 Analisis Masalah

Penelitian "*Analisis Kesuksesan Film Menggunakan Data Mining*" merupakan penelitian yang akan menggunakan *dataset* yang berasal dari *Kaggle*<sup>1</sup>. *Dataset* yang digunakan adalah kumpulan film populer dari tahun 2006 sampai 2016. Terdapat informasi yang terkait dengan sebuah film seperti judul, aktor, *genre*, *rating* dan lain-lain. Penjelasan lebih lengkap mengenai *dataset* dan proses penelitian akan dibahas pada Bab 4.

Berdasarkan Landasan Teori yang dijelaskan pada Bab 2 mengenai tahap *Data Mining*, penelitian ini akan menerapkan langkah-langkah *Data Mining* untuk mendapatkan pola yang menarik. Pola yang menarik adalah informasi yang dapat menjawab masalah yang sudah dijelaskan pada bagian Latar Belakang. Penelitian ini akan menghasilkan pengujian faktor yang mempengaruhi kesuksesan film dengan menggunakan *Data Mining* dan *Machine Learning*.

Penelitian ini akan membuat beberapa program terpisah yang akan memproses *dataset* yang sudah dipilih. Program yang dihasilkan bukan suatu program dengan antarmuka yang bisa menerima *input* dan mengeluarkan *output*, Tetapi hasil eksperimen berdasarkan *dataset* dengan menerapkan langkah *data mining* dan gambar visualisasi data. Penelitian ini akan menggunakan bahasa pemrograman *Python* dan memanfaatkan beberapa *library* pada *Python* yang akan membantu penelitian pada Bab 4. Berikut adalah kumpulan *library* yang digunakan untuk penelitian :

- *Pandas* : melakukan operasi pemrosesan analisis data pada tabel bernama *DataFrame*
- *NumPy* : melakukan beberapa fungsi hitung matematika pada kumpulan *Array*
- *Matplotlib* : melakukan analisis dalam bentuk visualisasi data
- *Seaborn* : melakukan visualisasi data dan menyediakan fungsi kompleks yang tidak dimiliki *matplotlib*
- *Scikit-learn* : melakukan analisis data dengan melakukan algoritma pada *Machine Learning*
- *Pickle* : menyimpan model / algoritma *Machine Learning* yang sudah diproses sehingga tidak perlu melakukan *training* untuk melakukan prediksi
- *os* : melakukan fungsi yang dapat berkomunikasi dengan sistem operasi seperti membaca dan menyimpan data

---

<sup>1</sup>PromptCloud, (2017) "IMDB data from 2006 to 2016" <https://www.kaggle.com/PromptCloudHQ/imdb-data> . 18 April 2020.

- *shutil* : melakukan operasi manipulasi dan memproses *file* pada sistem operasi
- *requests* : melakukan HTTP *Request* untuk melakukan pemanggilan API mengambil data yang tambahan yang akan digunakan saat penelitian
- *json* : melakukan konversi JSON (sintaks untuk menyimpan data) secara universal dan mengubah menjadi struktur data di *Python*

## 3.2 Tahap Pengerjaan Penelitian

Berdasarkan penjelasan pada bagian sebelumnya tentang Analisis Masalah, dibuat rencana yang berisi langkah-langkah yang dilakukan selama analisis data pada Bab 4. Berdasarkan ilustrasi gambar proses *Data Mining* pada Landasan Teori Bab 2, proses *Data Mining* dijelaskan sebagai sebuah siklus. Hasil sebuah siklus dapat dijadikan pengantar analisis pada siklus selanjutnya. Penelitian ini dibagi menjadi 3 siklus iterasi yaitu :

- Analisis Data Utama
- Analisis Data Tambahan IMDB
- Analisis Data Media Sosial

### 3.2.1 Analisis Data Utama

Penelitian dimulai dengan menggunakan *dataset* yang sudah dipilih. Pada analisis tahap ini akan melakukan *Data Cleaning*, *Data Selection*, analisis visualisasi dan percobaan prediksi kesuksesan yaitu *revenue* / pendapatan kotor. Percobaan prediksi akan menggunakan 2 algoritma regresi dan membandingkan performa yang dihasilkan.

Hasil prediksi regresi pada data utama akan dilanjutkan dengan analisis yang lebih detail. Akan dilakukan analisis *clustering* untuk mengelompokkan film dengan karakteristik yang sama. Setiap film memiliki sifat berbeda dalam mendefinisikan kesuksesannya. Fitur yang menjadi penentu kelompoknya adalah aktor dan *genre*.

### 3.2.2 Analisis Data Tambahan IMDB

Hasil dan performa analisis data utama akan digunakan sebagai latar belakang bagi analisis data tambahan. Pada tahap ini akan dilakukan pengumpulan data tambahan. Tahap pengumpulan data tambahan akan dilakukan dengan *Scraping* dan mengakses API dari situs IMDB. Data tambahan akan diintegrasikan ke *dataset* awal sesuai dengan tahap *Data Mining* yaitu *Data Integration*. Data tambahan akan dianalisis sesuai dengan tahap sebelumnya untuk menguji apakah data tambahan dapat membantu meningkatkan akurasi / pengaruh dalam memprediksi kesuksesan film.

Tahap *Data Transformation* juga dilakukan dimana kolom pada *dataset* utama dapat digabung dengan kolom tambahan sehingga mendapatkan kolom baru yang dapat dianalisis. Prediksi kolom baru akan dilakukan untuk mengetahui apakah terdapat faktor yang memiliki korelasi yang tinggi dengan apa yang ingin diprediksi.

### 3.2.3 Analisis Data Sosial Media

Penonton / penikmat film juga memiliki peran penting dalam menentukan kesuksesan film. *Analisis* data yang sudah dilakukan sebelumnya hanya terikat dengan informasi yang melekat pada filmnya saja seperti aktor, *genre*, *budget* dan *rating*. Media sosial merupakan salah satu cara pembuat film untuk mempromosikan hasil karyanya kepada penonton dengan merilis seperti *trailer* dan poster. Pada analisis data media sosial akan dilakukan pengumpulan data seperti *Youtube* dan *Instagram*.

Data media sosial tambahan yang sudah dikumpulkan akan dianalisis lebih lanjut sesuai dengan tahap *Data Mining*. Pengujian akan dilakukan apakah data media sosial dapat membantu meningkatkan akurasi prediksi kesuksesan film. Data media sosial juga akan dianalisis apakah memiliki korelasi yang tinggi dengan pendapatan.

### 3.3 Analisis Penerapan Data Cleaning

Penjelasan Bab 2 Landasan Teori mengenai *Data cleaning* menyebutkan bahwa terdapat beberapa jenis data kotor. Data yang tidak sesuai dan *missing value* adalah salah satu masalah *Data Cleaning*. Cara mengatasi hal tersebut adalah dengan mengabaikan data tersebut. Berikut adalah contoh data tabular *dataset* yang terdapat data kotor didalamnya.

Nama	Umur	Alamat
Jono	20	Jl. Ciumbuleuit
Agus	2000	Jl Bukit Jarian
Shinta	30	Jl Kemanggisan
Jl Bekasi	NaN	-

Annotations pointing to specific rows in the dataset table:

- A black arrow points to the second row (Agus, 2000, Jl. Ciumbuleuit) with the text "Baris dengan umur yang tidak sesuai".
- A black arrow points to the fifth row (Jl Bekasi, NaN, -) with the text "Baris dengan Missing Value".

Gambar 3.1: Contoh dataset yang memiliki data kotor

berdasarkan Gambar 3.1 yang sudah ditunjuk panah, terdapat baris yang tidak sesuai seperti baris terakhir yang pada kolom 'Nama' tidak terisi yang sesuai. Baris ketiga juga merupakan data kotor dikarenakan isi pada kolom 'Umur' tidak sesuai dengan rentang umur manusia pada umumnya sehingga data tersebut harus diabaikan.

*Library Pandas* menyediakan fungsi untuk mendeteksi baris pada data tabel yang terdapat nilai *Nan* pada kolomnya. Gambar 3.2 adalah potongan kode untuk membaca *DataFrame* dari sebuah file lalu menghilangkan baris yang *NaN*.

```

1 import pandas as pd
2
3 # read dataset
4 dataset = pd.read_csv('table.csv')
5
6 #remove row with Nan in dataset
7 dataset = dataset.dropna()

```

Gambar 3.2: Kode analisis data cleaning

### 3.4 Analisis Penerapan Data Transformation

Berdasarkan penjelasan Bab 2 Landasan Teori tentang *Data Transformation*, beberapa cara yang dapat dilakukan adalah *Attribute Construction* dan *Normalization*. Akan diberikan beberapa contoh penerapan *Data Transformation*. Berikut adalah contoh *dataset* dari penjualan barang pada toko oleh-oleh yang akan dibuat kolom baru.

### 3.4.1 Attribute Construction

Tabel 3.1: Tabel contoh yang perlu diubah

Barang	Harga Satuan	Jumlah Terjual	Total Terjual
Slondok	2000	20	40000
Keripik Singkong	2500	5	12500
Getuk	10000	1	10000

Tabel 3.1 menunjukkan hasil penjualan dari tiap barang. Dalam menentukan barang mana yang menguntungkan ternyata dibutuhkan persenan kontribusi keuntungan dari tiap item. Dibutuhkan kolom baru dari tiap barang untuk menghitung kontribusi. Kontribusi dapat dihitung menggunakan rumus di bawah.

$$KontribusiBarang(i) = \frac{TotalPenjualanBarang(i)}{TotalPenjualanSemuaBarang} \quad (3.1)$$

Persamaan 3.1 dapat digunakan untuk menghitung kontribusi setiap barang. Total penjualan semua barang adalah Rp.62500. Tabel 3.2 adalah hasil tabel yang baru setelah ditambahkan kolom baru bernama kontribusi.

Tabel 3.2: Tabel dengan kolom baru setelah *Data Transformation*

Nama Barang	Harga Satuan	Jumlah Terjual	Total Penjualan	Kontribusi
Slondok	2000	20	40000	0.64
Keripik Singkong	2500	5	12500	0.2
Getuk	10000	1	10000	0.16

Manipulasi sebuah data tabel dapat dilakukan oleh *Library Pandas*. Gambar 3.3 adalah kode implementasi menggunakan bahasa *Python* untuk melakukan *Data Transformation*.

```

1 #import library yang dibutuhkan
2 import pandas as pd
3
4 # membaca data tabel di file
5 dataset = pd.read_csv('items.csv')
6
7 # menghitung total penjualan semua item
8 alltotalpenjualan = dataset['totalpenjualan'].sum()
9
10 # membuat kolom baru berdasarkan kolom lain dan menghitung kontribusi item pada masing masing
11 dataset['kontribusi'] = dataset['totalpenjualan'] / alltotalpenjualan

```

Gambar 3.3: Kode contoh Attribute Construction

### 3.4.2 Normalization

Penjelasan mengenai *Normalization* pada Bab 2 yaitu salah satunya adalah dengan menggunakan *MinMaxNormalization*. Persamaan (3.2) adalah rumus yang dapat digunakan untuk melakukan normalisasi.

$$z = \frac{x - min(x)}{max(x) - min(x)} \quad (3.2)$$

Persamaan (3.2) di atas merupakan cara mengubah sebuah angka dari kumpulan data dalam bentuk yang sudah dinormalisasi.  $x$  adalah nilai yang ingin rubah dalam bentuk normalisasi.  $min(x)$  adalah nilai terkecil dalam kumpulan data dan  $max(x)$  adalah nilai terbesar. Selanjutnya akan

diberikan contoh dalam mengubah kumpulan dalam bentuk yang sudah dinormalisasi dalam bentuk tabel.

Tabel 3.3: Tabel Dataset yang ingin dinormalisasi

lamabelajar	nilai
2	90
4	70
6	86
3.5	85

Tabel 3.3 di atas merupakan tabel performa siswa yang akan dianalisis lebih lanjut sehingga perlu dilakukan normalisasi. Berikut adalah hasil tabel setelah dinormalisasi menggunakan *MinMaxNormalization* berdasarkan persamaan (3.2).

Tabel 3.4: Tabel dataset setelah dinormalisasi

Lama Belajar	Nilai	Norm Lama Belajar	Norm Nilai
2	90	0	1
4	70	0.5	0
6	86	1	0.8
3.5	85	0.375	0.75

Tabel 3.4 di atas merupakan hasil tabel *dataset* setelah dinormalisasi. Pada kolom 'Norm Lama Belajar' dan 'Norm Nilai' adalah nilai setelah sudah dirubah. *Normalization* dapat dilakukan sebelum tahap pemodelan *Machine Learning*. *Library Scikit Learn* pada *Python* menyediakan fungsi untuk mengonversi kumpulan nilai pada tabel dalam bentuk yang sudah dinormalisasi. Gambar 3.4 adalah contoh potongan kode dalam melakukan normalisasi.

```

1 | import pandas as pd
2 | from sklearn.preprocessing import MinMaxScaler
3 |
4 | dataset = pd.read_csv('nilai.csv')
5 |
6 | minmaxScaler = MinMaxScaler(feature_range = (0,1))
7 | x_norm = minmaxScaler.fit_transform(dataset)

```

Gambar 3.4: Kode contoh *Normalization*

## 3.5 Analisis Penerapan Data Integration

Salah satu cara yang dapat dilakukan dalam melakukan *Data Integration* adalah penggabungan 2 tabel menggunakan operasi *join* seperti pada basis data. Berikut terdapat 2 tabel terpisah yang memiliki relasi.

Tabel 3.5: Tabel kumpulan buku

ID_Buku	Judul Buku	Nama Pengarang
1	Pengantar Basis Data	Bill Gates
2	Pemrograman Dasar	Bill Gates
3	Terampil Memasak	Gordon Ramsey

Tabel 3.6: Tabel pengarang

Nama Pengarang	Kota Kelahiran
Bill Gates	Bekasi
Gordon Ramsey	Bandung

Tabel 3.5 dan 3.6 dapat digabung agar sebuah buku dapat mengetahui kota asal pengarangnya. Tabel buku berhubungan dengan tabel pengarang karena tabel buku memiliki kolom "Nama Pengarang" untuk mengetahui data pengarang. Berikut adalah tabel 3.7 yang sudah tergabung.

Tabel 3.7: Tabel buku dan pengarang sudah tergabung

ID_Buku	Nama Buku	Nama Pengarang	Kota kelahiran
1	Pengantar Basis Data	Bill Gates	Bekasi
2	Pemrograman Dasar	Bill Gates	Bekasi
3	Terampil Memasak	Gordon Ramsey	Bandung

*Library Pandas* pada *Python* dapat digunakan untuk dilakukan untuk melakukan operasi *join*. Berikut adalah potongan contoh kode pada *Python* untuk melakukan operasi *merge* pada Gambar 3.5.

```

1 import pandas as pd
2
3 # baca tabel buku
4 buku = pd.read_csv('buku.csv')
5
6 # baca tabel pengarang
7 pengarang = pd.read_csv('pengarang.csv')
8
9 # lakukan operasi join
10 dataset= pd.merge(buku, pengarang, how ='left', on ='nama_pengarang')
```

Gambar 3.5: Kode contoh *integration*

## 3.6 Analisis Penerapan Data Selection

Berdasarkan penjelasan pada Bab 2 mengenai *Data Selection*, *feature selection* adalah cara untuk menentukan fitur sebelum dimodelkan dengan *machine learning*. *Pearson correlation* adalah salah satu teknik untuk menguji korelasi antara 2 fitur. Akan diberikan ilustrasi dalam menentukan / memilih kolom yang memiliki korelasi kuat. Berikut adalah sebuah contoh sederhana dalam memilih kolom yang tepat untuk analisis. Diberikan contoh sebuah tabel penelitian belajar siswa.

Tabel 3.8: Tabel data performa belajar siswa

Lama Belajar dalam Jam	Jam memulai Belajar	Nilai Ujian
2	12	50
3	10	65
4	11	70

Berdasarkan Tabel 3.8, ingin dilakukan analisis untuk mengetahui apa yang mempengaruhi performa siswa khususnya dalam memperoleh nilai yang baik. Untuk menemukan kolom / fitur

yang berpengaruh, dilakukan perhitungan menggunakan *pearson correlation* untuk membandingkan antara kolom lama belajar dan jam memulai belajar. Berikut adalah hasil perhitungan *pearson correlation*.

Tabel 3.9: Tabel hasil perhitungan pearson

Pearson (Lama Belajar , Nilai)	Pearson (Jam Mulai , Nilai)
0.96	-0.72

Tabel 3.9 di atas merupakan hasil perhitungan *pearson correlation*. Nilai korelasi antara kolom lama belajar dan hasil nilai memiliki nilai 0.96. Nilai tersebut memiliki korelasi positif yang kuat. Nilai korelasi positif yang tinggi mendekati 1.0 menejelaskan bahwa semakin banyak waktu belajar yang dilakukan siswa, maka semakin tinggi kemungkinan siswa untuk mendapatkan nilai yang bagus. Kolom jam mulai dan nilai memiliki korelasi negatif. Berdasarkan itu juga dapat dimaknai bahwa semakin malam siswa belajar maka semakin menurun nilai yang diperoleh. Dapat disimpulkan untuk meningkatkan kemungkinan siswa mendapat nilai bagus adalah dengan meningkatkan frekuensi belajar untuk menguasai materi yang diuji.

Library *Sci Py* pada *Python* menyediakan fungsi *pearson* untuk menghitung korelasi antara 2 kolom nilai. Fungsi *pearsonr(colA , colB)* menerima 2 *input* berupa kolom pertama dan kolom kedua. Gambar 3.6 adalah potongan kode untuk menghitung korelasi.

```

1 import pandas as pd
2 from scipy.stats import pearsonr
3 dataset = pd.read_csv('nilaidataset.csv')
4 nilaipearson_lamabelajar_nilai = pearsonr(dataset['lamabelajar'] , dataset['nilai'])[0]
5 nilaipearson_jambelajar_nilai = pearsonr(dataset['jambelajar'] , dataset['nilai'])[0]
```

Gambar 3.6: Kode contoh *pearson correlation*

## 3.7 Analisis Penerapan Linear Regression

*Linear regression* adalah algoritma untuk melakukan prediksi data kontinu dengan menghasilkan persamaan *linear*. Akan dilakukan contoh perhitungan menggunakan *linear regression*. Terdapat *dataset* dengan 2 variabel yaitu suhu ruangan dan jumlah cacat produksi. Diberikan sebuah *dataset* contoh regresi.

Tabel 3.10: Tabel dataset regresi

Biaya Promosi (X)	Volume Penjualan (Y)
12	56
14	62
13	60
12	61
15	65
13	66
14	60
15	63
13	65
14	62

Biaya promosi pada Tabel 3.10 merupakan variabel prediktor / independen (X). Volume penjualan pada Tabel 3.10 merupakan variabel dependen / respon (Y). Tujuan *linear regression* adalah memprediksi volume penjualan berdasarkan biaya promosi yang dikeluarkan. Menggunakan persamaan *linear regression*, maka kita harus mencari a (konstanta) dan b (koefisien) pada persamaan 2.8. Berikut adalah tabel perhitungan  $\sum y$ ,  $\sum x^2$ ,  $\sum xy$ ,  $(\sum x)^2$  pada Tabel 3.11.

Tabel 3.11: Perhitungan komponen konstanta dan koefisien

no	x	y	$x^2$	xy
1	12	56	144	672
2	14	62	196	868
3	13	60	169	780
4	12	61	144	732
5	15	65	225	975
6	13	66	169	858
7	14	60	196	840
8	15	63	225	945
9	13	65	169	845
10	14	62	196	868
$\Sigma$	135	620	1833	8383

Setelah menghitung komponen untuk konstanta dan koefisien, maka akan diterapkan pada rumus konstanta dan koefisien sesuai Persamaan 2.10.

$$a = \frac{(620)(1833) - (135)(8383)}{10(1833) - 18255} = 45.29 \quad (3.3)$$

$$b = \frac{10(8383) - (135)(620)}{10(1833) - 18255} = 1.24 \quad (3.4)$$

Koefisien dan konstanta yang diperoleh pada Persamaan 3.4 dan 3.3 dapat diterapkan pada rumus *linear regression* pada Persamaan 2.8. Persamaan *linear regression* dapat dibuat.

$$Y = 45.29 + 1.24X \quad (3.5)$$

Menggunakan Persamaan 3.5 yang sudah diperoleh, maka prediksi volume penjualan menggunakan *linear regression* dapat dihitung menggunakan biaya promosi. Berikut adalah tabel perbandingan antara volume penjualan asli (Y) dan volume penjualan prediksi *linear regression* ( $\hat{Y}$ ) pada Tabel 3.12.

Tabel 3.12: Tabel perbandingan volume penjualan prediksi dan volume penjualan asli

Biaya Promosi (X)	Volume Penjualan (Y)	Volume Penjualan Prediksi ( $\hat{Y}$ )
12	56	60.14
14	62	62.62
13	60	61.38
12	61	60.14
15	65	63.86
13	66	61.38
14	60	62.61
15	63	63.86
13	65	61.38
14	62	62.61

Library *Scikit learn* pada *Python* menyediakan fungsi untuk melakukan prediksi menggunakan algoritma *Linear Regression*. Kelas *Linear Regression* tersedia di package *sklearn.model*. Pemodelan *linear regression* menerima *input* kolom *predictor* (Biaya Promosi) dan kolom *response* (Volume Penjualan). Gambar 3.7 adalah potongan kode untuk melakukan *Linear Regression*.

```

1 import sklearn.linear_model import LinearRegression
2
3 x = dataset['biayapromosi']
4 y = dataset['volumepenjualan']
5
6 # membuat kelas linear regression
7 model = LinearRegression()
8
9 # melakukan training
10 model.train(x,y)
11
12 #melakukan prediksi nilai baru
13 ypred = model.predict(x)

```

Gambar 3.7: Kode contoh *linear regression*

## 3.8 Analisis Penerapan Polynomial Regression

Berdasarkan penjelasan pada Landasan Teori Bab 2 tentang *Polynomial Regression*, maka diberikan contoh perhitungan prediksi menggunakan *polynomial regression*. Terdapat *dataset* dengan 2 variabel yaitu suhu ruangan dan jumlah cacat produksi.

Tabel 3.13: Tabel dataset polinomial

Biaya Promosi (X)	Volume Penjualan (Y)
12	56
14	62
13	60
12	61
15	65
13	66
14	60
15	63
13	65
14	62

Biaya promosi pada Tabel 3.13 merupakan variabel prediktor / independen (X). Volume penjualan pada Tabel 3.10 merupakan variabel dependen / respon (Y). Tujuan *polynomial* adalah memprediksi volume penjualan berdasarkan biaya promosi yang dikeluarkan. Menggunakan persamaan *polynomial regression*, maka perlu menghitung komponen untuk persamaan (2.12) hubungan berupa  $\sum x$  ,  $\sum y$  ,  $\sum x^2$  ,  $\sum x^3$  ,  $\sum x^4$  ,  $\sum xy$  dan  $\sum x^2y$  pada Tabel 3.14.

Tabel 3.14: perhitungan komponen matriks hubungan polinom orde 2

no	x	y	$x^2$	$x^3$	$x^4$	$xy$	$x^2y$
1	12	56	144	1728	20736	672	9064
2	14	62	196	2744	48416	868	12152
3	13	60	169	2197	28561	780	10140
4	12	61	144	1728	20736	732	8784
5	15	65	225	3375	50625	975	14625
6	13	66	169	2197	28561	858	11154
7	14	60	196	2744	38416	840	11760
8	15	63	225	3375	50625	945	14175
9	13	65	169	2197	28561	845	10985
10	14	62	196	2744	38416	868	12152
$\Sigma$	135	620	1833	25029	343653	8383	113991

Perhitungan Tabel 3.14 akan dimasukkan ke persamaan hubungan polinom.

$$\begin{cases} (10)a_0 + (135)a_1 + (1833)a_2 &= 620 \\ (135)a_0 + (1833)a_1 + (25049)a_2 &= 8383 \\ (1833)a_0 + (25049)a_1 + (343653)a_2 &= 113991 \end{cases}$$

Menghitung koefisien a0,a1 dan a2 dapat menggunakan operasi matriks perkalian invers.

$$\begin{bmatrix} 10 & 135 & 1833 \\ 135 & 1833 & 25049 \\ 1833 & 25049 & 343653 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 620 \\ 8383 \\ 113991 \end{bmatrix}$$

Berdasarkan perhitungan matriks, maka didapatkan  $a_0 = -67.96428571$  ,  $a_1 = 18.11309524$  dan  $a_2 = -0.625$ . Hasil perhitungan tiap koefisien dapat dimasukkan pada persamaan 2.11 sehingga membentuk persamaan untuk menghitung volume penjualan (Y) menggunakan biaya promosi (X).

$$Y = -67.96 + 18.11(BiayaPromosi) + (-0.63)(BiayaPromosi)^2$$

Menggunakan persamaan yang sudah diperoleh, maka prediksi volume penjualan (Y) menggunakan *polynomial regression* dapat dihitung menggunakan biaya promosi. Berikut adalah perbandingan antara volume penjualan asli (Y) dan volume penjualan prediksi *polynomial regression* ( $\hat{Y}$ ) pada Tabel 3.15.

Tabel 3.15: Perbandingan prediksi volume penjualan (Y) polynomial regression

Biaya Promosi (X)	Volume Penjualan (Y)	Prediksi Volume Penjualan ( $\hat{Y}$ )
12	56	59.39
14	62	63.11
13	60	61.88
12	61	59.39
15	65	63.10
13	66	61.88
14	60	63.11
15	63	63.10
13	65	61.88
14	62	63.11

Library *Scikit Learn* pada *Python* menyediakan fungsi untuk melakukan prediksi menggunakan *Polynomial Regression*. Kelas yang digunakan sama seperti *Linear Regression*, tetapi ada modifikasi yaitu pengubahan variabel *prediktor* untuk diubah sesuai dengan bentuk matriks koefisien untuk mendapatkan bentuk polinom dengan order sesuai *input*. Gambar 3.8 adalah contoh potongan kode untuk melakukan prediksi menggunakan *polynomial regression*.

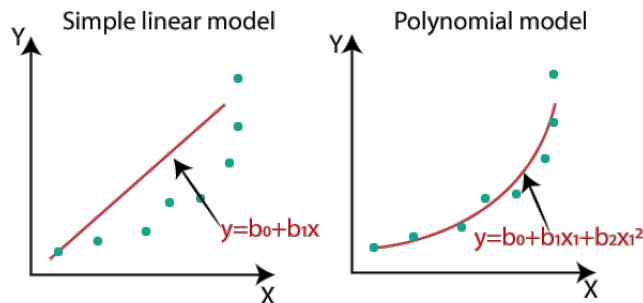
```

1 # transforming the data to include another axis
2 import numpy as np
3
4 # mengambil prediktor dan respons
5 x = dataset[['biayapromosi']]
6 y = dataset[['volumepenjualan']]
7
8 x = np.array(x)
9 y = np.array(y)
10
11 # mengubah sesuai order yang dipilih
12 polynomial_features= PolynomialFeatures(degree=2)
13 x_poly = polynomial_features.fit_transform(x)
14
15 # melakukan training dan prediksi data
16 model = LinearRegression()
17 model.fit(x_poly, y)
18 y_poly_pred = model.predict(x_poly)

```

Gambar 3.8: Kode contoh *polynomial regression*

Berdasarkan analisis mengenai penggunaan *Linear Regression* dan *Polynomial Regression*, Terdapat beberapa perbedaan yaitu kurva yang dihasilkan. Berikut adalah 2 gambar visualisasi kurva yang dihasilkan *Linear* dan *Polynomial Regression*.



Gambar 3.9: Perbandingan kurva linear dan polinom

Uraian Gambar 3.9 di atas menjelaskan perbedaan kurva yang dihasilkan dari kedua algoritma. Masing-masing memiliki karakteristik yang dapat dicocokan berdasarkan sifat data yang digunakan. Selain itu, berikut adalah perbedaan yang dapat digunakan untuk memahami kedua algoritma *Regression* yaitu :

- *Linear Regression* digunakan untuk korelasi data yang memiliki hubungan linear
- *Linear Regression* dengan menggunakan 2 atau lebih prediktor dapat disebut dengan *Multiple Linear Regression*
- *Polynomial Regression* digunakan untuk korelasi data yang memiliki hubungan non linear seperti eksponensial
- Pemilihan *order* pada *Polynomial Regression* dapat mempengaruhi seberapa *fit* (cocok) *model* prediksi yang dibuat berdasarkan data yang digunakan
- Pemilihan angka *order* yang besar pada *Polynomial Regression* dapat menyebabkan model prediksi yang dibuat menyebabkan *overfit*. *Overfit* adalah kondisi dimana model yang dibuat memiliki akurasi yang sangat baik terhadap data *test* tetapi akan membuat prediksi terhadap data yang tidak terduga menjadi buruk.

### 3.9 Analisis Penerapan Evaluasi Regresi

Hasil prediksi nilai yang diprediksi menggunakan *Linear Regression* dan *Polynomial Regression* dapat diukur seberapa akurat menggunakan *Mean Squared Error (MSE)* dan *Coefficient Determination (R<sup>2</sup>)*. Berdasarkan hasil prediksi yang dijelaskan pada bagian prediksi *Linear Regression* sebelumnya, berikut adalah perhitungan mengukur akurasi prediksi menggunakan persamaan *Coefficient of Determination* yang akan ditunjukkan pada tabel di bawah.

Tabel 3.16: Tabel perhitungan R<sup>2</sup> berdasarkan contoh prediksi Linear Regression

Volume Penjualan (Y)	Prediksi Volume Penjualan ( $\hat{Y}$ )	$(Y - \hat{Y})^2$	$(Y - \bar{Y})^2$
56	60.14	17.14	37.21
62	62.62	0.38	0.01
60	61.38	1.90	4.41
61	60.14	0.73	1.21
65	63.86	1.29	8.41
66	61.38	21.34	15.21
60	62.61	6.81	4.41
63	63.86	0.73	0.81
65	61.38	13.10	8.41
63	62.61	0.15	0.81
•			
$\sum \text{Sum}$		63.62	80.9
Mean (Y)			62.1

Berdasarkan uraian tabel 3.16 di atas, nilai *Coefficient Determination* dari prediksi Volume Penjualan menggunakan *Linear Regression* dapat dihitung.

$$R^2 = 1 - \frac{\sum (y - \hat{y})^2}{(y - \bar{y})^2} \quad (3.6)$$

$$R^2 = 1 - \frac{63.62}{90.9} = 0.21 \quad (3.7)$$

Berdasarkan hasil persamaan (3.7) di atas, evaluasi prediksi Volume Penjualan menggunakan *Linear Regression* berdasarkan R<sup>2</sup> adalah 0.21. Nilai R<sup>2</sup> adalah nilai yang berada diantara dari 0 sampai 1.0. Semakin tinggi nilainya semakin akurat prediksinya. Nilai R<sup>2</sup> memberikan evaluasi adanya korelasi antara prediktor dan *response* tetapi tidak terlalu kuat. Umumnya nilai prediksi yang bagus dan akurat adalah di atas 0.7.

*Library Scikit Learn* pada *Python* menyediakan perhitungan *Coefficient Determination* pada *Package metrics* yaitu *method r2\_score(actual , pred)* yang menerima nilai asli dan nilai prediksi untuk dihitung nilai akurasi. Berikut adalah potongan kode *r2\_score* yaitu :

```

1 #asumsikan sudah dilakukan prediksi
2 # variabel nilai resone asli = y
3 # variabel nilai response prediksi model = ypred
4 from sklearn.metrics import r2_score
5
6 nilaiAkurasi = r2_score(y, ypred)

```

Gambar 3.10: Kode contoh *evaluasi regresi*

## 3.10 Analisis Penerapan K-NN

Berdasarkan penjelasan pada Bab 2 mengenai klasifikasi, *K-NN* adalah algoritma *classification* untuk menentukan label sebuah data. Akan diberikan contoh perhitungan menggunakan *k-Nearest Neighbor* untuk memprediksi nilai kategori pada data baru. Berikut adalah contoh *data set* yang digunakan sebagai *training set* untuk membuat model *k-Nearest Neighbor*.

Tabel 3.17: Tabel dataset k-nearest neighbor

x	y	kategori
7	6	Bad
6	6	Bad
6	5	Bad
1	3	Good
2	4	Good
2	2	Good

Tabel 3.17 berisi 2 variabel prediktor / independen yaitu x dan y. Kategori merupakan variabel respon / dependen. Diberikan sebuah data objek baru dengan x = 3 dan y = 5. Algoritma *k-Nearest Neighbors* memilih k tetangga terdekat. Nilai k yang ditentukan adalah 3. Menggunakan *euclidean distance*, berikut adalah perhitungan jarak setiap data pada *data set* dengan data baru.

Tabel 3.18: Tabel perhitungan euclidean distance dengan data baru

x	y	kategori	euclidean distance dengan data baru	perhitungan
7	6	Bad	4.12	$\sqrt{(7-3)^2 + (6-5)^2}$
6	6	Bad	3.16	$\sqrt{(6-3)^2 + (6-5)^2}$
6	5	Bad	3	$\sqrt{(6-3)^2 + (5-5)^2}$
1	3	Good	2.82	$\sqrt{(1-3)^2 + (3-5)^2}$
2	4	Good	1.41	$\sqrt{(2-3)^2 + (3-5)^2}$
2	2	Good	3.16	$\sqrt{(2-3)^2 + (2-5)^2}$

Setelah menghitung setiap jarak dengan data baru, maka *k-Nearest Neighbors* akan memilih tetangga terdekat sebanyak k. Karena nilai k adalah 3, maka akan dipilih kategori berdasarkan

tiga tetangga dengan selisih jarak *euclidean distance* minimum. Berikut adalah 3 tetangga terdekat dengan data baru pada Tabel 3.19.

Tabel 3.19: Tabel 3 tetangga terdekat berdasarkan perhitungan euclidean distance

x	y	kategori	Jarak dengan data baru
2	4	Good	1.41
1	3	Good	2.82
6	5	Bad	3

Tabel 3.19 menunjukkan bahwa terdapat 3 tetangga terdekat. Algoritma *k-Nearest Neighbors* akan menentukan kategori dari data baru berdasarkan jarak terdekat dan paling banyak. Pada 3 jarak terdekat, terdapat 2 kategori "Good" dan 1 kategori "Bad". Karena jumlah "Good" lebih banyak daripada "Bad", maka data baru akan memiliki kategori "Good". Melakukan evaluasi pada hasil prediksi menggunakan *classification* menghitung *accuracy*. *Accuracy* dapat diperoleh dengan menghitung jumlah perbandingan prediksi yang benar dari semua data yang diprediksi.

*Library Scikit Learn* pada *Python* menyediakan fungsi untuk melakukan klasifikasi menggunakan K-NN yaitu *package neighbors* kelas *KNeighborsClassifier*. Prediksi label kelas menggunakan *library* menerima sebuah *input* yaitu *n\_neighbors* yaitu jumlah tetangga terdekat yang menjadi penentu jumlah label terdekat. Berikut potongan kode *K-NN*.

```

1 from sklearn.neighbors import KNeighborsClassifier
2
3 #Buat model dgn jumlah neighbor = 2
4 neighbor = 2
5 kNN_model_iris = KNeighborsClassifier(n_neighbors=neighbor)
6
7 #Train the model using the training sets
8 kNN_model_iris.fit(x,y)
9
10 #Predict the response for test dataset
11 y_pred = kNN_model_iris.predict(x)

```

Gambar 3.11: Kode contoh *K-NN*

### 3.11 Analisis Penerapan Evaluasi Klasifikasi

Evaluasi klasifikasi yaitu adalah cara melakukan evaluasi apakah hasil prediksi yang dihasilkan akurat. Berdasarkan Landasan Teori tentang evaluasi Klasifikasi, berikut akan diberikan contoh dalam melakukan evaluasi klasifikasi. Diberikan *dataset* nilai label asli dan hasil prediksi.

Tabel 3.20: Tabel dataset evaluasi klasifikasi

Label Asli ( $y$ )	Label Prediksi ( $\hat{y}$ )
GOOD	BAD
GOOD	GOOD
BAD	BAD
BAD	BAD

Tabel 3.20 di atas merupakan hasil perbandingan nilai label asli dan hasil label prediksi menggunakan klasifikasi. Terdapat 1 baris data yang tidak konsisten. Label asli dan label prediksinya tidak sesuai yang berarti prediksinya salah. Dari 4 observasi, terdapat 3 observasi yang prediksinya sesuai sehingga nilai evaluasinya adalah 0.75. Gambar 3.12 adalah contoh potongan kode untuk menghitung evaluasi klasifikasi.

```

1| from sklearn.neighbors import KNeighborsClassifier
2|
3| #Buat model dgn jumlah neighbor = 5
4| neighbor = 5
5| kNN_model_iris = KNeighborsClassifier(n_neighbors=neighbor)
6|
7| #Train the model using the training sets
8| kNN_model_iris.fit(X_train, y_train)
9|
10| #Predict the response for test dataset
11| y_pred = kNN_model_iris.predict(X_test)
12|
13| #Import scikit-learn metrics module for accuracy calculation
14| from sklearn import metrics
15| # Model Accuracy, how often is the classifier correct?
16| print("Accuracy_klasifikasi_Iris_dgn_neighbor_=_" , neighbor, ":",metrics.accuracy_score(y_test, y_pred))

```

Gambar 3.12: Kode contoh evaluasi klasifikasi

## 3.12 Analisis Penerapan K-Means

Berdasarkan Landasan Teori Bab 2 tentang algoritma *clustering*, *K-Means* adalah cara untuk mengelompokkan data. Diberikan sebuah contoh perhitungan *k-Means* untuk menentukan kelompok data. Berikut adalah contoh *data set* nilai yang digunakan.

Tabel 3.21: Tabel dataset k-Means

Nama	UTS	ART	UAS
Jonathan	75	40	95
James	90	95	100
Pedro	55	90	75
Luna	85	65	85
Harry	85	80	80
Chloe	55	50	51

Tabel 3.21 memiliki informasi / atribut berupa nilai UTS,ART dan UAS. K atau jumlah *cluster* yang ditentukan adalah K=2. Titik awal *centroid* yaitu c1 dan c2. Sesuai dengan cara kerja *K-Means*, titik *Centroid* diinisialisasi secara *random* yaitu :

- c1 : UTS(80) , ART(80),UAS(80)
- c2 : UAS(50) , ART(50), UAS(50)

Berikut adalah perhitungan tiap data objek dengan menggunakan *euclidean distance* pada persamaan (2.22).

Tabel 3.22: Tabel perhitungan k-Means iterasi ke-1

Nama	UTS	ART	UAS	Dist(data(i),c1)	Dist(data(i),c2)	Kelompok
Jonathan	75	40	95	43.01	53.44	c1
James	90	95	100	26.92	78.26	c1
Pedro	55	90	75	27.38	47.43	c1
Luna	85	65	85	16.58	51.72	c1
Harry	85	80	80	5	55	c1
Chloe	55	50	51	48.64	5.09	c2

Tabel 3.22 menunjukkan perhitungan iterasi pertama untuk mendapatkan kelompok tiap data objek. Titik *centroid* akan diubah dengan menghitung rata-rata atribut tiap anggota kelompok yang baru ditentukan pada Tabel 3.23.

Tabel 3.23: Tabel perubahan centroid iterasi 2

Nama	UTS	ART	UAS
c1	$\frac{(75+90+55+85+85)}{5} = 78$	$\frac{(40+95+90+65+80)}{5} = 74$	$\frac{(95+100+75+85+80)}{5} = 87$
c2	55	50	51

Setelah titik *centroid* berubah, maka perhitungan tiap data objek dengan *centroid* yang baru akan dilakukan.

Tabel 3.24: Tabel perhitungan k-Means iterasi ke-2

Nama	UTS	ART	UAS	dist(c1)	dist(c2)	kelompok
Jonathan	75	40	95	35.05	49.35	c1
James	90	95	100	27.45	75.17	c1
Pedro	55	90	75	30.47	46.64	c1
Luna	85	65	85	11.57	51.39	c1
Harry	85	80	80	11.57	51.39	c1
Chloe	55	50	51	49	0	c2

Karena pada Tabel 3.24 tiap data objek tidak berpindah ke *cluster* lain, sehingga *centroid* tidak perlu hitung kembali karena sudah konvergen.

*Library Scikit Learn* pada *Python* menyediakan fungsi untuk melakukan *clustering K-Means* menggunakan *package cluster* yaitu kelas *KMeans*. Kelas *KMeans* dapat memanggil fungsi *fit* dengan parameter *n\_cluster* yaitu jumlah kelompok yang ingin dibuat. Gambar 3.13 adalah potongan kode untuk melakukan *clustering* dengan *K-Means*.

```

1 #Clustering dgn k-Means
2 from sklearn.cluster import KMeans
3
4 #Lakukan clustering (fit) dgn jumlah cluster = 3
5 kmeans_model = KMeans(n_clusters=3, random_state=0).fit(X)
6
7 #Print label cluster pada tiap rekord/objek
8 kmeans_model.labels_
9
10 #Print centroid (cluster centers)
11 kmeans_model.cluster_centers_

```

Gambar 3.13: Kode contoh *clustering K-Means*

### 3.13 Analisis Penerapan Agglomerative

Berdasarkan penjelasan Landasan Teori Bab 2 tentang *Agglomerative Clustering*, diberikan contoh perhitungan cara memprediksi sebuah nilai dengan *Agglomerative Clustering*. Berikut adalah sebuah *dataset* yang digunakan.

Tabel 3.25: Tabel dataset perhitungan agglomerative

Nama	UTS	ART	UAS
Jonathan	74	40	95
James	90	95	100
Pedro	55	90	75
Luna	85	65	85

Tabel 3.25 berisi komponen nilai tiap siswa yaitu UTS,ART dan UAS. Berikut adalah perhitungan jarak tiap data objek menggunakan *euclidean distance*.

Tabel 3.26: Tabel perhitungan euclidean distance iterasi pertama

	Jonathan	James	Pedro	Luna
Jonathan	0			
James	$\sqrt{(75 - 90)^2 + (40 - 95)^2 + (95 - 100)^2} = 57.22$	0		
Pedro	$\sqrt{(75 - 55)^2 + (40 - 90)^2 + (95 - 75)^2} = 57.44$	43.4	0	
Luna	28.72	33.91	40.31	0

Tabel 3.26 berisi perhitungan jarak tiap data objek. Karena jarak Jonathan dan Luna paling kecil, maka akan dikelompokkan menjadi satu *cluster* yang sama yaitu c1. Satu *cluster* yang terdiri dari beberapa titik dapat dihitung berdasarkan rata-rata tiap atribut sehingga isi *data set* berubah.

Tabel 3.27: Tabel perubahan nilai cluster iterasi 1

Euclidean Distance	Jonathan,Luna	James	Pedro
Jonathan,Luna	$75 + 85/2 = 80$	$40 + 65/2 = 52.5$	$95 + 85/2 = 85$
James	90	95	100
Pedro	55	80	75

Tabel 3.27 menunjukkan perubahan titik tengah *cluster* berasal dari rata-rata atribut Jonathan dan Luna. Iterasi selanjutnya akan menghitung jarak antara tiap titik untuk mencari jarak terpendek.

Tabel 3.28: Tabel perhitungan euclidean distance iterasi kedua

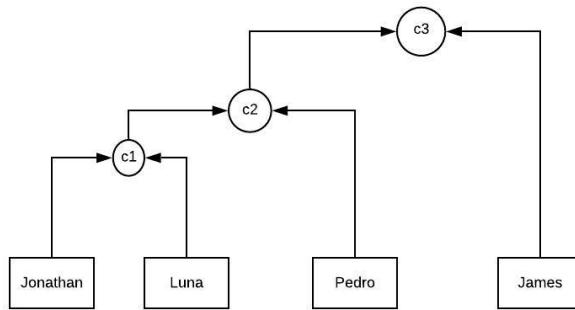
Euclidean Distance	Jonathan,Luna (c1)	James	Pedro
Jonathan,Luna (c1)	0		
James	44.79	0	
Pedro	40.07	45.55	0

Tabel 3.28 menunjukkan bahwa jarak antara c1 (Jonathan,Luna) dan Pedro memiliki jarak terpendek sehingga dikelompokkan menjadi satu *cluster*. Setelah Pedro digabungkan dengan c1 menjadi c2, maka titik tengah dari c2 (Jonathan,Luna,Pedro) berubah.

Tabel 3.29: Tabel perubahan nilai cluster iterasi 2

	UTS	ART	UAS
c2 (Jonathan,Luna, Pedro)	$(80 + 55)/2 = 67.5$	$(52.55 + 80)/2 = 66.25$	$(90 + 75)/2 = 82.5$
James	90	95	100

Tabel 3.29 menunjukkan bahwa tinggal terdapat 2 data objek yaitu *cluster* c2 dan James. Kedua data objek dapat digabung langsung tanpa harus menghitung jarak menjadi c3 (Jonathan, James, Pedro, Luna). Visualisasi *agglomerative clustering* dapat menggunakan dendogram.



Gambar 3.14: Dendogram example

*Library Scikit Learn* pada *Python* menyediakan fungsi untuk melakukan *clustering* menggunakan *Agglomerative*. Memanggil *package cluster* dan kelas *AgglomerativeClustering* untuk melakukan *clustering* membutuhkan 1 parameter yaitu *n\_clusters* yaitu jumlah *cluster* yang ingin dihasilkan dari *clustering*. Berikut adalah potongan kode yang digunakan yaitu :

```

1 #Clustering dgn Agglomeratives
2 from sklearn.cluster import AgglomerativeClustering
3 #Lakukan clustering (fit) dgn jumlah cluster = 3
4 agglo_model = AgglomerativeClustering(n_clusters=3).fit(X)
5 #Print label cluster pada tiap rekord/objek
6 agglo_model.labels_
  
```

Gambar 3.15: Kode contoh *agglomerative clustering*

### 3.14 Analisis Penerapan Evaluasi Clustering

Berdasarkan Landasan Teori pada Bab 2 mengenai evaluasi *clustering* terdapat salah satu teknik evaluasi *clustering* yaitu *silhouette plot*. Berikut diberikan *dataset* contoh.

Murid	Nilai	Kelompok
A	20	C1
B	30	C1
C	40	C1
D	70	C2
E	75	C2
F	90	C3
G	95	C3
H	100	C3

Tabel 3.30: Tabel dataset contoh yang akan di evaluasi

Tabel 3.30 adalah contoh hasil *clustering* untuk mengelompokkan siswa berdasarkan nilai. Pada tahap ini akan dihitung nilai *silhouette plot* dari murid "A" untuk mengukur kualitas *cluster*.

$$S(i) = \frac{(b(i) - a(i))}{\max(a(i), b(i))} \quad (3.8)$$

Persamaan 3.8 dapat digunakan untuk menghitung kualitas *cluster*. Untuk menghitung *silhouette plot* dari murid "A" / S(a), dibutuhkan nilai a dan b. Nilai a(i) adalah rata-rata jarak murid "A" ke

murid lain pada *cluster* yang sama. Nilai  $b(i)$  adalah rata-rata jarak murid "A" ke tetangga pada *cluster* lain.

$$\begin{aligned}
 b(i) &= (dist(A, B) + dist(A, C))/2 \\
 b(i) &= (dist(A, D) + dist(A, F))/2 \\
 S(A) &= \frac{b(i) - a(i)}{\max(b(i), a(i))} \\
 S(A) &= \frac{(60 - 15)}{\max(60, 15)} = 0.75
 \end{aligned} \tag{3.9}$$

Persamaan 3.9 adalah contoh hasil perhitungan untuk evaluasi *cluster* pada titik "A". Nilai *silhouette* adalah nilai yang memiliki rentang 0.0 sampai 1.0. Semakin tinggi nilai *silhouette* maka semakin baik sebuah *cluster*. Library *Scikit Learn* pada *Python* menyediakan fungsi untuk menghitung hasil *clustering* menggunakan *silhouette plot*. Gambar 3.16 adalah contoh kode untuk menghitung *silhouette plot*.

```

1 | from sklearn.metrics import silhouette_score
2 |
3 | clusterer = KMeans(n_clusters=n_clusters)
4 | preds = clusterer.fit_predict(df)
5 | score = silhouette_score(df, preds)

```

Gambar 3.16: Kode contoh evaluasi clustering

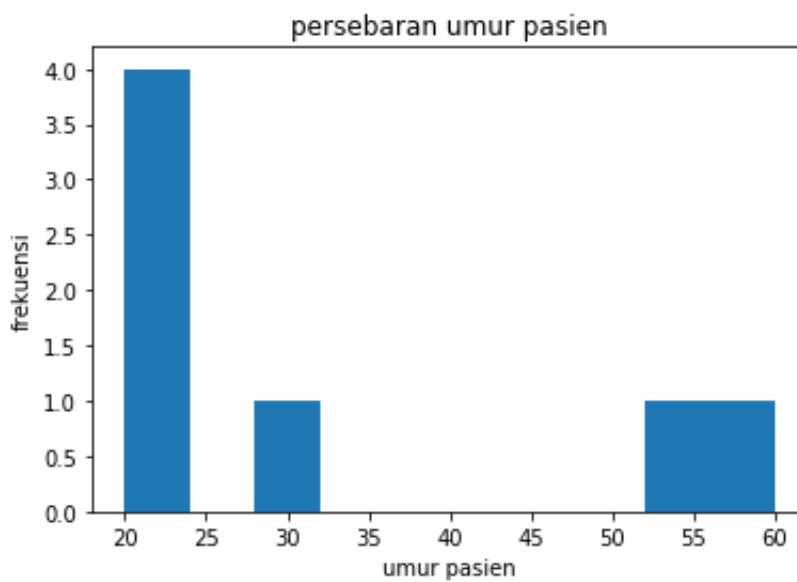
## 3.15 Analisis Visualisasi Data

Visualisasi Data berdasarkan Landasan Teori Bab 2 merupakan cara untuk merepresentasikan kumpulan data dalam bentuk gambar agar membantu memudahkan dalam melakukan analisis. Selain dari fungsi visualisasi, visualisasi data dapat dibagi berdasarkan dimensi data yang ditampilkan. *Histogram* dan *Boxplot* dapat digunakan untuk menampilkan data satu dimensi. Berikut diberikan contoh *dataset* yang akan digunakan untuk melakukan visualisasi.

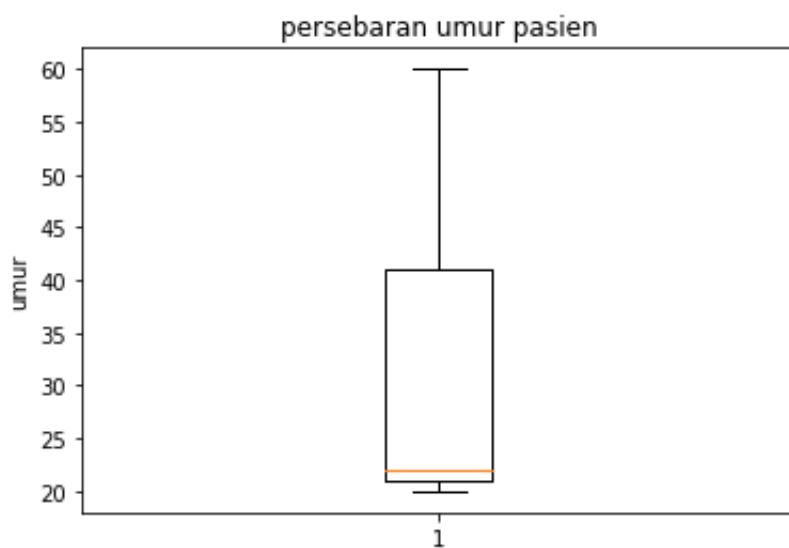
Tabel 3.31: Tabel dataset analisis visualisasi

Nama Pasien	Umur Pasien
Hashrul	21
Giovanny	22
Rasyif	21
Syafira	20
Nouval	52
Alit	30
Timoti	60

Tabel 3.31 di atas merupakan contoh kumpulan data pasien beserta umurnya. Dengan visualisasi data, analisis untuk melihat persebaran pasien berdasarkan umur akan lebih mudah dengan *Histogram* dan *Boxplot*. Berikut adalah hasil dari kedua visualisasi yang dibuat.



Gambar 3.17: Contoh Analisis Histogram



Gambar 3.18: Contoh Analisis Boxplot

Gambar 3.17 dan 3.18 merupakan hasil visualisasi dari *dataset* yang sudah dijelaskan sebelumnya. Berdasarkan *histogram*, dapat diidentifikasi kelompok umur mana yang paling banyak menjadi pasien. Berdasarkan *boxplot* yang dihasilkan, dapat dideteksi nilai maksimum, minimum , Q1,Q2 dan Q3 dari data pasien. *Library Matplotlib* pada *Python* menyediakan fungsi untuk melakukan visualisasi seperti *histogram* dan *boxplot*. Gambar 3.19 adalah contoh potongan kode untuk melakukan visualisasi.

```

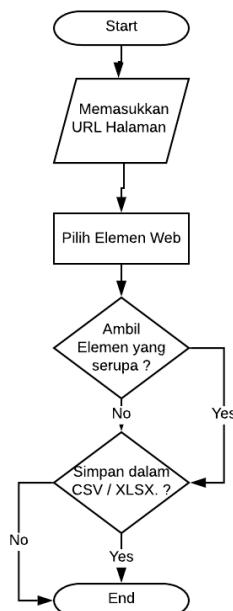
1 import matplotlib.pyplot as plt
2 import pandas as pd
3
4 dataset = pd.read_csv('umurdataset.csv')
5
6 #visualisasi menggunakan histogram
7 plt.title('persebaran_umur_pasien')
8 plt.xlabel('umur_pasien')
9 plt.ylabel('frekuensi')
10 plt.hist(dataset.umur)
11 plt.show()
12 plt.clf()

```

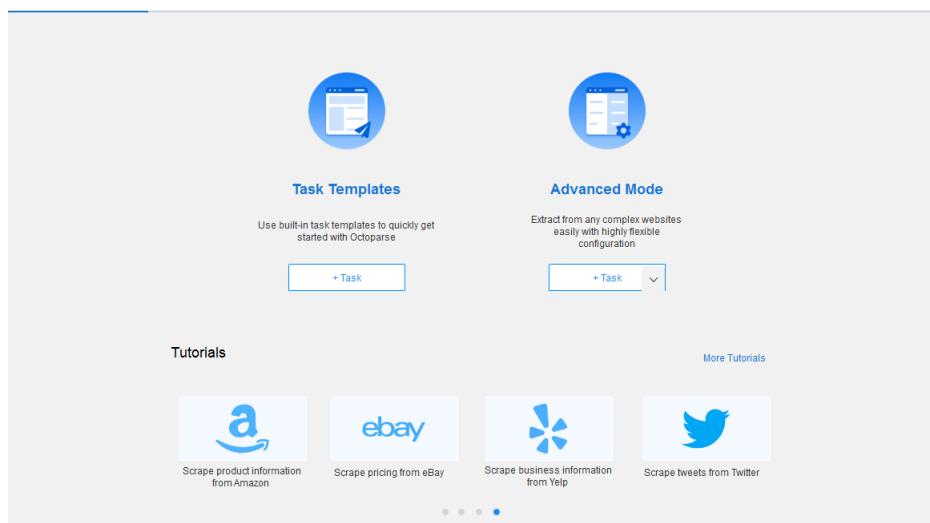
Gambar 3.19: Kode contoh visualisasi

### 3.16 Analisis Web Scrapping

Penggunaan *Web Scrapping* salah satunya adalah dengan menggunakan perangkat lunak yang memiliki fitur untuk melakukan *Web Scrapping*. *Octoparse* adalah perangkat lunak dengan antarmuka yang dapat mengambil sekumpulan elemen serupa pada sekumpulan *web page*. Perangkat lunak ini sudah memiliki tampilan antarmuka sehingga lebih mudah untuk digunakan berikut adalah tampilan perangkat lunak dari *octoparse*. Salah satu perangkat lunak yang akan digunakan pada penelitian ini adalah *Octoparse*. Berikut adalah ilustrasi langkah *Scraping* menggunakan *octoparse* dalam bentuk *flowchart* pada Gambar 3.20.

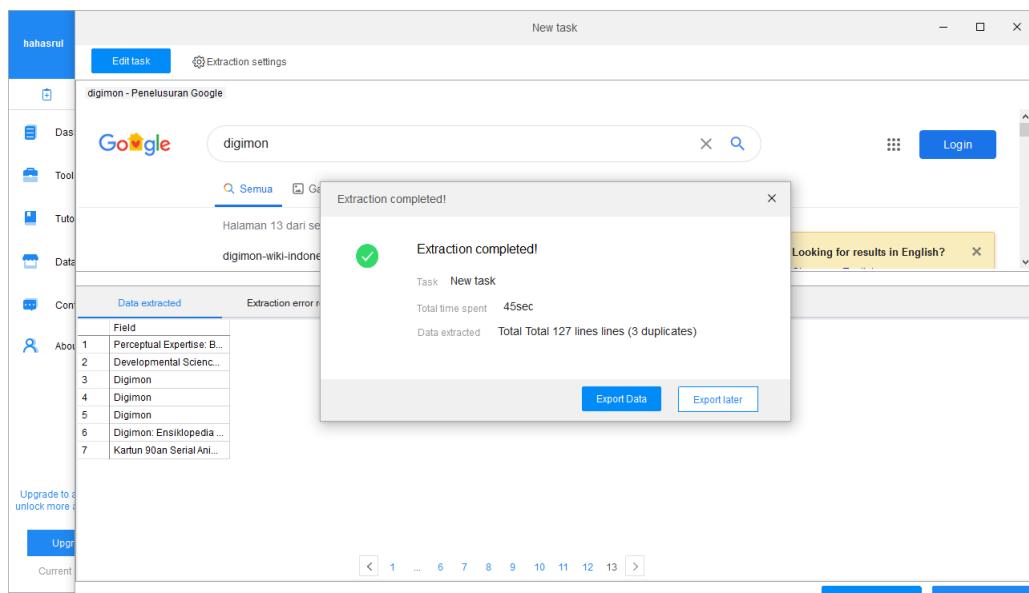


Gambar 3.20: Flowchart Octoparse



Gambar 3.21: Tampilan utama Octoparse

Gambar 3.21 adalah tampilan utama dari *Octoparse*. Diberikan contoh ilustrasi dalam melakukan *scrapping* hasil pencarian situs *google* dengan *keyword* *digimon* untuk mengambil judul halaman pada pencarian.



Gambar 3.22: Tampilan scrapping pada Octoparse

*Output* dari *Scrapping* sesuai Gambar 3.22 berupa data tabel yang dapat disimpan dalam beberapa bentuk seperti CSV, Excel dan JSON. Hasil *scrapping* serupa akan digunakan pada penelitian untuk melakukan *data collection* agar analisis tambahan dapat dilakukan.

## BAB 4

# IMPLEMENTASI

Bab ini akan membahas lingkungan pengembangan eksperimen yang sudah dilakukan. Pada bab ini akan dijelaskan proses hasil analisis data dan hasil prediksi beberapa fitur yang berhubungan dengan kesuksesan film. Selain itu, bab ini akan menjelaskan analisis pola-pola yang menarik dalam bentuk visualisasi data.

### 4.1 Lingkungan Perangkat Keras

Perangkat keras yang akan digunakan untuk melakukan eksperimen ini adalah *Laptop Lenovo Ideapad 330* dengan detail spesifikasi berupa :

- Prosesor : Intel(R) Core(TM) i5-8250U
- Memori : 4GB RAM.

Lingkungan perangkat keras dapat berpengaruh pada waktu pemrosesan dalam melakukan eksperimen.

### 4.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan untuk melakukan eksperimen memiliki spesifikasi berupa :

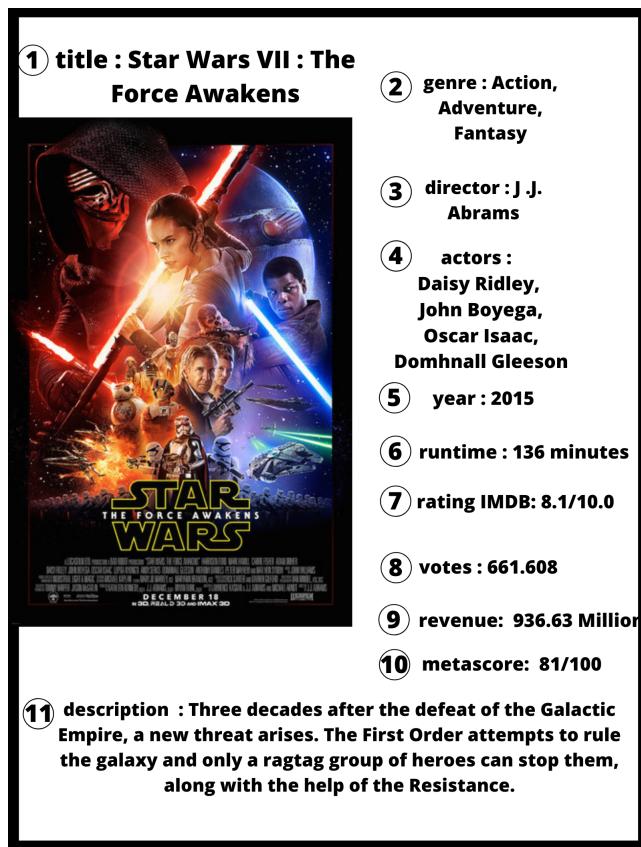
- Sistem Operasi : Windows 10 64-bit
- Bahasa Pemrograman : Python
- IDE : Spyder (Anaconda 3)

### 4.3 Deskripsi Dataset

Nama *dataset* yang digunakan adalah "*IMDB data from 2006 to 2016*". *Dataset* ini adalah file tabular dalam format CSV . *Dataset* ini berisi 1000 data film dari tahun 2006 sampai 2016. Deskripsi utama *dataset* ini adalah :

- Nama file : IMDB-Movie-Data.csv
- Sumber data : <https://www.kaggle.com/PromptCloudHQ/imdb-data>
- Size : 302KB
- Jumlah Baris: 1000 baris
- Jumlah Kolom: 12 kolom

Berikut adalah ilustrasi sebuah baris dari *dataset*.



Gambar 4.1: Ilustrasi sebuah baris film pada dataset

Berdasarkan deskripsi utama dan Gambar 4.1 di atas, sebuah baris pada *dataset* merepresentasikan sebuah film dengan berbagai informasi yang terkait seperti judul, aktor yang terlibat, skor *rating* dan lain-lain. Berikut adalah deskripsi dari masing masing kolom *dataset* pada Tabel 4.1.

Tabel 4.1: Deskripsi dataset

Nama	Deskripsi	Tipe
Title	Judul film	String
Genre	Jenis film. Sebuah film dapat memiliki lebih dari 1 genre. Genre ditulis dan dipisah menggunakan tanda koma	String
Description	Sinopsis film	String
Director	Nama sutradara	String
Actor	Nama pemain-pemain film. Sebuah film dapat memiliki lebih dari satu aktor. Aktor ditulis terpisah dengan tanda koma	String
Year	Tahun rilis film	Int
Runtime	Durasi film dalam satuan menit	Int
Rating	Skor Review dari situs IMDB	Float
Votes	Jumlah pengguna situs IMDB yang menyukai sebuah film	Int
Revenue	Pendapatan Kotor Film. Ditulis dalam satuan juta USD	Float
Metascore	Skor review dari situs Metacritic	Float

Untuk memahami bagaimana bentuk *dataset*, berikut adalah 5 baris awal dari *dataset* pada Tabel 4.2

Tabel 4.2: 5 Contoh Data pada dataset

Title	Genre	Description	Director	Actors	Year	Runtime	Rating	Votes	Revenue	Metascore
Guardians of the galaxy	Action, Adventure, Sci-fi	A group of inter...	James Gunn	Chriss Pratt, Zoe Saldana, Nootmi Rapace, Logan Marshall...	2014	121	8.1	757074	333.13	76
Promet-heus	Adventure, Mystery, Sci-fi	Following clues ...	Ridley Scott	Nootmi Rapace, Logan Marshall...	2012	124	7	485820	126.46	65
Split	Horror, Thriller	Three girls ...	M. Night Shyamalan	James McAvoy, Anya Taylor...	2016	117	7.3	157606	138.12	62
Sing	Animation, Comedy, Family	in a city of ...	Christophe Lourdelet	Matthew McConaughey, Reese Witherspoon	2016	108	7.2	60545	270.32	59
Suicide Squad	Action, Adventure, Fantasy	A secret government...	David Ayer	Will Smith, Jared Leto	2016	123	6.2	393727	325.02	40

## 4.4 Proses Analisis Data Utama

Pada subbab ini dijelaskan analisis awal yang dilakukan dengan menggunakan *dataset* yang sudah disediakan. Tahap-tahap yang dilakukan selama proses analisis data utama adalah :

- Melakukan *Data Cleaning* untuk mengatasi *missing value*

- Melakukan Analisis Visualisasi untuk menemukan pola menarik
- Melakukan *Data Selection* untuk memilih fitur yang berpengaruh untuk prediksi *revenue*
- Melakukan percobaan prediksi *revenue* berdasarkan fitur menarik menggunakan regresi

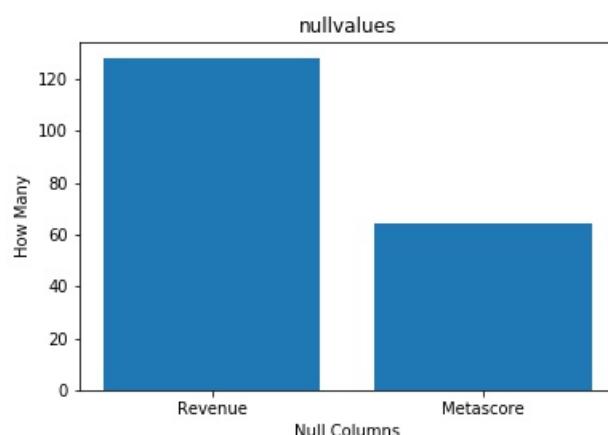
Penjelasan secara detail mengenai proses analisis data utama ada pada subbab selanjutnya. Berdasarkan hasil analisis data utama yang dilakukan, informasi yang didapat adalah :

- *Genre* film yang paling banyak dibuat berdasarkan jumlah film dan besar *revenue* yang diperoleh adalah *Action* dan *Adventure*
- Selera penonton (*votes*) dalam menilai film tidak sama selera para kritikus / ahli film (*review*)
- Sebuah film relatif lebih mudah mendapatkan nilai *review* yang bagus dari situs IMDB dibanding situs Metacritic
- Film bioskop umumnya memiliki durasi film dari 60 menit sampai 180 menit. Film yang paling banyak diproduksi adalah film 100 sampai 120 menit. Ada kecenderungan semakin lama durasi film maka semakin banyak peminat
- Investasi usaha film dapat menguntungkan karena akumulasi *revenue* dan jumlah film yang dibuat tiap tahunnya selalu meningkat
- Nilai *review* yang tinggi pada sebuah film tidak menjamin film tersebut dapat meraih keuntungan yang maksimal
- Penilaian film pada penonton (*votes*) jauh lebih berpengaruh dari pada penilaian pada situs *review* untuk menentukan kesuksesan film berdasarkan perbandingan korelasi.

Penjelasan secara detail mengenai analisis data utama akan dijelaskan selanjutnya.

#### 4.4.1 Data Cleaning

Pada proses *data cleaning* dilakukan deteksi baris yang mengandung *Null* dengan menggunakan fungsi *dropna()* dari library *pandas*. *Barchart 4.2* adalah visualisasi kolom dan perbandingan jumlah baris yang memiliki *missing value*.



Gambar 4.2: Barchart deteksi jumlah baris yang kolomnya terdapat null

Berdasarkan Gambar 4.2 di atas, ternyata terdapat baris yang kolom Metascore dan Revenue terdapat *Null*. Baris dengan kolom yang terdapat *Null* dibuang dan tidak diikutsertakan dalam analisis lebih lanjut karena akan menghasilkan *error*. Berdasarkan Gambar 4.2, terdapat 162 baris dengan *null* yang dihilangkan.

#### 4.4.2 Analisis Data Utama Menggunakan Visualisasi

Pada tahap ini dalam proses analisis data utama dilakukan analisis menggunakan visualisasi. Proses visualisasi *dataset* dilakukan untuk menemukan pola menarik.

##### Hubungan Genre Dengan Revenue

Sebuah film dapat memiliki 1 atau lebih *genre*. Fitur *genre* adalah salah satu fitur yang membutuhkan pemrosesan lebih lanjut. Gambar 4.1 menunjukkan bahwa film *Star Wars* memiliki lebih dari 1 *genre* yaitu *Action*, *Adventure* dan *Fantasy*. Untuk menghitung jumlah *genre* untuk setiap film, baris pada *dataset* yang dipisahkan dengan koma harus dipecah menjadi baris yang berbeda.

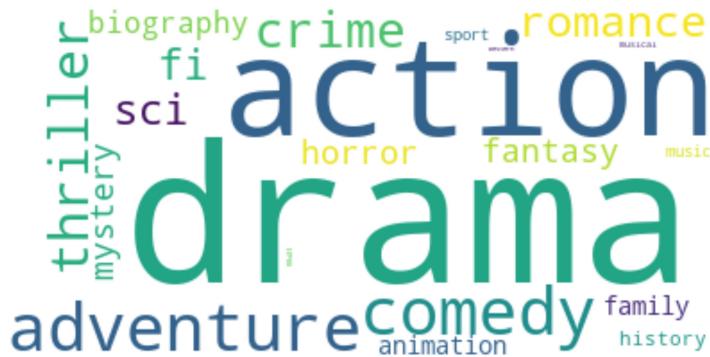
Tabel 4.3: Tabel sebelum *split*

title	genre
Star Wars VII : The Force Awakens	Action,Adventure,Fantasy

Tabel 4.4: Tabel setelah *split*

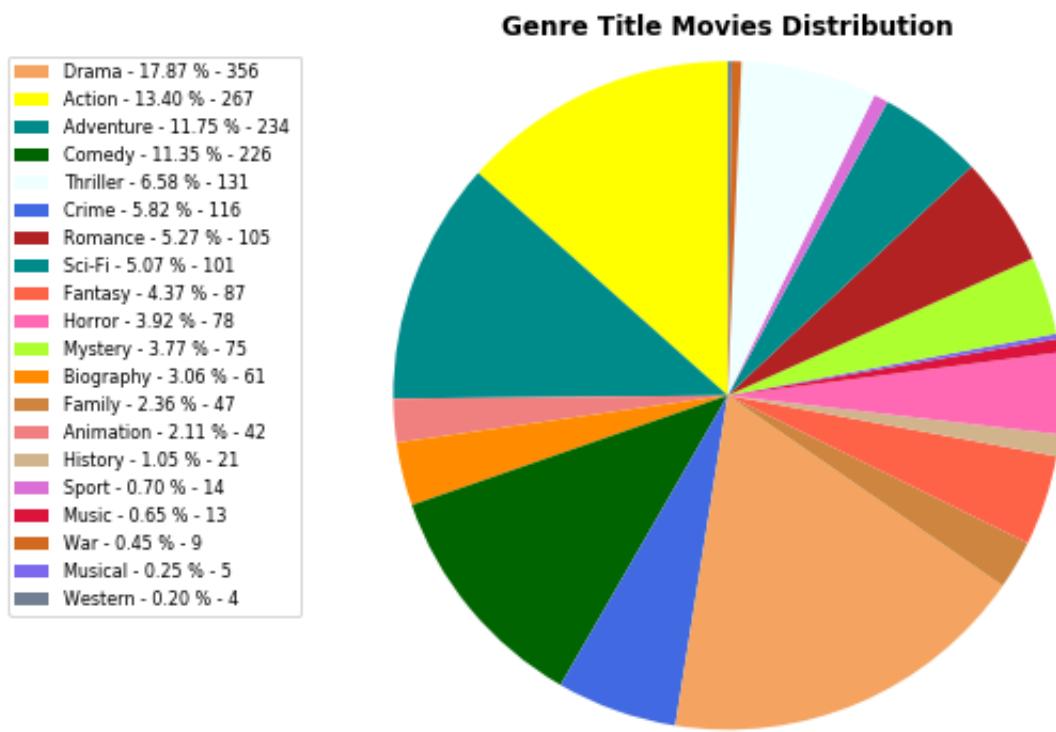
title	genre
Star Wars VII : The Force Awakens	Action
Star Wars VII : The Force Awakens	Adventure
Star Wars VII : The Force Awakens	Fantasy

Ilustrasi Tabel 4.3 dan Tabel 4.4 menunjukkan teknik untuk memecah *String* pada kolom *genre* yang tergabung dengan *delimiter* koma sehingga menjadi baris yang berbeda. Hasil *genre* yang terpisah sekarang dapat dihitung frekuensi tiap *genrenya*. Visualisasi frekuensi kemunculan tiap *genre* untuk semua film pada *dataset* dapat menggunakan *wordcloud*.



Gambar 4.3: Wordcloud kolom genre pada dataset

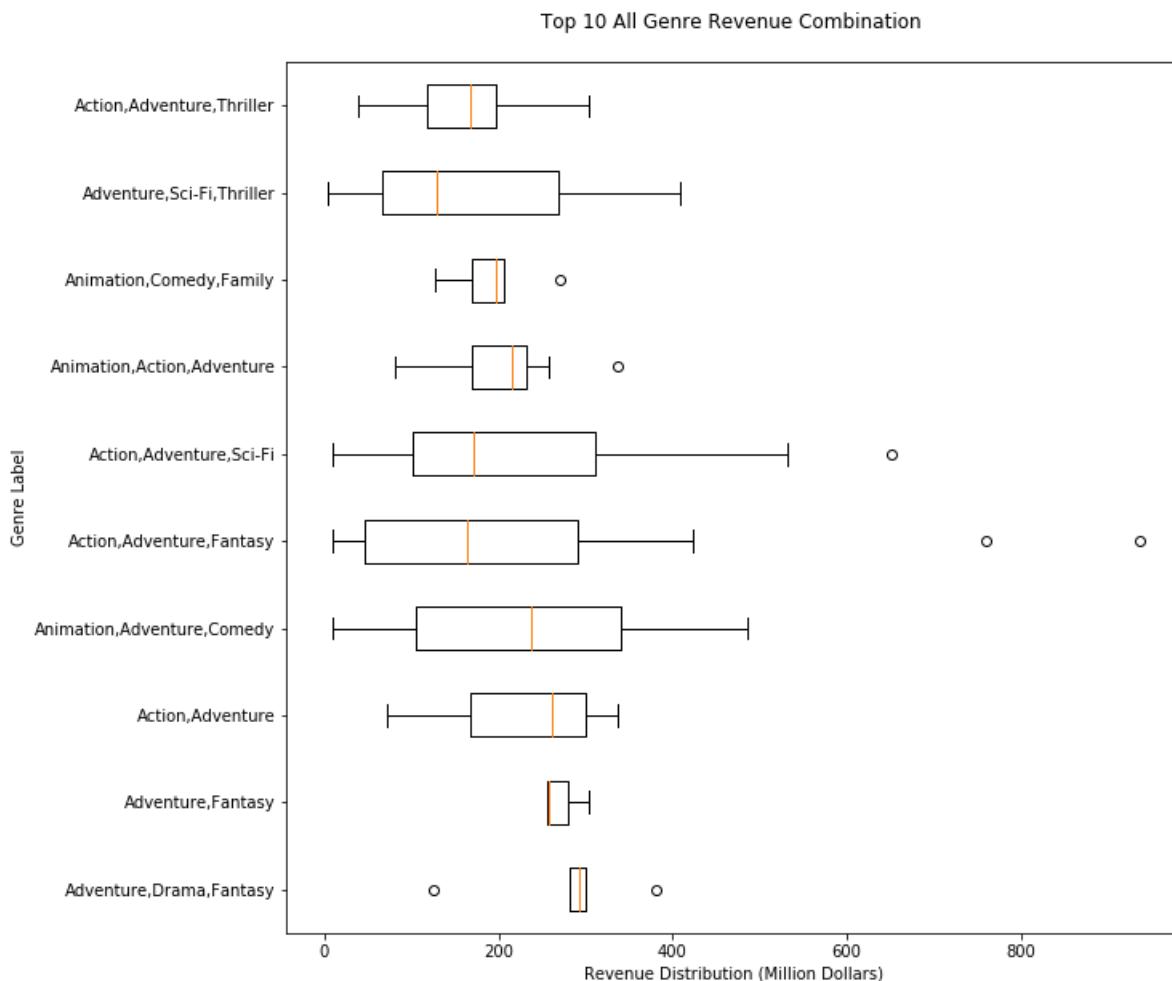
Gambar 4.3 merupakan hasil *wordcloud* dari kolom *genre*. *Wordcloud* adalah visualisasi kemunculan tiap kata yaitu *genre* pada semua *dataset*. Semakin besar huruf suatu kata berarti semakin besar frekuensi kemunculan kata tersebut pada *dataset*. Visualisasi *Piechart* adalah jumlah kemunculan *genre* tiap film pada *dataset*.



Gambar 4.4: Piechart persebaran film berdasarkan genre

Berdasarkan Gambar 4.4 di atas, jumlah film yang dibuat terbanyak salah satunya adalah *Drama*, *Action* dan *Adventure*. Distribusi film berdasarkan *genre* dapat dijadikan referensi kepada para pembuat film mengenai *genre* apa saja yang sering dibuat.

Berdasarkan *dataset* yang digunakan, sebuah film memiliki 1 atau lebih *genre*. Visualisasi analisis 10 kombinasi *genre* dengan *revenue* tertinggi dapat menggunakan *boxplot*. *Boxplot* tersebut dapat digunakan untuk membandingkan *revenue* tiap kombinasi *genre* pada *dataset*.

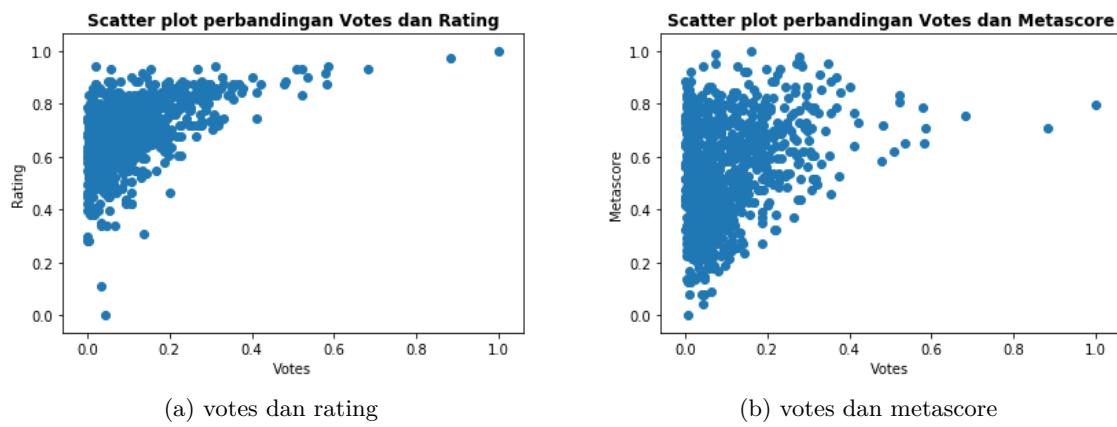


Gambar 4.5: Top 10 Kombinasi genre dengan pendapatan kotor (Revenue) Terbaik

Gambar 4.5 di atas menunjukkan 10 kombinasi *genre* yang menghasilkan *revenue* paling banyak. Kombinasi *genre* dengan *revenue* terbaik adalah kombinasi *Adventure, Drama, Fantasy*. Referensi ini dapat dijadikan *feedback* untuk pembuat film dalam menentukan *genre*. Film yang mengandung *genre action* dan *adventure* cenderung memperoleh kesuksesan karena dari jumlah yang dibuat dan rentang pendapatan relatif tinggi.

### Hubungan Votes dan Review

Sebuah film memiliki 2 metrik penilaian berdasarkan siapa yang menilainya yaitu *votes* (selera penonton) dan *review* (selera para kritikus). Pada tahap ini akan dianalisis hubungan *votes* dengan fitur *review* yaitu *metascore* dan *rating*.

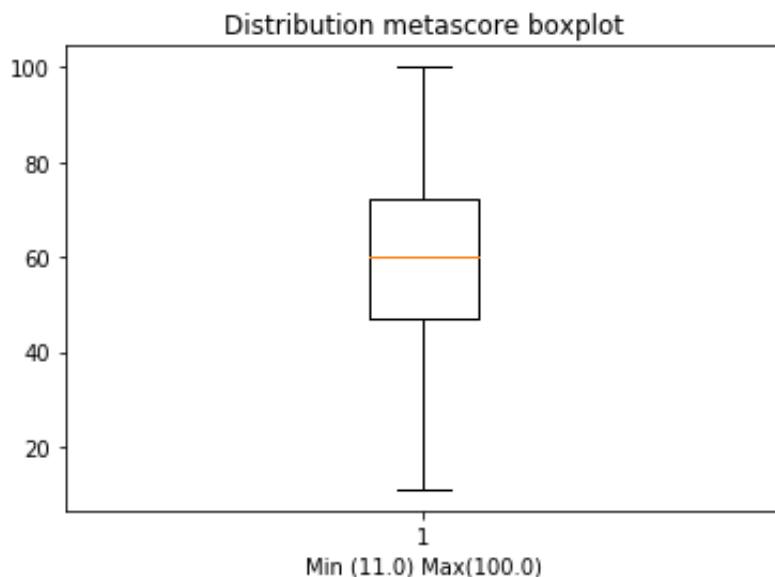


Gambar 4.6: Hubungan korelasi selera penonton dengan *review*

Gambar 4.6 adalah visualisasi korelasi antara *votes* dengan *review* menggunakan *scatter plot*. Berdasarkan visualisasi *scatter plot*, *votes* tidak menunjukkan korelasi dengan *rating* maupun *metascore*. Berdasarkan visualisasi yang diberikan, selera penonton pada sebuah film tidak sama dengan selera kritikus.

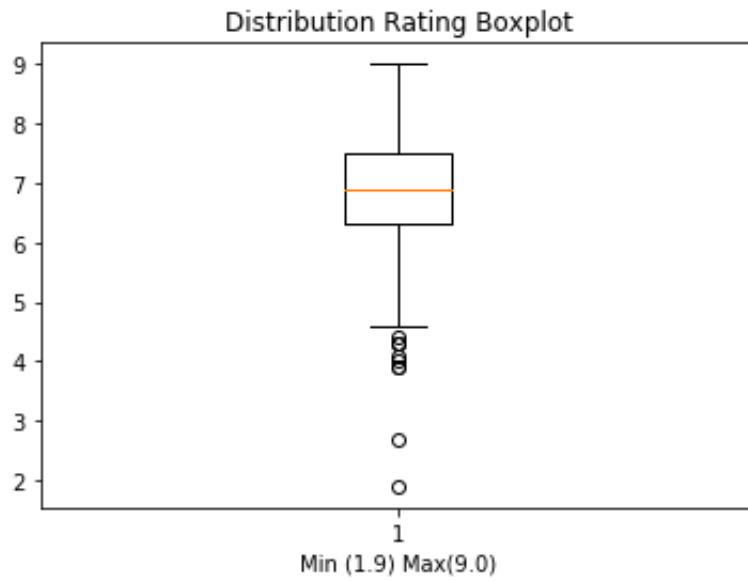
### Hubungan Metascore Dengan Rating

Pada tahap ini dilakukan analisis hubungan fitur *metascore* dan *rating* sebagai salah satu metrik penilaian kualitas sebuah film.



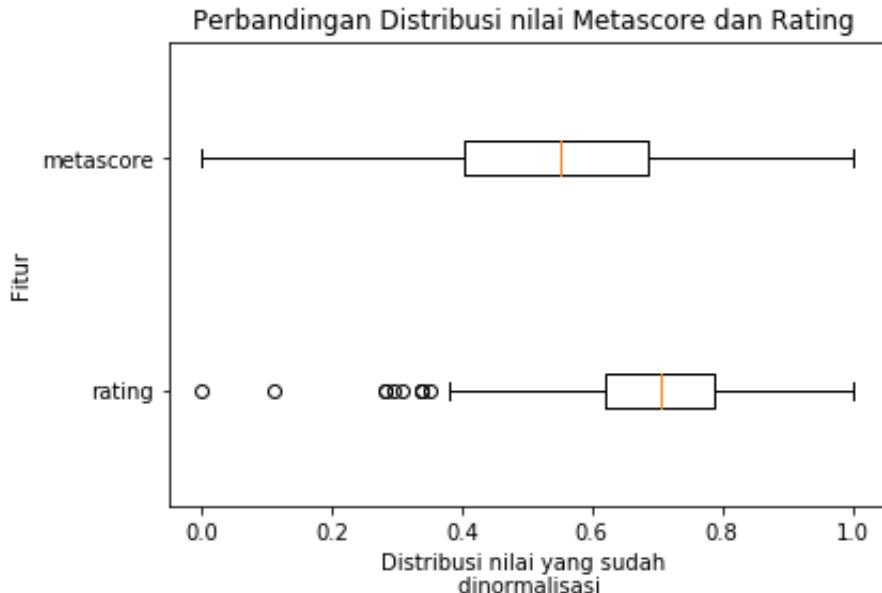
Gambar 4.7: Distribusi nilai kolom metascore

Gambar 4.7 di atas merupakan distribusi nilai dari *metascore* dengan menggunakan *boxplot*. Nilai minimum dari distribusi *metascore* adalah 11 sehingga terdapat film yang bisa mendapatkan nilai sangat kecil. Perbandingan nilai *metascore* akan dibandingkan dengan nilai *rating*.



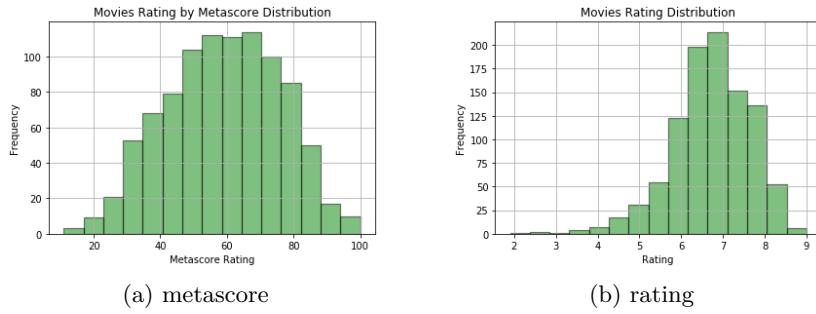
Gambar 4.8: Distribusi nilai kolom rating

Gambar 4.8 di atas merupakan distribusi nilai dari *rating* menggunakan *boxplot*. Nilai minimum adalah 1.9 dan maksimum dari distribusi *rating* adalah 9.0. Pada tahap ini dilakukan perbandingan *metascore* dan *rating* *boxplot* yang sudah dinormalisasi.



Gambar 4.9: Perbandingan Distribusi nilai kolom review

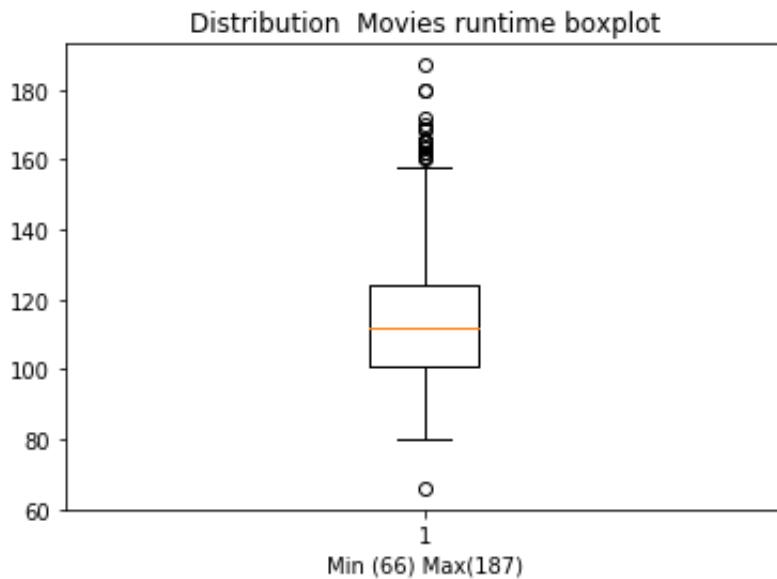
Gambar 4.9 adalah perbandingan rentang nilai *boxplot metascore* dan *rating* yang sudah dinormalisasi. Nilai Q2 pada *boxplot rating* dan *boxplot metascore* menunjukkan bahwa sebuah film dapat lebih mudah memperoleh nilai lebih baik dengan rating dibanding *metascore*.



Gambar 4.10: Histogram kolom metascore dan rating

Gambar 4.10 merupakan distribusi nilai *review* pada situs yang berbeda menggunakan *histogram*. Kolom *rating* merupakan nilai *review* dari sumber asal *dataset* yaitu situs IMDB<sup>1</sup>. Kolom *metascore* adalah nilai *review* yang diberikan oleh situs *review* film yang lain yaitu Metacritic<sup>2</sup>. Sebuah film cenderung mendapatkan *review* yang bagus di situs IMDB dibanding situs *metacritic*.

### Hubungan Runtime Dengan Votes

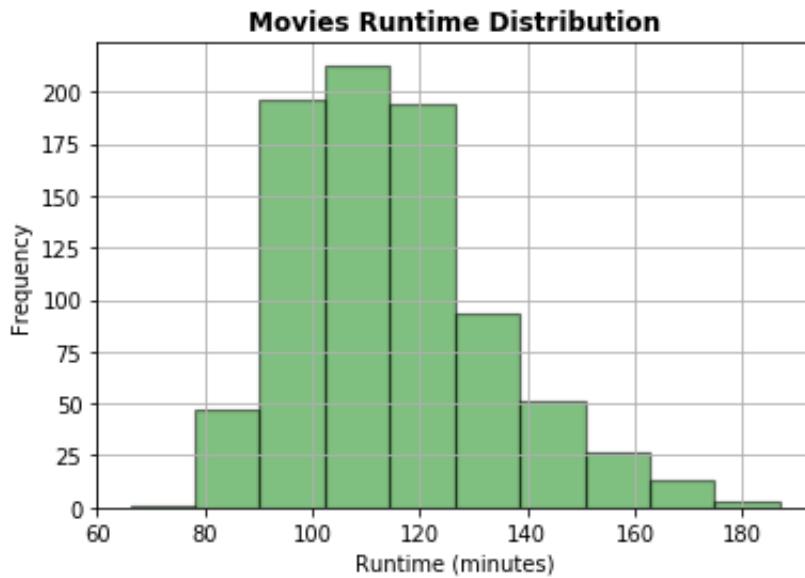


Gambar 4.11: Distribusi nilai kolom runtime

Gambar 4.11 di atas merupakan distribusi nilai dari *runtime* menggunakan *boxplot*. *Boxplot* ini dapat membantu pembuat film untuk melihat *range* untuk membuat film dari rentang minimum yaitu sekitar 60 menit hingga 180 menit. Pada tahap selanjutnya dilihat persebaran frekuensi *runtime* menggunakan *histogram*.

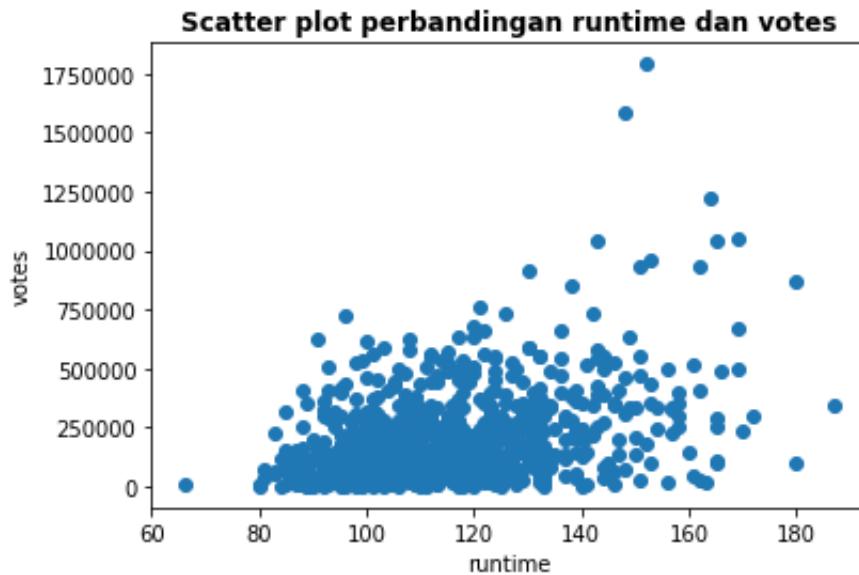
<sup>1</sup><https://www.imdb.com/>

<sup>2</sup><https://www.metacritic.com/>



Gambar 4.12: Distribusi kelas nilai runtime

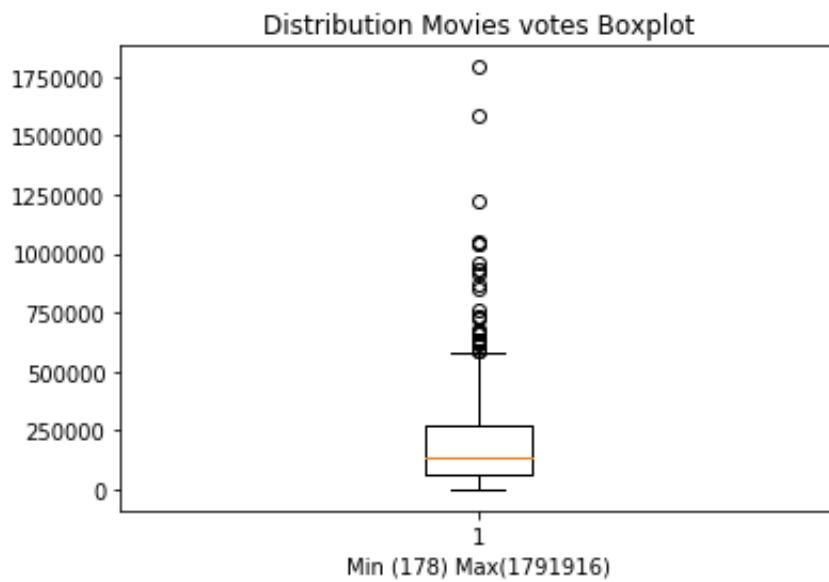
Berdasarkan Gambar 4.12, distribusi *runtime* memiliki distribusi normal. Mayoritas film dibuat dengan memiliki durasi 100-120 menit. Pada tahap ini ingin diketahui apakah durasi film yang dibuat sudah memenuhi minat penonton.



Gambar 4.13: Hubungan korelasi runtime dan votes

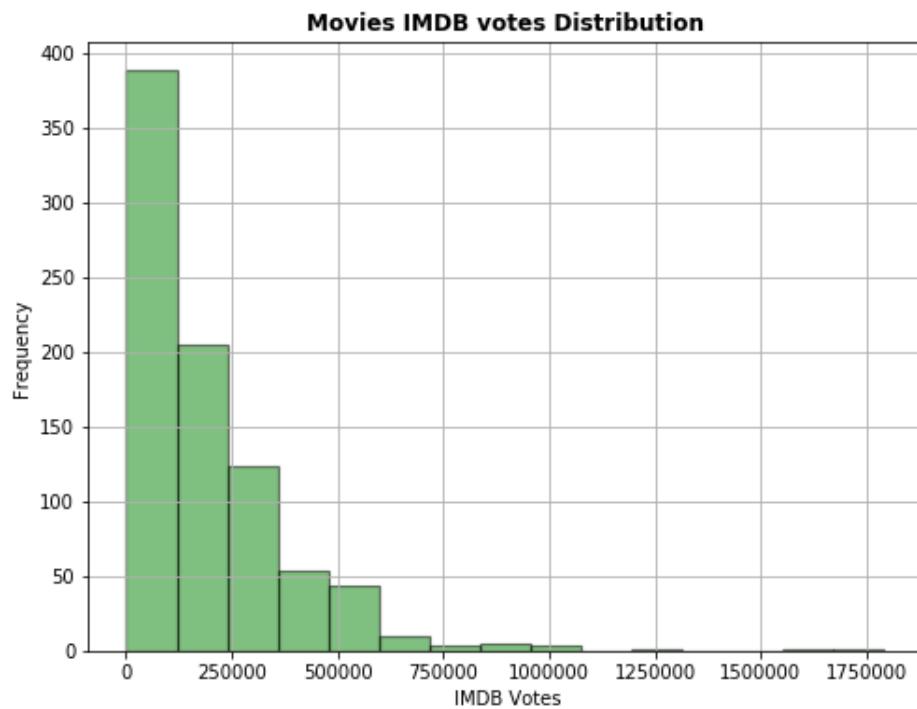
*Scatter plot* pada Gambar 4.13 adalah hubungan korelasi antara durasi film dan jumlah pendukung film pada situs IMDB. Berdasarkan visualisasi yang diberikan, sebuah film cenderung memiliki banyak pendukung jika durasi film semakin lama. Pembuat film dapat memenuhi minat penonton jika pihak pembuat film membuat film dengan durasi yang lebih lama.

### Analisis Votes Dengan Visualisasi



Gambar 4.14: Distribusi nilai kolom votes

Gambar 4.14 di atas merupakan distribusi nilai dari *votes* menggunakan *boxplot*. Terdapat sebuah film yang memiliki nilai *votes* sangat tinggi. Pada *boxplot* ini, dilakukan deteksi *outlier* dengan cara menghitung *interquartile range* ( $Q3 - Q1$ ). Nilai yang melebihi nilai  $Q3 + 1.5 * \text{interquartile range}$  merupakan *outlier*.



Gambar 4.15: Histogram distribusi kolom votes

Kolom *votes* merupakan jumlah pengguna situs IMDB yang menyukai sebuah film menggunakan *histogram*. Tiap bar merepresentasikan kelompok *range votes* yang diperoleh sebuah film. Gambar

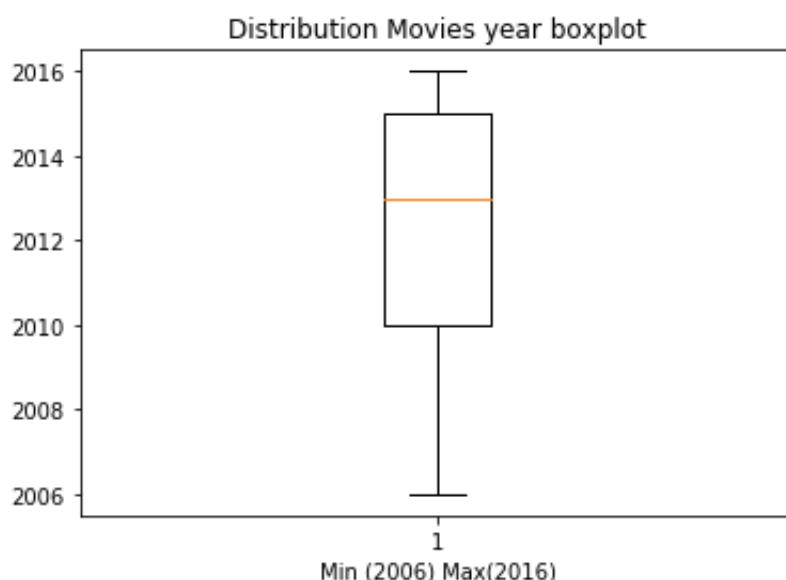
4.15 merupakan histogram persebaran *votes* film pada *dataset*. Gambar tersebut menunjukkan bahwa *film* umumnya mendapatkan sampai jumlah 250.000 *votes*. Film-film yang memiliki *votes* tinggi cenderung merupakan film-film *blockbuster*. Film *blockbuster* adalah sebutan untuk film yang sangat populer dan mendapatkan keuntungan setinggi-tingginya.

Tabel 4.5: Film *outlier* berdasarkan *votes*

Title	Rating	Runtime	Metascore	Revenue	Votes
The Dark Knight	9	152	82	533.32	1,791,916
Inception	8.8	148	74	292.57	1,583,625
The Dark Knight Rises	8.5	164	78	484.13	1,222,645
Interstellar	8.6	169	74	187.99	1,047,747
The Avengers	8.1	143	69	623.28	1,045,588
Django Unchained	8.4	165	81	162.8	1,039,115
Inglourious Basterds	8.3	153	69	120.52	959,065
The Departed	8.5	151	85	132.37	937,414
Avatar	7.8	162	83	760.51	935,408
The Prestige	8.5	130	66	53.08	913,152
The Wolf of Wall Street	8.2	180	75	116.87	865,134
Shutter Island	8.1	138	63	127.97	855,604
Guardians of The Galaxy	8.1	121	76	333.13	757,074
Iron Man	7.9	126	79	318.3	737,719
The Hunger Games	7.2	142	68	408	735,604
Up	8.3	96	88	292.98	722,203

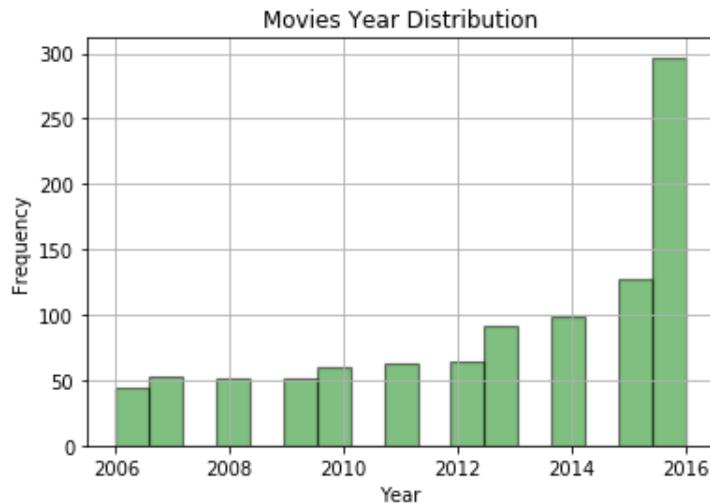
Tabel 4.5 di atas, menunjukkan film *outlier* dengan *votes* tertinggi. *Film* dengan *votes* tinggi cenderung memiliki keuntungan yang besar yaitu lebih dari 100 juta USD. Film *outlier* dengan *votes* terbanyak memiliki keuntungan hingga ratusan juta dollar. Berdasarkan tabel di atas pernyataan pada Gambar 4.10 juga terbukti. Nilai *review* yang lebih baik mudah diraih di situs IMDB dibanding situs Metacritic.

### Tren Revenue Dari Tahun Ke Tahun



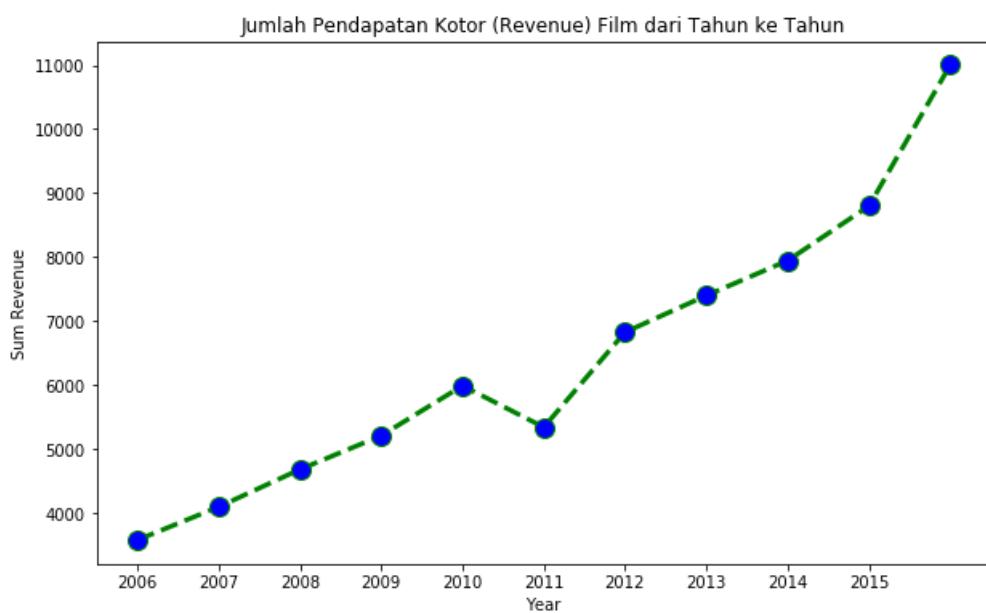
Gambar 4.16: Distribusi nilai kolom year

Gambar 4.16 merupakan distribusi nilai dari kolom year pada *dataset*. *Boxplot* ini juga dapat digunakan untuk menguji apakah benar *dataset* hanya film yang tahun rilisnya 2006 sampai 2016 sesuai judul dari *dataset* yang digunakan. Selain analisis distribusi dengan *boxplot*, *histogram* juga dapat digunakan untuk melihat distribusi berdasarkan interval nilai kelas.



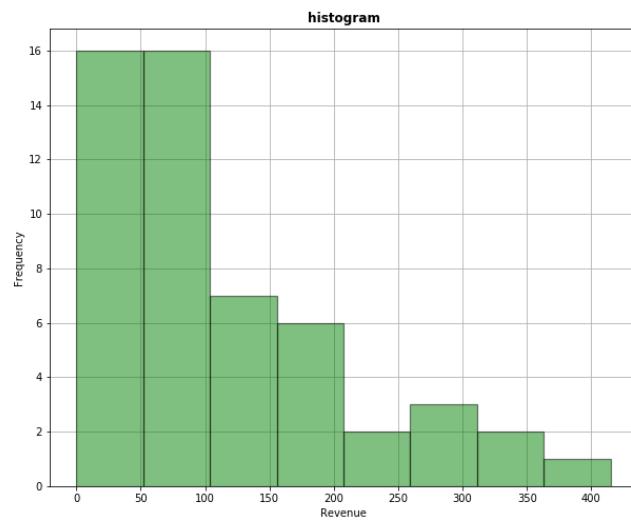
Gambar 4.17: Histogram distribusi kolom year

Gambar 4.17 merupakan Histogram kolom year. Hasil visualisasi menunjukkan bahwa jumlah film paling banyak dibuat berdasarkan *dataset* ini adalah film yang dibuat pada tahun 2016. Jumlah film yang dibuat dari tahun ke tahun juga meningkat berarti adanya peningkatan minat produksi film sebagai industri hiburan.

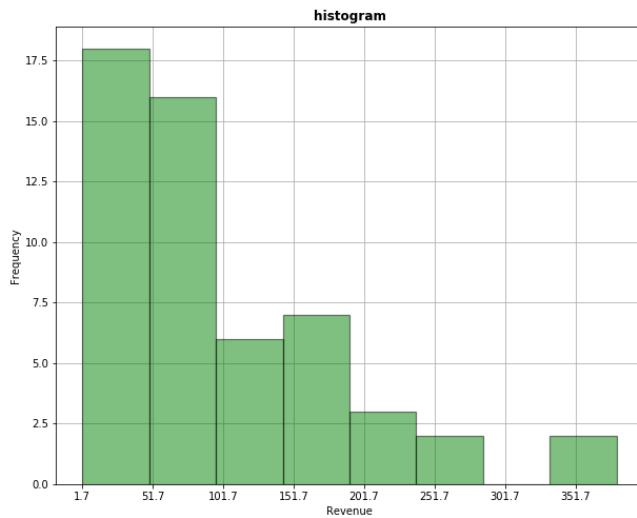


Gambar 4.18: Line plot Jumlah Revenue dari tahun ke tahun

Gambar 4.18 di atas merupakan visualisasi akumulasi *revenue* dari tahun ke tahun untuk semua film pada *dataset* menggunakan *line plot*. Berdasarkan visualisasi ini, dapat disimpulkan bahwa dari tahun ke tahun akumulasi *revenue* film semakin meningkat sehingga semakin menguntungkan kecuali pada tahun 2011. Pada tahun 2011 terjadi penurunan akumulasi *revenue*.



(a) Histogram Revenue Tahun 2010



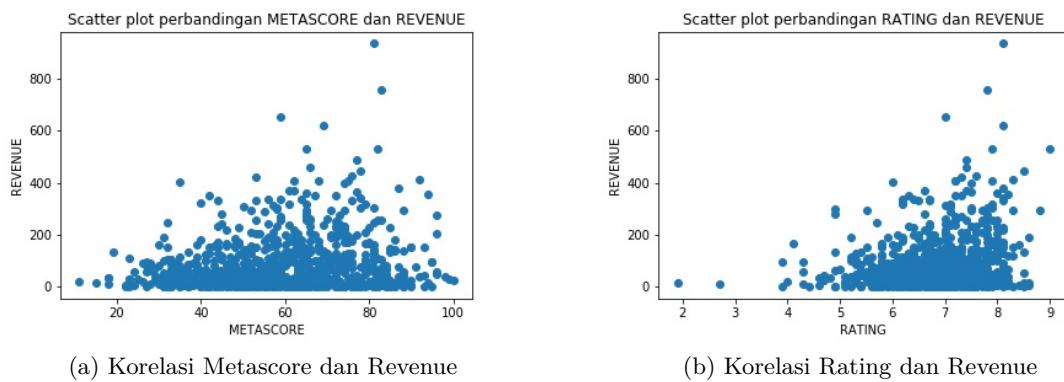
(b) Histogram Revenue Tahun 2011

Gambar 4.19: Histogram Revenue Tahun 2011

Gambar 4.19 adalah perbandingan distribusi *revenue* film yang dihasilkan pada tahun 2010 dan 2011 menggunakan *histogram*. Berdasarkan visualisasi, tahun 2011 memang mengalami penurunan karena lebih banyak film yang menghasilkan *revenue* kecil dibanding distribusi pada tahun 2010.

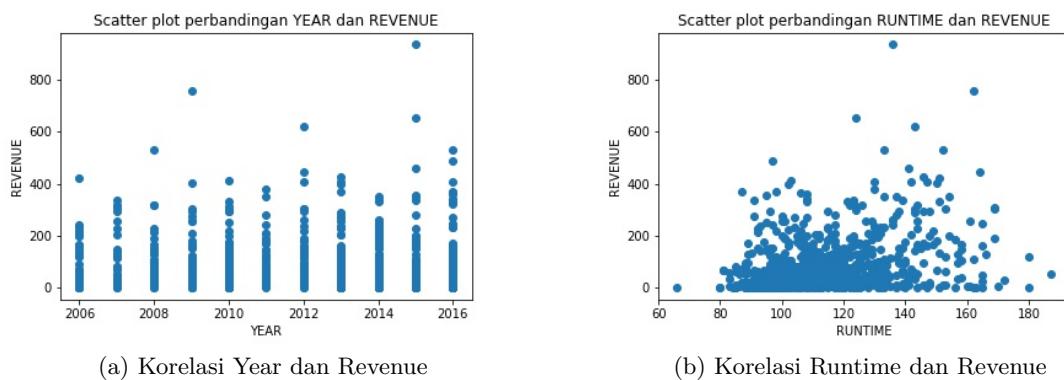
#### 4.4.3 Data Selection

Pada subbab ini dilakukan analisis tentang fitur / kolom yang memiliki korelasi dengan *revenue* sebagai penentu kesuksesan sebuah film. Berdasarkan apa yang sudah dijelaskan pada Bab 2, visualisasi dengan *scatter plot* dan analisis korelasi numerik *pearson* dapat digunakan untuk menentukan korelasi. Fitur yang memiliki fitur terbaik digunakan sebagai fitur prediktor untuk memprediksi *revenue*. *Dataset* ini memiliki 5 kolom yang menjadi prediktor yaitu *rating*, *metascore*, *runtime*, *year* dan *votes*. *Revenue* akan menjadi target yang akan diprediksi.



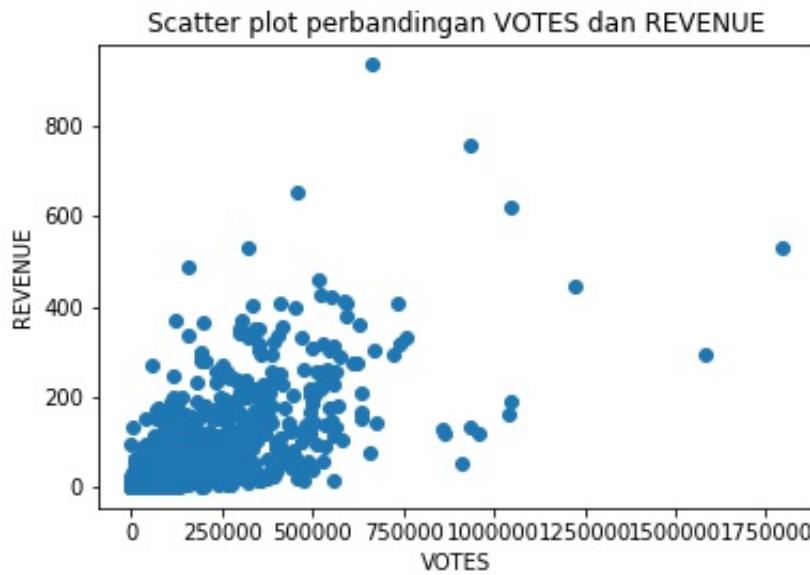
Gambar 4.20: Analisis Korelasi antara situs review dengan revenue

Gambar 4.20 merupakan *scatter plot* metascore revenue dan rating revenue. Pada gambar tersebut ingin diidentifikasi apakah metascore atau rating mempengaruhi revenue. Berdasarkan gambar yang ditunjukkan rating cenderung lebih memiliki korelasi positif dengan revenue.



Gambar 4.21: Analisis Korelasi runtime year dan revenue

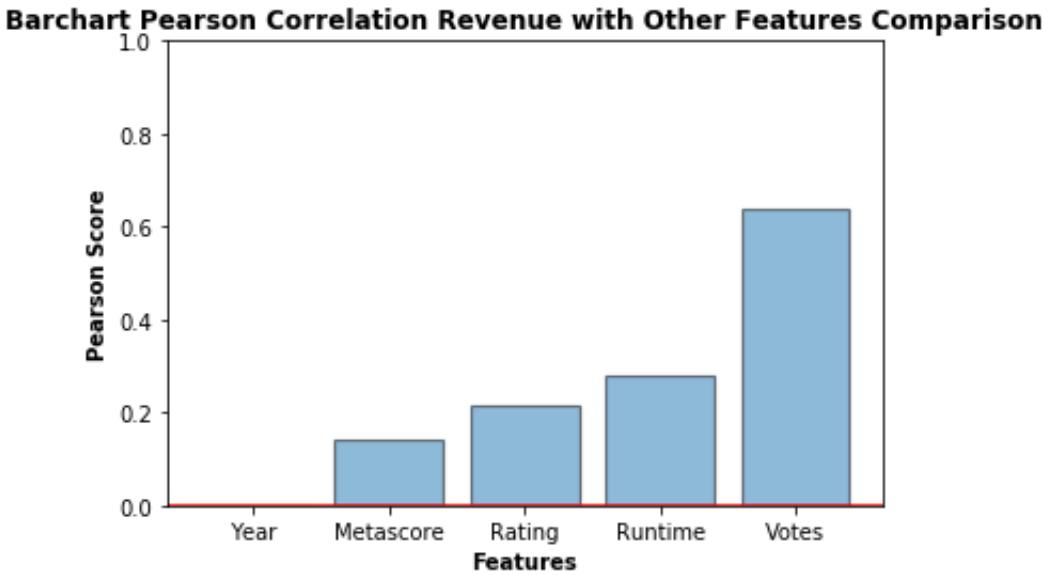
Gambar 4.21 merupakan *scatter plot* yang menunjukkan hubungan antara *year revenue* dan *runtime revenue*. Terlihat jelas bahwa *year* tidak ada pengaruh sama sekali terhadap *revenue*. Semakin baru film dirilis tidak menjamin film itu akan sukses menghasilkan keuntungan. Hubungan antara *runtime* dan *revenue* tidak menunjukkan adanya korelasi yang kuat. Durasi film tidak mempengaruhi seberapa bagus film.



Gambar 4.22: Scatter plot votes dan revenue

Selain *runtime*, *year*, *rating* dan *metascore*, *votes* adalah salah satu kolom numerik yang dapat dianalisis hubungan korelasinya dengan *revenue*. Gambar 4.22 adalah visualisasi *scatter plot* votes dan revenue. Hasil visualisasi menunjukkan bahwa votes memiliki korelasi positif dengan revenue. Semakin banyak votes yang diperoleh sebuah film di situs *review* IMDB, maka semakin tinggi kemungkinan film itu akan memperoleh keuntungan.

Selain visualisasi menggunakan *scatter plot*, pengujian *pearson correlation* berdasarkan subbab 2.3.5 dapat digunakan untuk mengukur korelasi antara 2 fitur.



Gambar 4.23: Pengujian perbandingan korelasi tiap fitur dengan revenue menggunakan pearson correlation

Gambar 4.23 merupakan *barchart* perbandingan nilai *pearson revenue* dengan fitur lain. Gambar tersebut menunjukkan bahwa dari semua fitur numerik, *votes* memiliki korelasi yang paling tinggi. Nilai *pearson* dari *votes* dan *revenue* adalah 0.6. *Pearson runtime* dan *revenue* adalah 0.3. *Pearson rating* dan *revenue* adalah 0.2. *Pearson metascore* dan *revenue* adalah 0.1. Dapat disimpulkan

*votes* dapat mempengaruhi revenue yaitu pendapatan kotor dari film.

#### 4.4.4 Percobaan Prediksi Fitur Data Utama

Pada subbab ini dilakukan percobaan membuat model regresi yang dapat memprediksi keuntungan film yaitu revenue. Prediksi regresi menggunakan *Linear Regression* dan *Polynomial Regression*. Berdasarkan subbab 4.4.3, fitur yang memiliki korelasi terbaik adalah *votes*. Pada tahap ini dilakukan pengujian prediksi *revenue* menggunakan *votes*.

##### Prediksi Linear Regression Data Utama

*Dataset* yang digunakan dibagi menjadi 2 bagian yaitu *training* dan *test*. *Dataset* dibagi menjadi 80 persen dan 20 persen *test*. Untuk semua pengujian, 80 persen data pertama akan dijadikan *train* dan sisanya dijadikan data *test*. Data *training* akan digunakan untuk menghasilkan fungsi prediksi dari regresi. Fungsi regresi yang dihasilkan akan digunakan untuk prediksi pada data *test* sehingga akurasi / skor R2 dapat diperoleh.

Berdasarkan hasil data *training*, fungsi prediksi *revenue* adalah

$$\text{revenueLinearRegression} = 15.093793130383503 + 0.00036708749219229494(\text{Votes}) \quad (4.1)$$

Persamaan 4.1 adalah fungsi yang dihasilkan dari regresi menggunakan *Linear Regression*. Fungsi pengujian ini dapat digunakan untuk melakukan percobaan prediksi pendapatan kotor/ revenue. Berikut adalah contoh 5 data film yang diambil dari data *test* dan hasil prediksinya menggunakan fungsi *Linear Regression*.

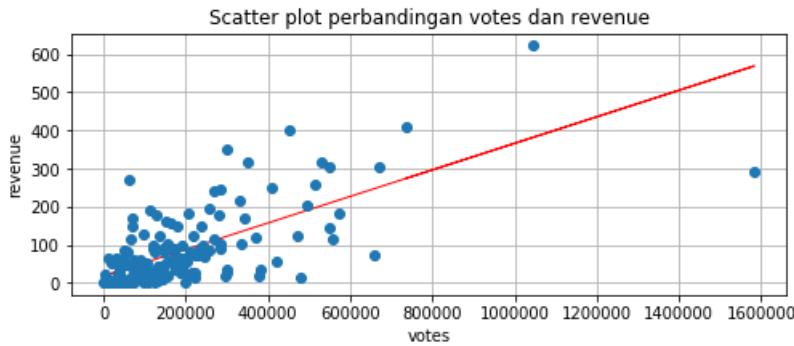
Tabel 4.6: Tabel Percobaan Regresi Linear untuk prediksi revenue menggunakan votes

Title	Votes	Revenue Asli Dataset	Revenue Hasil Prediksi Linear
300:Rise of Empire	237887	106.37	102.4191
Casino Royale	495106	167.01	196.841
True Grit	254904	171.03	108.6659
Norman	664	2.27	15.33753923 6
Self/less	67196	12.28	39.7606

Tabel 4.6 dapat digunakan untuk melihat perbandingan nilai revenue asli dan hasil prediksi yang dihasilkan oleh *Linear Regression* dengan fitur *votes*.

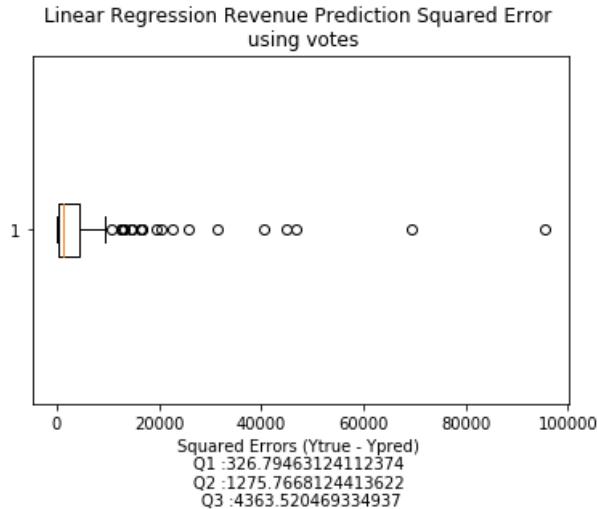
##### Evaluasi Prediksi Linear Regression Data Utama

Pengujian prediksi iterasi ini menggunakan 2 dimensi data yaitu prediksi revenue dengan *votes* sehingga visualisasi *Linear Regression* dapat dibuat. Visualisasi dilakukan dengan membuat *scatter plot* dan menarik garis fungsi yang dihasilkan dari *Linear Regression* menggunakan fitur *votes* pada Gambar 4.24.



Gambar 4.24: Plot linear regression prediksi revenue dengan votes

Berikut adalah pengujian evaluasi *Linear Regression* dengan melihat distribusi *squared error* menggunakan *boxplot*. *Squared Error* (SE) adalah ukuran seberapa jauh hasil prediksi dengan nilai asli dari *revenue*.



Gambar 4.25: Distribusi nilai *squared error*

Gambar 4.25 adalah distribusi nilai *squared error* dari setiap percobaan prediksi pada data *test* menggunakan *boxplot*. Terdapat nilai SE yang sangat tinggi artinya *revenue* asli dan prediksi yang tidak akurat.

### Prediksi Polynomial Regression Data Utama

Selain *Linear Regression*, percobaan prediksi *revenue* juga dilakukan dengan menggunakan *Polynomial Regression* dengan orde = 2. Hasil kedua algoritma yang digunakan akan dibandingkan performanya dalam melakukan prediksi *revenue*. Fungsi prediksi *revenue* menggunakan *votes* yang dihasilkan *polynomial regression* adalah.

$$\text{revenuePolynomial} = 1.162 + 0.000507 * (\text{votes}) + -1.78E - 10 * (\text{votes})^2 \quad (4.2)$$

Berdasarkan persamaan 4.2, akan dilakukan prediksi 5 data *test* dengan fungsi polinom yang dihasilkan.

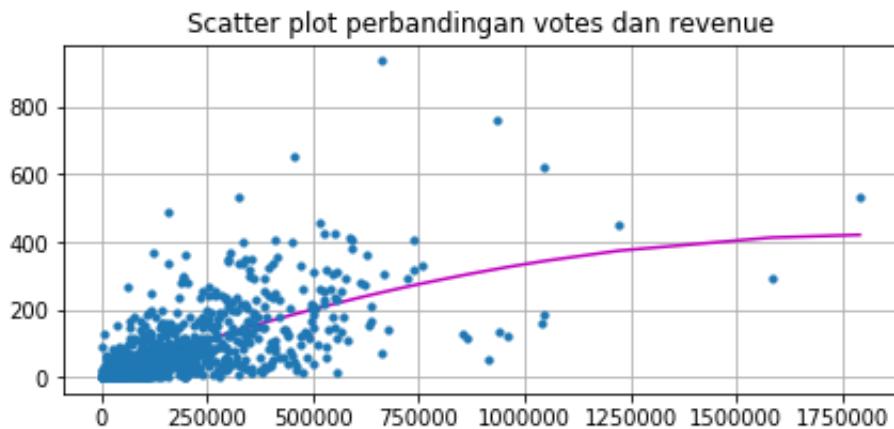
Tabel 4.7: Tabel Percobaan Regresi Polinom untuk prediksi revenue menggunakan votes

Votes	Revenue Asli Dataset	Revenue Hasil Prediksi Polinom
237887	106.37	111.78
495106	167.01	208.67
254904	171.03	118.92
664	2.27	1.50
67196	12.28	34.45

Tabel 4.7 di atas adalah tabel perbandingan revenue asli dengan prediksi menggunakan fungsi polinom votes. Terdapat hasil prediksi yang jauh pada data ke-2 dan ke-3.

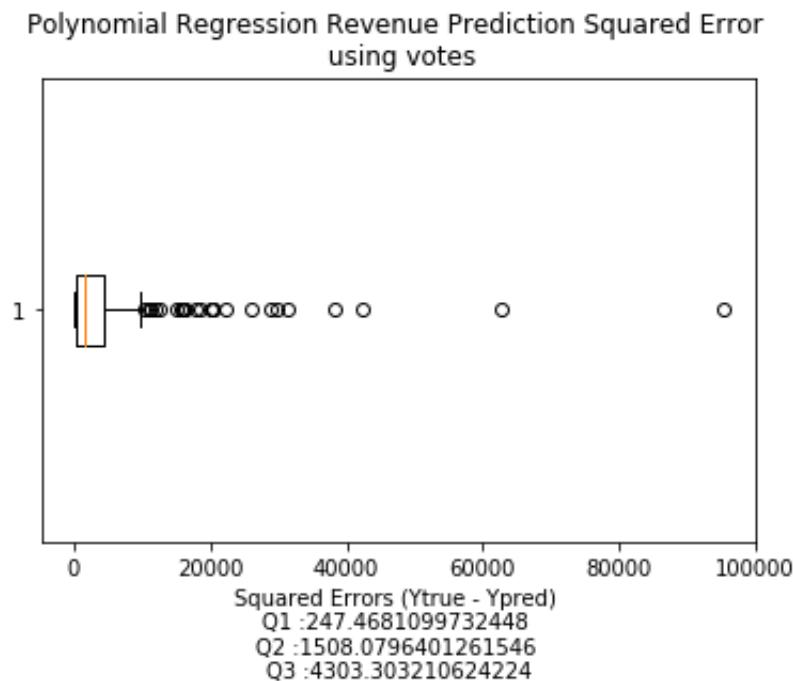
### Evaluasi Prediksi Polynomial Regression Data Utama

Karena fungsi polinom dihasilkan dari 2 dimensi menggunakan votes untuk memprediksi revenue, maka visualisasi kurva polinom yang dihasilkan adalah.



Gambar 4.26: Plot polynominal regression prediksi revenue dengan votes

Gambar 4.26 adalah visualisasi kurva yang dihasilkan fungsi polinom menggunakan *scatter plot*. Gambar tersebut dan Gambar 4.24 dapat menunjukkan perbedaan garis yang dihasilkan. Fungsi linear menghasilkan garis lurus dan fungsi polinom menghasilkan kurva landai. Berikut adalah distribusi nilai *squared error* dari hasil prediksi menggunakan *polynomial regression*. Visualisasi distribusi menggunakan *boxplot*.



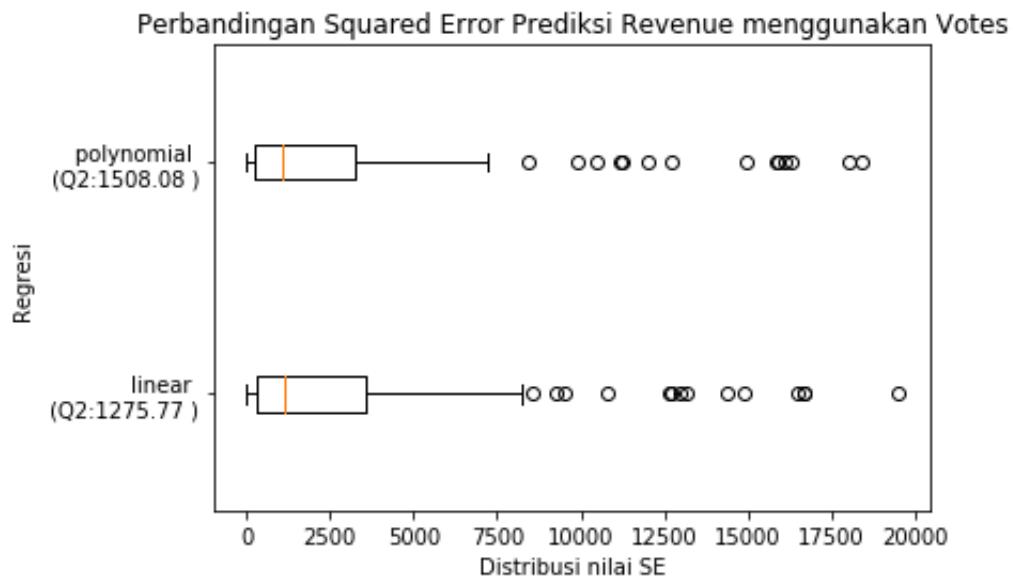
Gambar 4.27: Distribusi nilai squared error prediksi revenue menggunakan *polynomial regression*

### Pengujian Prediksi Data Utama

Berdasarkan percobaan prediksi dengan fungsi linear dan polinom pada data *test*, akan dilakukan pengujian akurasi nilai ketepatan prediksi menggunakan nilai *coefficient determination* (R2).

Tabel 4.8: Tabel perbandingan skor akurasi r2 menggunakan regresi

Nilai R2 Fungsi Linear	Nilai R2 Fungsi Polinom
0.38	0.36



Gambar 4.28: Distribusi nilai squared error prediksi revenue dan perbandigan tiap algoritma regresi yang digunakan

Tabel 4.8 di atas adalah hasil pengujian skor akurasi fungsi prediksi pada fungsi linear dan polinom. Berdasarkan subbab 2.4.1, R<sup>2</sup> dapat digunakan untuk menguji seberapa baik prediksi yang dihasilkan. Berdasarkan Gambar 4.28, nilai Q<sup>2</sup> dari *error polynomial* lebih besar dari *linear*. Berdasarkan tabel di atas skor akurasi linear lebih baik dari polinom tetapi, skor yang dihasilkan fungsi linear masih belum tepat untuk melakukan prediksi karena skor yang dihasilkan relatif masih kecil.

## 4.5 Proses Analisis Data Tambahan

Berdasarkan informasi kesimpulan yang dihasilkan Subbab 4.4 yaitu subbab sebelumnya, model prediksi yang dihasilkan oleh *Linear Regression* dan *Polynomial Regression* masih belum tepat untuk dijadikan model prediksi karena skor akurasi yang kecil yaitu 0.38. Hasil analisis *revenue* yang dilakukan juga masih kurang dikarenakan mengukur kesuksesan dengan *revenue* tidak cukup. Analisis *revenue* secara keseluruhan tidak dapat diterapkan karena sebuah film masih memiliki faktor penentu lain yang dominan yaitu *budget*.

Tahap-tahap yang dilakukan selama proses analisis data tambahan adalah :

- Melakukan pengumpulan data kolom tambahan *budget* menggunakan *Octoparse* dan API *library* IMDB
- Melakukan *Data Integration* untuk menggabungkan *dataset* dengan data tambahan *budget*
- Melakukan *Data Transformation* untuk membuat kolom *profit* dan *Return of Investment* (ROI) berdasarkan *budget*
- Melakukan analisis menggunakan visualisasi untuk menemukan pola menarik dari data tambahan
- Melakukan percobaan prediksi *revenue* dan *profit* berdasarkan fitur menarik menggunakan regresi

Penjelasan secara detail mengenai proses analisis data tambahan dapat dilihat pada subbab berikutnya. Berdasarkan hasil analisis data tambahan yang dilakukan, informasi yang didapat adalah :

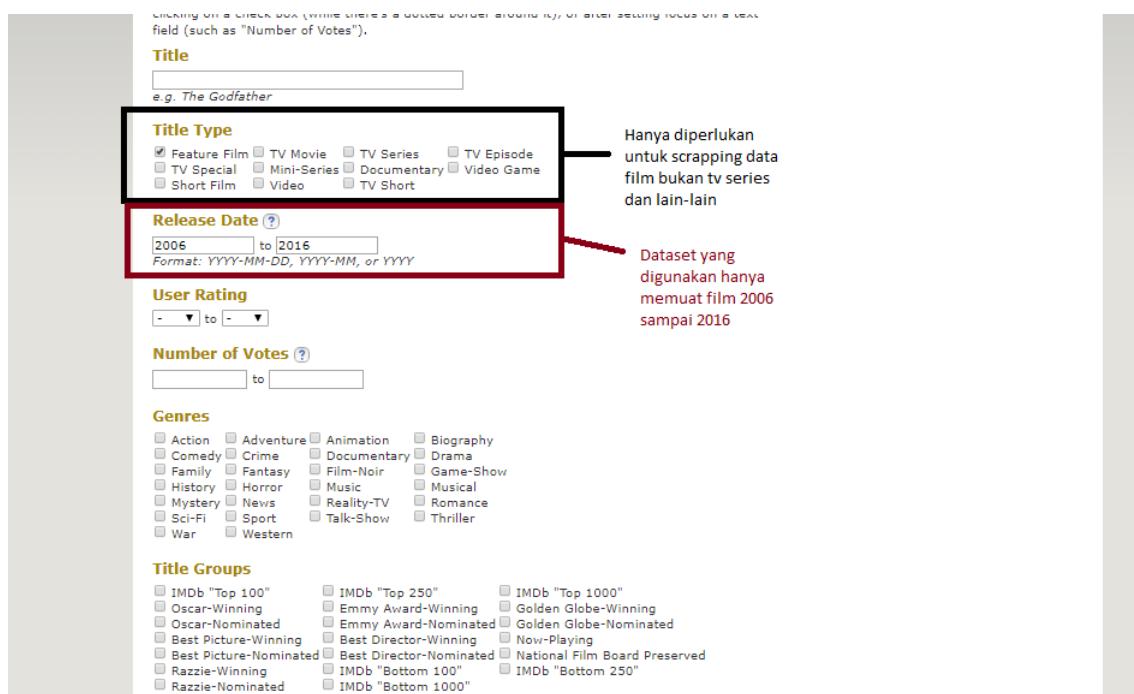
- Hasil data yang diperoleh dari *scraping* tidak selalu dalam keadaan bersih / sesuai kebutuhan
- Sebuah film dapat memperoleh *revenue* yang tinggi tetapi belum tentu menghasilkan keuntungan (*profit*)
- Sebuah film dapat mengalami kerugian jika pengeluaran *budget* yang dikeluarkan lebih besar dari pendapatan yang diperoleh *revenue*
- Sekitar 61 persen film dari *dataset* memperoleh keuntungan (*revenue > budget*) dan 39 persen memperoleh kerugian (*budget > revenue*)
- Akumulasi *budget* dan *profit* meningkat tiap tahunnya
- *Budget* dan *votes* memiliki korelasi yang tinggi dengan *revenue* sehingga penting untuk membuat film yang sesuai keinginan penonton dan menggunakan *budget* semaksimal mungkin untuk kualitas.
- Nilai evaluasi R2 prediksi *revenue* meningkat dari 0.38 menjadi 0.60 sehingga model prediksi *revenue* semakin membaik
- Nilai evaluasi R2 prediksi *profit* sangat kecil yaitu 0.29 sehingga tidak dapat digunakan untuk model prediksi. Hal ini disebabkan oleh data film-film yang mengalami kerugian (*profit* negatif) sehingga korelasi semakin melemah
- Film *genre Horror/Mystery/Thriller* adalah *genre* film yang dapat menguntungkan walaupun dengan *budget* yang kecil
- Film *genre Animation* adalah film yang sangat *profitable* tetapi membutuhkan *budget* yang besar
- Film *Adventure,Fantasy* adalah salah satu kombinasi *genre* terpopuler

#### 4.5.1 Data Collection

Pengumpulan data *budget* pada *dataset* dilakukan dengan 2 cara yaitu menggunakan *Web Scraping tool* *Octoparse* dan *Library open source API* dari situs IMDB yaitu *IMDBPy*.

#### Pengumpulan Menggunakan Octoparse

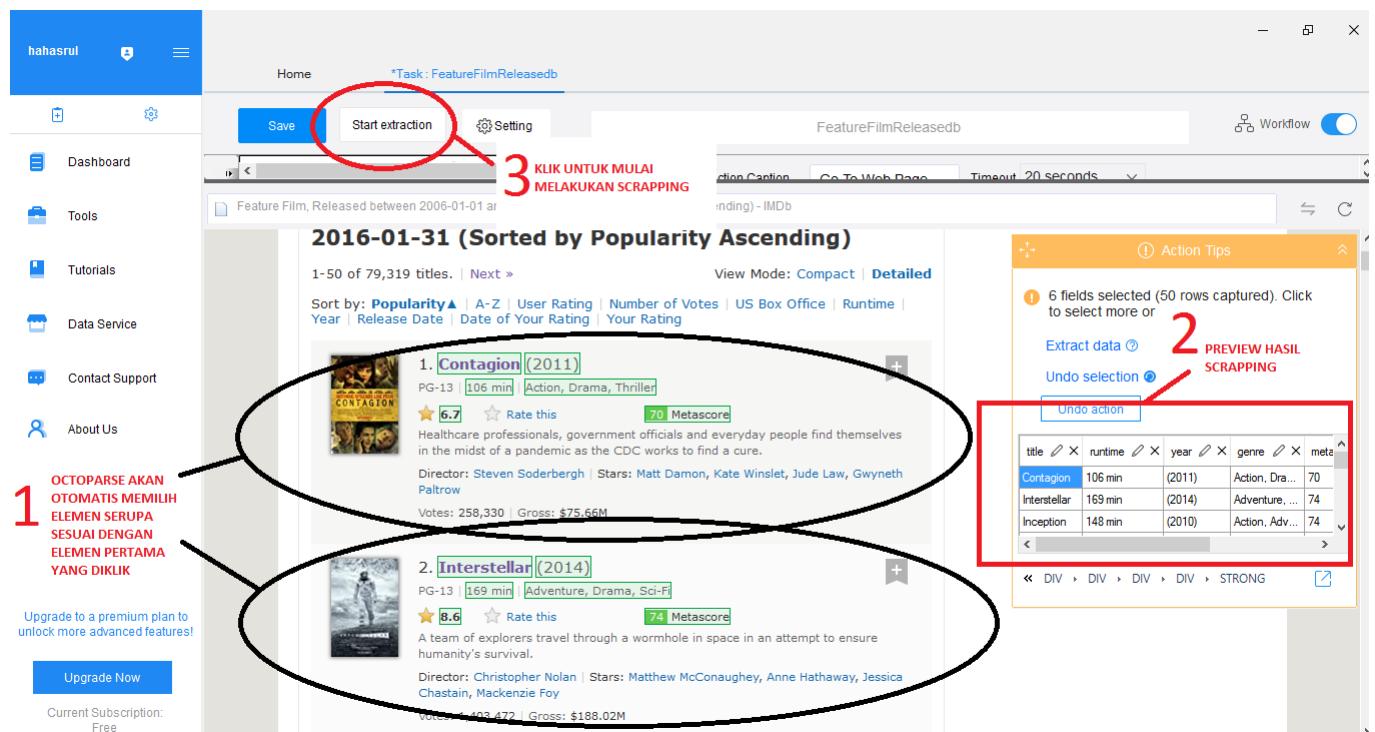
Berdasarkan penjelasan 3.16, perangkat lunak *tool Octoparse* dapat digunakan untuk mengambil data teks pada sebuah *website*. Pada proses pengumpulan data ini dilakukan pengambilan data *budget* untuk setiap film pada *dataset*. *Octoparse* hanya dapat mengambil data pada halaman tetapi tidak dapat mengambil data film secara spesifik. Melakukan *scraping* pada semua data film di IMDB akan memakan waktu yang sangat lama dan memperlambat proses analisis. Untuk dapat mengatasi masalah yang dijelaskan, akan dimanfaatkan fitur *advanced search* dari situs IMDB.



Gambar 4.29: Fitur advance search pada situs IMDB

Gambar 4.29 adalah fitur *advanced search* pada Situs IMDB. Situs IMDB menuliskan bahwa situs ini memiliki lebih dari 6.5 juta data film. Fitur *advanced search* ini akan membantu proses *scraping* karena hanya akan melakukan *scraping* pada data yang sesuai dengan kriteria *dataset* yang digunakan untuk penelitian. *Dataset* yang diperoleh adalah film terkenal dari tahun 2006 sampai 2016. *Advanced search* pada situs IMDB dapat dimanfaatkan untuk mengambil data khusus sesuai *filter* yang diterapkan sehingga mengurangi waktu untuk melakukan *scraping* pada data yang tidak dibutuhkan.

Hasil dari *advanced search* adalah sebuah hasil *query* dalam bentuk *list*. Sebuah elemen *list* adalah satu objek film beserta data yang dibutuhkan berdasarkan kriteria pencarian pada *advanced search* yang sudah diterapkan. Tipe halaman dengan berisi *list* ini yang dapat dimanfaatkan *Octoparse* agar dapat mengambil data yang dibutuhkan.



Gambar 4.30: Pengaturan octoparse pada hasil pencarian advanced search

Gambar 4.30 adalah tampilan pengaturan *Octoparse* untuk melakukan *scraping* pada data IMDB. *Octoparse* secara otomatis memberikan rekomendasi untuk elemen yang serupa (film urutan ke-2). Elemen pada *list* merupakan elemen *clickable* yang dapat ditekan *user* untuk masuk halaman detail sebuah film. *List* tersebut harus diklik untuk dapat mengakses data yang lebih detail yaitu *budget*.

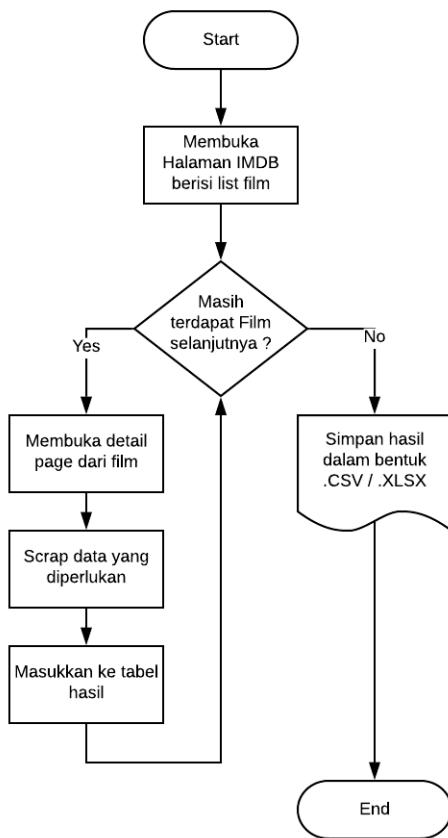
Gambar 4.31: Detail page sebuah film pada situs IMDB

Gambar 4.31 adalah halaman ketika sebuah film pada gambar sebelumnya diklik. Data *budget*

Tabel 4.9: Sampel contoh data hasil scraping budget IMDB menggunakan Octoparse

title	rating	director	budget	runtime
Inception (2010)	8.8	Christopher Nolan	Budget: \$160,000,000	148 min
The Boy (2016)	6.0	William Brent Bell	Budget: \$10,000,000	97 min
Sing (2016)	7.1	Garth Jennings	Budget: \$ 75,000,000	108 min
Moonlight (2016)	7.4	Barry Jenkins	Filming Location : Miami	•

akan tersedia di halaman detail sebuah film. *Flowchart* alur pengambilan data *budget* menggunakan *Octoparse*.



Gambar 4.32: Flowchart cara kerja scraping budget IMDB menggunakan Octoparse

Gambar 4.32 adalah *flowchart* cara kerja *Octoparse* untuk melakukan *scraping* data tambahan pada situs IMDB. Hasil *scraping* data IMDB menggunakan berjumlah 9337 data film.

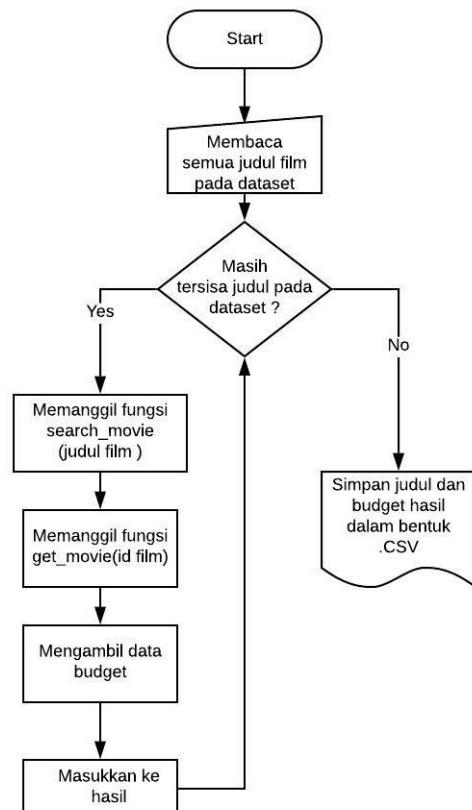
Tabel 4.9 adalah contoh data hasil *scraping* menggunakan *Octoparse*. Terdapat data *budget* yang berhasil diambil dan tidak. Data yang tidak berhasil diperoleh disebabkan karena data *budget* tidak ditampilkan pada halaman sehingga *Octoparse* membaca elemen lain untuk digantikan yaitu *filming location*. Berdasarkan eksperimen pengumpulan data dengan *Octoparse*, data yang diperoleh tidak dapat diintegrasikan ke *dataset* yang dimiliki. Pada tahap selanjutnya dilakukan percobaan pengumpulan data dengan cara lain yaitu mengakses *open source API* dari IMDB.

### Pengumpulan Menggunakan API

Situs IMDB menyediakan *open source API* yang dapat diakses untuk mengambil data film sesuai kebutuhan. Bahasa pemrograman *Python* membuat *wrapper library* bernama *IMDBpy* untuk mengakses API dari situs IMDB. *Library* ini memiliki beberapa *method* yang dapat dimanfaatkan :

- *IMDBpy.search\_movie(title)* : *method* untuk melakukan pencarian film dengan parameter *input string* judul film. *Method* ini mengembalikan kumpulan objek berisi judul film dan id film tersebut
- *IMDBpy.get\_movie(id)* : *method* untuk mengambil data film secara lengkap. *Method* ini menerima *input* id dari film yang ingin dicari datanya. *Method* ini mengembalikan sebuah objek film dalam bentuk *dictionary* dengan pasangan *key value* berisi detail data film. *Key* merupakan alamat data yang ingin diambil. *Value* merupakan data yang dihasilkan dari memanggil *key* pada *dictionary*
- *movie.infoset2keys()* : *method* untuk melihat kumpulan *key* beserta informasi yang dihasilkan

Dengan memanfaatkan *method* yang disediakan *IMDBpy*, berikut adalah *flowchart* alur kerja pengambilan data *budget* menggunakan *library* *IMDBpy*



Gambar 4.33: Flowchart cara kerja scraping budget IMDB menggunakan library *IMDBpy*

Hasil *scraping* dari *IMDBpy* yang dilakukan adalah sebuah *dataset* dengan kolom judul dan budget dari film tersebut. Jumlah baris dari data tambahan yang diperoleh berjumlah 838 data objek, sesuai dengan jumlah semua film pada *dataset*.

Tabel 4.10: Tabel 10 sampel hasil data scrapping menggunakan IMDBpy

title	budget
Guardians of the Galaxy	\$170,000,000 (estimated)
Furious 6	\$160,000,000 (estimated)
No Country for Old Men	\$25,000,000 (estimated)
The Great Gatsby	\$105,000,000 (estimated)
Shutter Island	\$80,000,000 (estimated)
Django Unchained	\$100,000,000 (estimated)
Busanhaeng	KRW10,000,000 (estimated)
The Dressmaker	AUD17,000,000 (estimated)
Criminal	0
The Best of Me	0

Tabel 4.10 adalah beberapa sampel hasil *scraping* yang dilakukan. Data yang diberikan merupakan estimasi *budget* dari setiap film. Mayoritas film memiliki *budget* dalam mata uang USD. Terdapat beberapa baris yang data *budget* diberikan dalam mata uang selain dollar. Ada beberapa film juga yang memang tidak menyimpan data *budget* sehingga diberikan nilai 0. Hasil data yang berhasil diperoleh digunakan untuk analisis subbab selanjutnya.

#### 4.5.2 Data Integration

Data tambahan *budget* yang sudah diperoleh sebelumnya akan digunakan untuk digabungkan dengan *dataset*. Cara penggabungan akan dilakukan dengan operasi *merge*. Tiap baris pada *dataset* dipasangkan dengan baris pada data *budget*. Masing-masing tabel memiliki kolom unik yang serupa yaitu *title* sehingga *title* akan dijadikan *key*.

Tabel 4.11: Tabel sampel data integrasi antara dataset dengan budget

title	genre	budget
Guardians of the Galaxy	Action,Adventure,Sci-Fi	\$170,000,000 (estimated)
Prometheus	Adventure,Mystery,Sci-Fi	\$130,000,000 (estimated)
Split	Horror,Thriller	\$90,000,000 (estimated)

#### 4.5.3 Data Transformation

Data *budget* yang diperoleh pada subbab sebelumnya adalah data biaya yang dikeluarkan untuk membuat sebuah film. Masing-masing film memiliki *budget* yang bervariasi. Berdasarkan Tabel 4.11, data *budget* belum bisa dianalisis karena :

- Data *budget* masih berbentuk tipe data *string* (contoh : "\$130,000,000")
- Masih terdapat karakter untuk menentukan mata uang (contoh : "KRW" , "\$" , "EUR")
- Ada data yang memiliki mata uang yang berbeda

Berdasarkan permasalahan *budget* yang dijelaskan sebelumnya, maka dilakukan *data transformation* untuk mengubah data *budget* menjadi bentuk yang lebih sesuai. Kolom *revenue* sebagai pendapatan kotor memiliki tipe *float* dan disimpan dalam satuan juta *dollar* sehingga kolom *budget* perlu diubah menjadi bentuk yang sesuai dengan *revenue*.

Tabel 4.12: Tabel sampel beberapa konversi budget pada dataset

Title Film	Budget Sebelum	Operasi Konversi	Pengali Mata Uang	Budget Sesudah
Guardians of the Galaxy	\$170,000,000	-	-	170
Furious 6	\$160,000,000	-	-	160
Busanhaeng	KRW10,000,000	KRW -> USD	0.00082	8.2
The Dressmaker	AUD17,000,000	AUD -> USD	0.65	11.05
...	...	...	...	...

Tabel 4.12 adalah tabel sampel data pada *dataset* yang diubah sesuai format yang serupa dengan *revenue*. Terdapat film yang memiliki mata uang yang berbeda sehingga harus diubah secara manual menggunakan perangkat lunak Microsoft Excel. Setelah semua *budget* sudah diubah dalam bentuk *dollar*, lalu *budget* diubah dalam bentuk satuan juta USD.

*Dataset* yang digunakan sudah memiliki kolom *budget* yang sesuai untuk dianalisis. Pada tahap selanjutnya dilakukan *data transformation* dengan teknik *attribute construction* sesuai dengan subbab 2.3.4. *Attribute construction* adalah teknik membuat kolom baru berdasarkan perhitungan kolom yang ada. Pada tahap ini dilakukan dua kali *Attribute construction* untuk membuat kolom pendapatan bersih (profit) dan *return of investment* (ROI).

$$\text{Pendapatan Bersih/Profit} = \text{Revenue} - \text{Budget} \quad (4.3)$$

Persamaan 4.3 adalah cara menghitung pendapatan bersih / *profit*. Berikut adalah tabel 5 film sampel pada *dataset* pada Tabel 4.13.

Tabel 4.13: Tabel 5 sampel pembuatan kolom profit pada dataset

title	revenue	budget	revenue - budget = profit
Ted 2	81.26	68	81.26 - 68 = 13.26
The Conjuring 2	102.46	40	102.46 - 40 = 62.46
Toy Story 3	414.98	200	414.98-200 = 214.98
The Skin I Live In	3.19	13	3.19 - 13 = (-9.81)
Nine Lives	19.64	30	19.46 - 40 = (-20.54)

Tabel 4.13 adalah 5 sampel yang diambil dari *dataset* dan perhitungan *profit* tiap datanya untuk menghasilkan kolom baru. Selain kolom *profit*, dilakukan *attribute construction* untuk membuat kolom *Return Of Investment* (ROI). ROI adalah laba di atas pengeluaran. ROI adalah persentase seberapa untung laba berdasarkan perbandingannya dengan pengeluaran / *budget*. Berikut adalah cara menghitung ROI.

$$ROI = \text{profit}/\text{budget} * 100 \quad (4.4)$$

Persamaan 4.4 adalah rumus yang dapat digunakan untuk memperoleh ROI sebuah film. Berikut adalah 5 sampel contoh perhitungan ROI pada *dataset*.

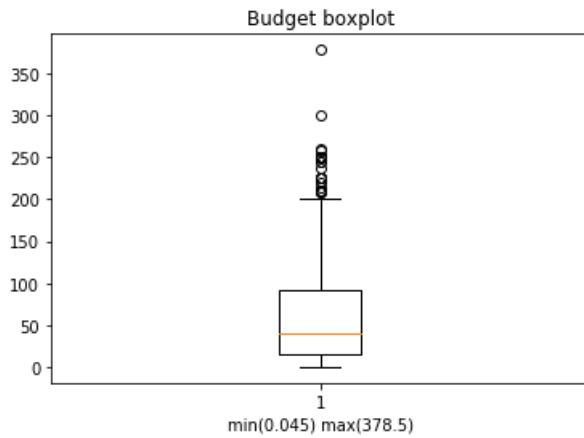
Tabel 4.14: Tabel 5 sampel pembuatan kolom roi pada dataset

title	budget	profit	roi = profit / budget * 100
Spider-Man 3	259	78.53	$\frac{78.53}{259} * 100 = 30.43\%$
The Intern	35	40.27	$\frac{40.27}{35} * 100 = 115.06\%$
Silver Linings Playbook	21	111.09	$\frac{111.09}{21} * 100 = 529\%$
A United Kingdom	14	-10.1	$\frac{-10.1}{14} * 100 = -72.14\%$
The Curious Case of Benjamin Button	150	-22.51	$\frac{-22.51}{150} * 100 = -15.01\%$

Tabel 4.14 adalah contoh 5 sampel perhitungan ROI pada *dataset*. Berdasarkan contoh yang diberikan, terdapat film yang memperoleh kerugian bukan keuntungan karena *profit* dan *roi* bernilai negatif. Nilai negatif berarti pengeluaran yang dikeluarkan lebih besar dari keuntungan yang didapat. Pada subbab selanjutnya akan dilakukan analisis visualisasi terhadap *dataset* dengan kolom baru.

#### 4.5.4 Analisis Data Tambahan Menggunakan Visualisasi

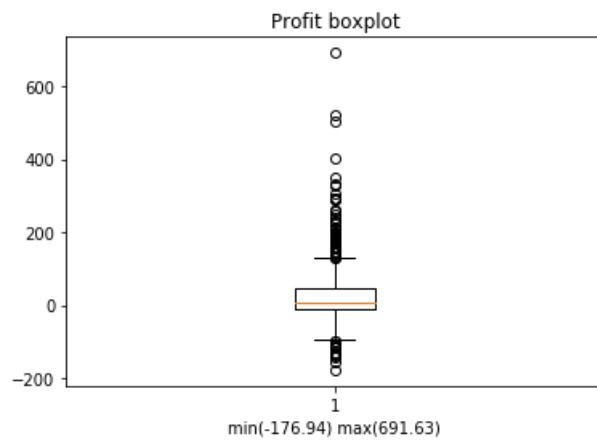
Berdasarkan data tambahan yang sudah diperoleh pada subbab sebelumnya, dilakukan visualisasi distribusi nilai kolom tambahan yaitu *budget*, *profit* dan *ROI*.



Gambar 4.34: Distribusi nilai kolom budget

Gambar 4.34 adalah visualisasi distribusi *budget* menggunakan *boxplot*. Visualisasi ini menunjukkan bahwa untuk membuat film dengan modal yang kecil. Nilai minimum *budget* pada *dataset* ini adalah 0.045. Nilai tersebut ditulis secara satuan juta *dollar* sehingga nilai minimum dari *budget* adalah 45,000 *dollar*.

#### Hubungan Profit Sebagai Keuntungan



Gambar 4.35: Distribusi nilai kolom profit

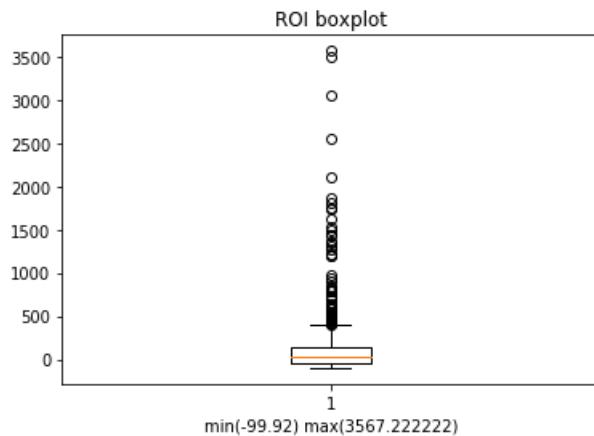
Gambar 4.35 adalah visualisasi distribusi *profit* menggunakan *boxplot*. Berdasarkan *boxplot profit* yang ditunjukkan, usaha film ternyata bisa menyebabkan kerugian ketika modal / *budget* yang dikeluarkan jauh lebih besar dari keuntungan yang diperoleh. Nilai minimum dari *profit* yang dapat

diperoleh adalah  $-179.94$  juta USD. Berikut adalah perbandingan film yang rugi dan menguntungkan pada Gambar 4.36



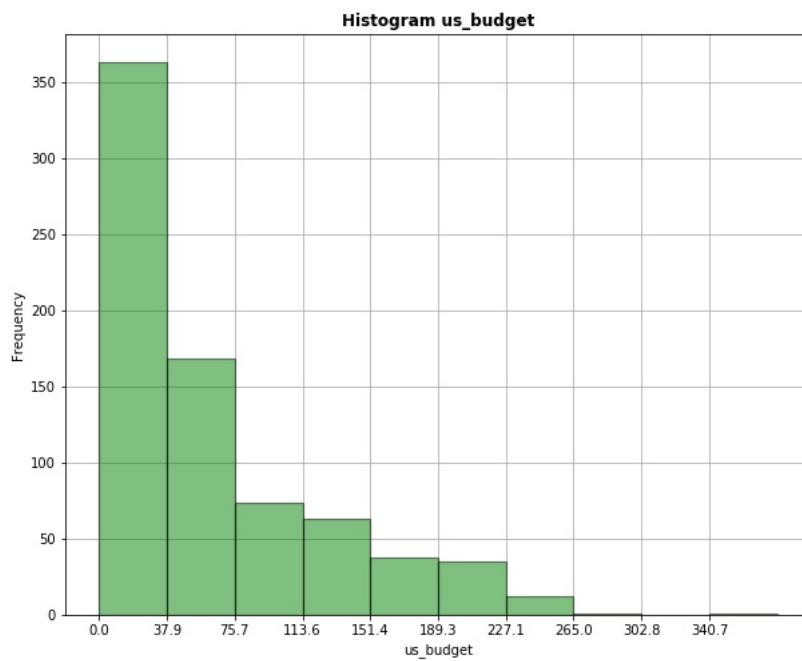
Gambar 4.36: Perbandingan film yang rugi dan untung pada dataset

### Analisis ROI

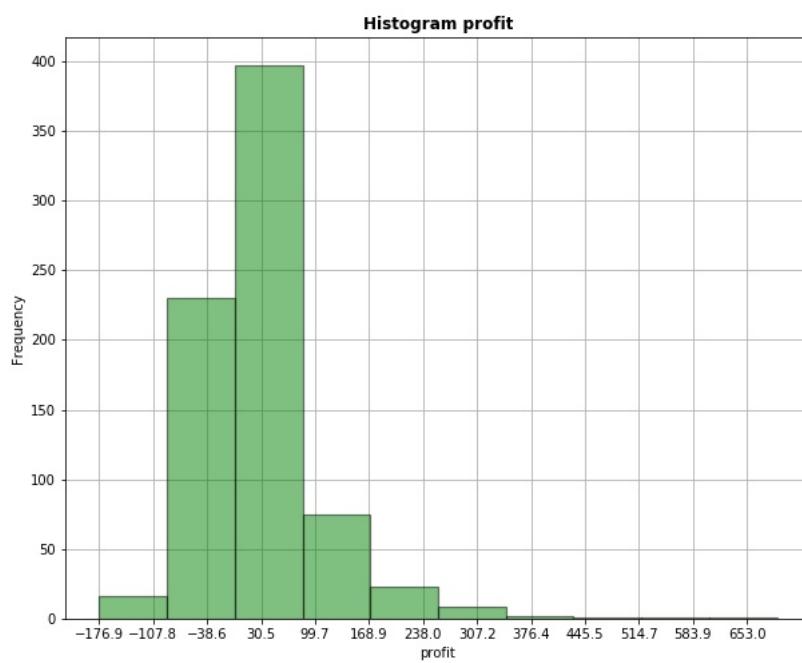


Gambar 4.37: Distribusi nilai kolom ROI

Gambar 4.37 adalah visualisasi distribusi *roi* dataset menggunakan *boxplot*. Berdasarkan *boxplot* *ROI* yang ditunjukkan, sebuah film bisa saja rugi sampai 2 kali lipat. Dapat dilihat bahwa nilai minimumnya adalah  $-99.92$ . Artinya sebuah film bisa memperoleh kerugian sampai 99 persen tetapi, ada film yang bisa memperoleh keuntungan sampai jauh lebih tinggi sampai lebih dari 100 persen. Selain visualisasi dengan *boxplot*, maka diberikan visualisasi distribusi menggunakan *histogram*.

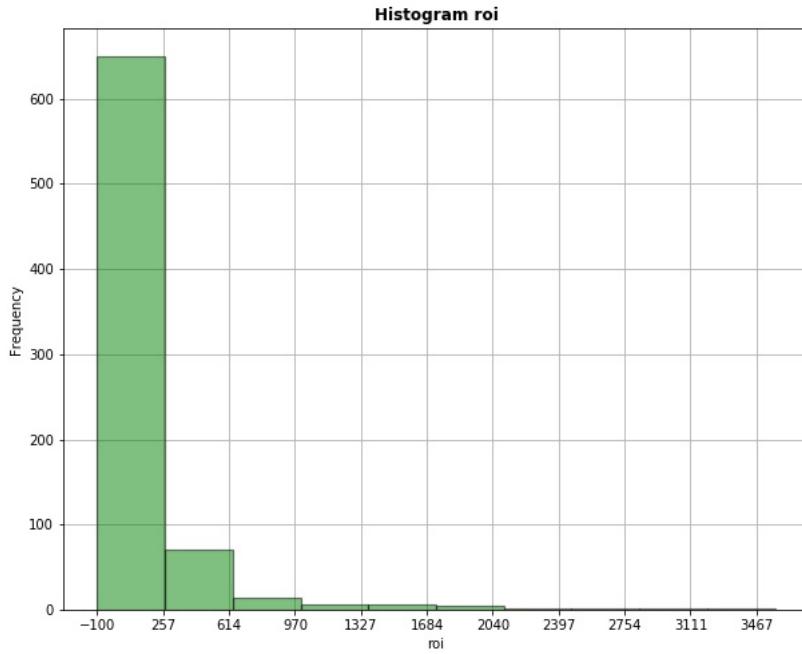
**Analisis Histogram**

Gambar 4.38: Histogram kolom Budget



Gambar 4.39: Histogram kolom Profit

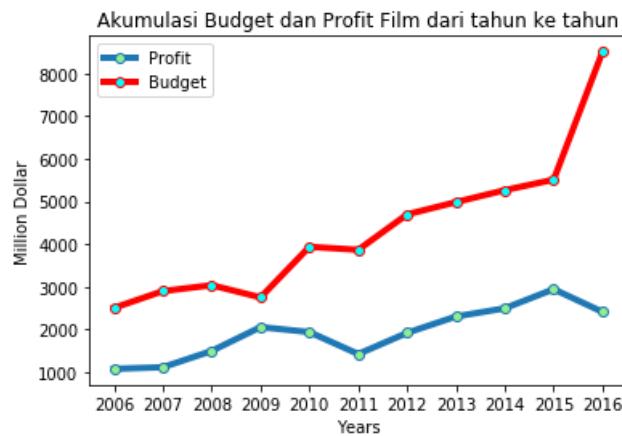
Gambar 4.38 dan 4.39 adalah perbandingan *histogram budget* dengan *profit*. Hasil visualisasi dapat menunjukkan bahwa umumnya sebuah banyak film yang menghabiskan *budget* pada kelompok 0 sampai 30 juta *dollar*. Selain itu, visualisasi *histogram profit* menunjukkan bahwa banyak film dapat memperoleh sekitar 90 juta *dollar*.



Gambar 4.40: Histogram kolom ROI

Gambar 4.37 adalah persebaran frekuensi kelompok ROI pada *dataset*. Berdasarkan persebaran distribusinya mayoritas dari film pada *dataset* memperoleh dari  $-100\%$  sampai  $200\%$ . Sebuah film bisa saja rugi atau memperoleh keuntungan sampai 2 kali lipat. Pada tahap ini dilakukan analisis akumulasi *budget* dan *profit* film dari tahun ke tahun.

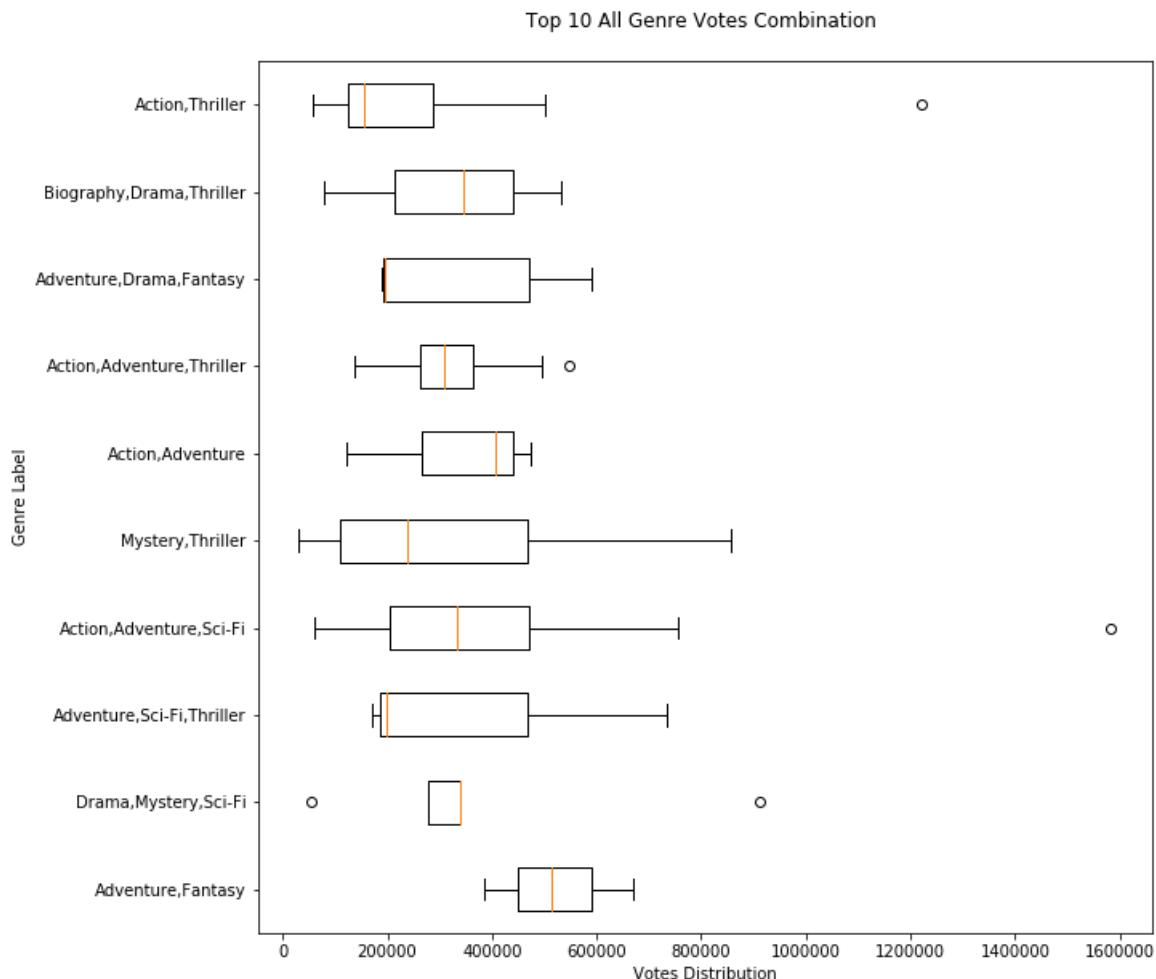
### Tren Budget Dan Profit Dari Tahun ke Tahun



Gambar 4.41: Line plot akumulasi profit dan budget semua film tiap tahun

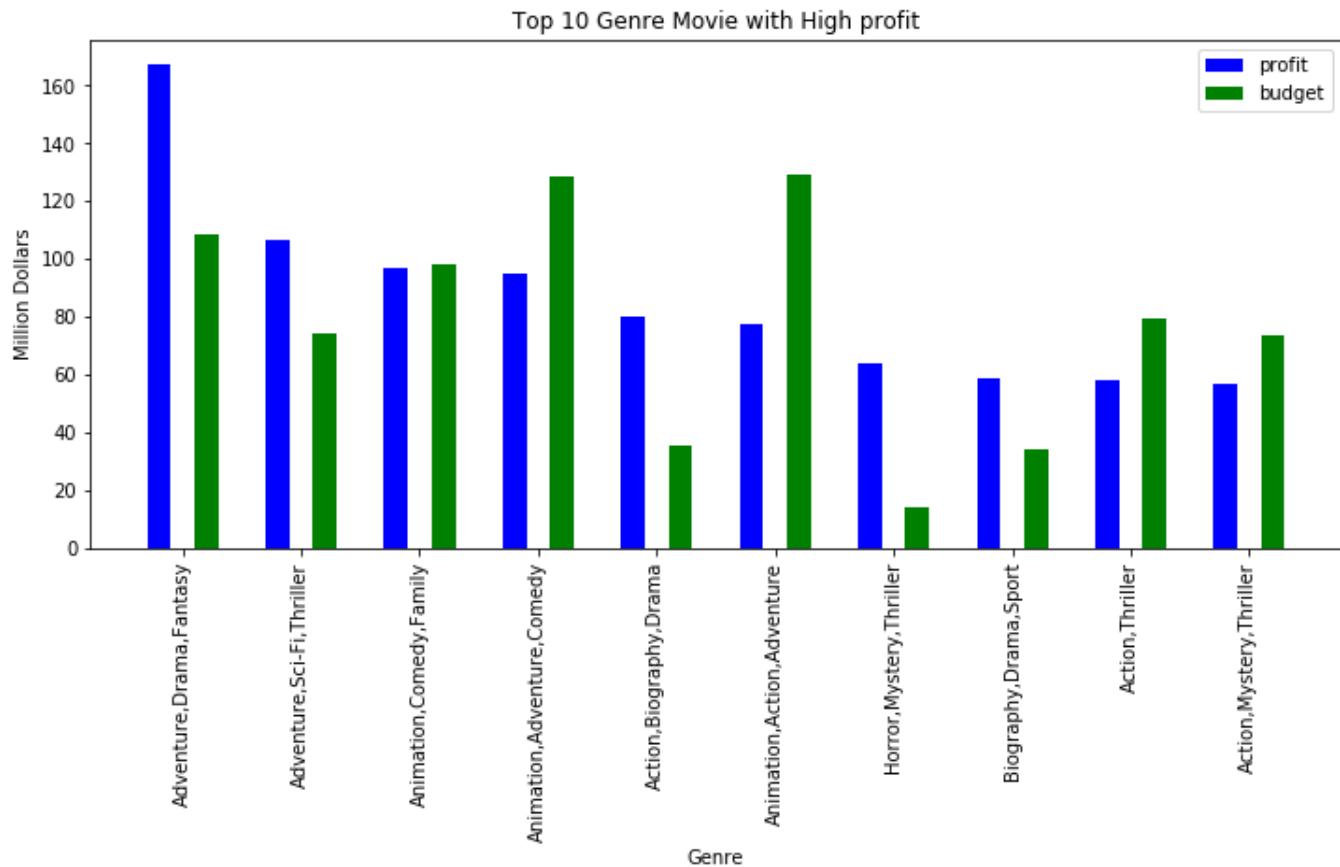
Gambar 4.41 adalah analisis akumulasi *budget* dan *profit* dari tahun ke tahun menggunakan *line plot*. Berdasarkan visualisasi dapat disimpulkan bahwa *profit* keuntungan dari membuat film tiap tahun meningkat. *Profit* meningkat seiring dengan *budget* yang harus dikeluarkan. Ada peningkatan signifikan dari tahun 2015 ke tahun 2016. Hal ini disebabkan oleh data film yang rilis tahun 2016 jauh lebih banyak dari data film selain 2016 pada *dataset*.

Berdasarkan kesimpulan pada subbab 4.4 pada bagian *data selection*, kolom *votes* adalah kolom yang memiliki korelasi positif paling tinggi terhadap keuntungan yaitu *revenue*. Pada tahap ini dilakukan analisis distribusi *votes* yang diperoleh tiap kombinasi *genre* film. Berikut adalah visualisasi 10 kombinasi *genre* yang memiliki distribusi *votes* yang paling besar menggunakan *boxplot*.



Gambar 4.42: Top 10 Kombinasi genre dengan votes terbaik

Gambar 4.42 di atas adalah kombinasi *genre* yang memiliki distribusi *votes* tertinggi. Kombinasi *genre* dengan nilai Q2 terbaik adalah *adventure,fantasy*. Berdasarkan 10 *votes* terbaik *Thriller*, *Adventure* dan *Action* adalah *genre* paling sering muncul. Referensi ini dapat dijadikan *feedback* untuk pembuat film *genre* mana yang memiliki dukungan *user* paling banyak.



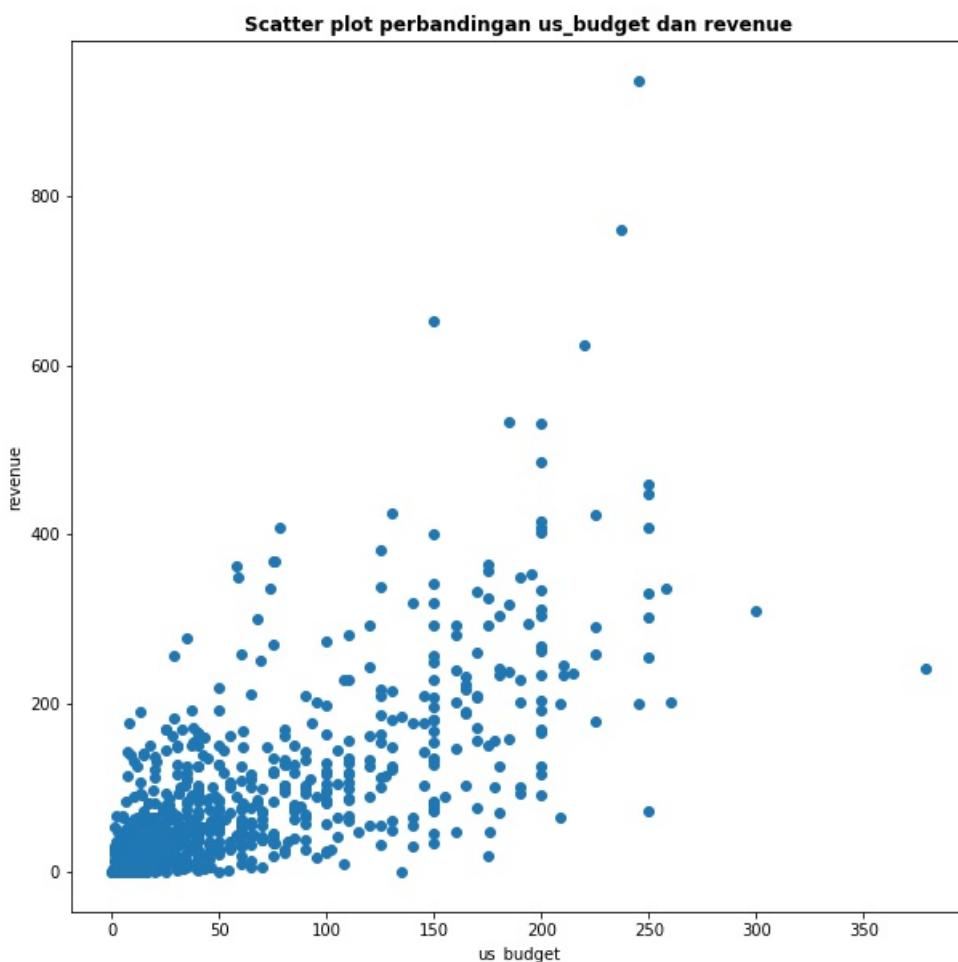
Gambar 4.43: 10 kombinasi genre dengan perbandingan budget dan revenue tertinggi (profit)

Gambar 4.43 adalah visualisasi 10 kombinasi *genre* terbaik dengan *profit* tertinggi menggunakan *barchart*. Visualisasi ini dapat memberikan *feedback* kepada pembuat film kombinasi *genre* mana saja yang menghasilkan *profit*. Film *Mystery*, *Horror* dan *Thriller* adalah kombinasi *genre* yang *budget* yang dikeluarkan tidak terlalu tinggi tetapi masih menghasilkan keuntungan. Film yang mengandung *genre animation* membutuhkan *budget* yang besar tetapi masih menghasilkan keuntungan yang besar juga. Kombinasi *genre* terbaik adalah "*Adventure,Drama,Fantasy*".

#### 4.5.5 Data Selection

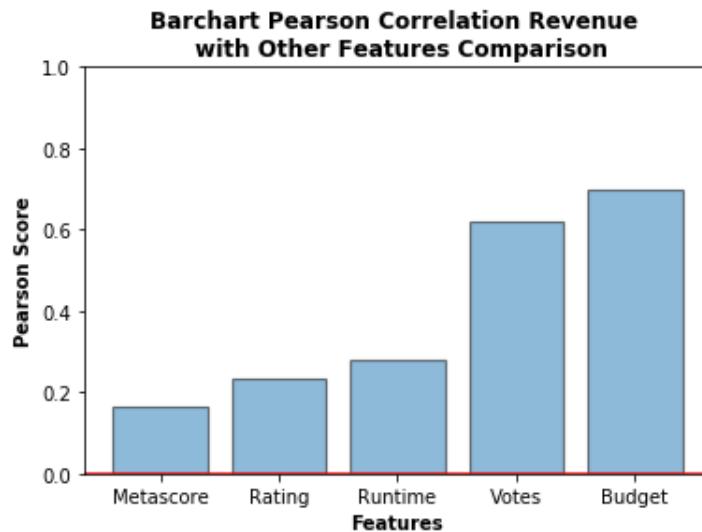
Pada subbab ini dilakukan analisis korelasi terhadap kolom tambahan yaitu *budget*, *profit* dan *roi* dengan fitur prediktor yang ada pada *dataset*. Analisis dilakukan dengan 2 cara yaitu visualisasi korelasi dengan *scatter plot* dan pengujian nilai *pearson*. Fitur prediktor yang memiliki korelasi positif yang tinggi akan dipilih untuk menjadi prediktor untuk prediksi *revenue*, *profit* dan *roi*.

Ingin dianalisis apakah *budget* dan *revenue* / pendapatan kotor memiliki korelasi positif satu sama lain. Berikut adalah visualisasi korelasi *revenue* dan *budget* dengan *scatter plot*.



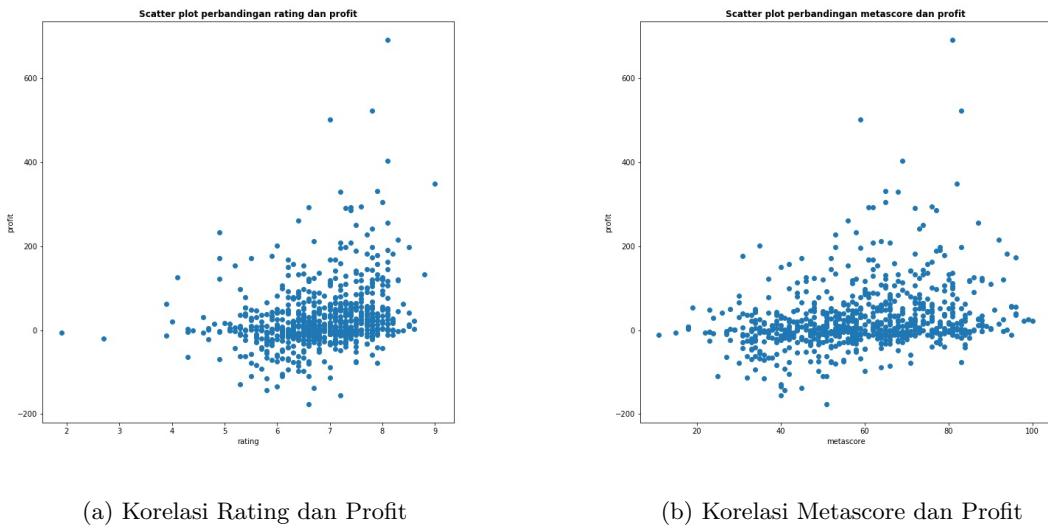
Gambar 4.44: Analisis korelasi kolom Budget dan Revenue

Gambar 4.44 menunjukkan bahwa *budget* dan *revenue* memiliki korelasi positif yang tinggi. Dapat dilihat semakin besar *budget* yang dikeluarkan, maka semakin tinggi *revenue* yang diperoleh. *Budget* bisa menjadi salah satu fitur yang dapat dipilih untuk memprediksi *revenue*. Berikut adalah pengujian korelasi tiap fitur prediktor dengan *revenue* menggunakan *pearson correlation* untuk memilih fitur prediksi terbaik.

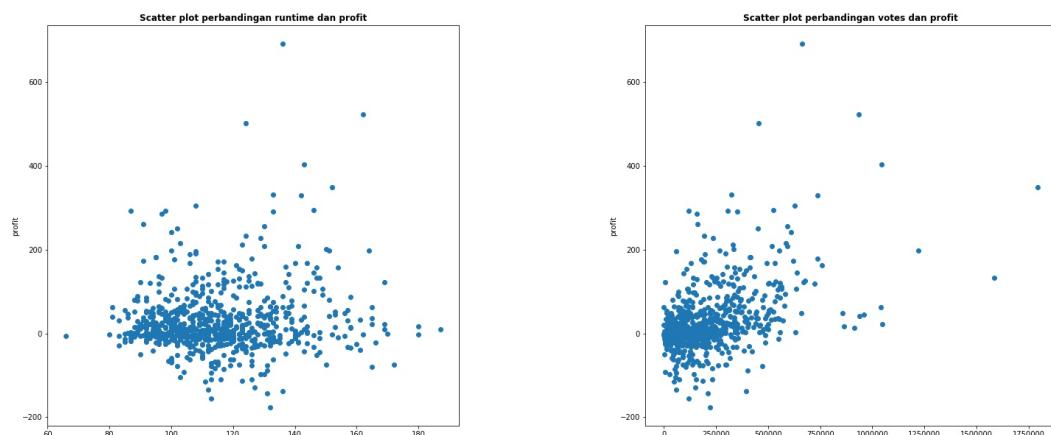


Gambar 4.45: Pengujian perbandingan korelasi tiap fitur dengan revenue menggunakan pearson correlation ke-2

Gambar 4.45 adalah pengujian nilai *pearson* tiap fitur prediktor dan target yaitu *revenue* menggunakan *barchart*. Berdasarkan pengujian sebelumnya pada subbab 4.4, *votes* merupakan fitur dengan nilai *pearson*. Tetapi, fitur *budget* ternyata lebih tinggi korelasinya dengan *votes*. Berdasarkan penjelasan *pearson* pada subbab 2.3.5, nilai *pearson* terbaik adalah 1.0. *Budget* memiliki nilai korelasi yaitu 0.7 sehingga memiliki korelasi yang tinggi. Selain *revenue*, dilakukan analisis korelasi fitur terhadap *profit*. Berikut adalah visualisasi korelasi *profit* dan fitur prediktor lain dengan *scatter plot*.



Gambar 4.46: Analisis korelasi situs review dan profit



(a) Korelasi Runtime dan Profit

(b) Korelasi Votes dan Profit

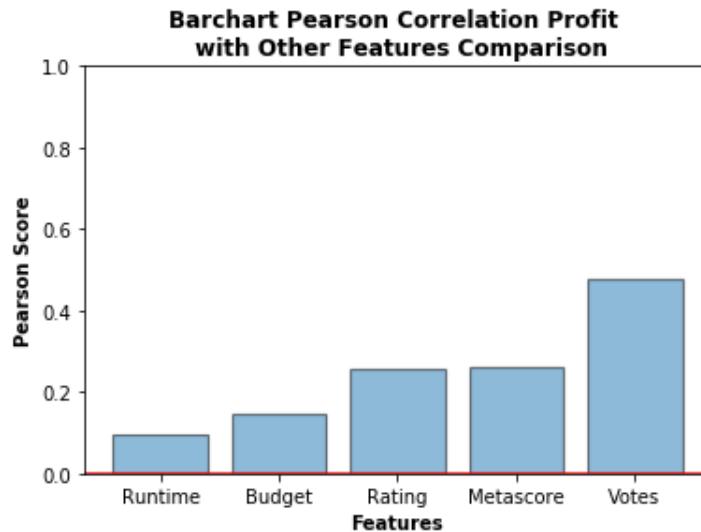
Gambar 4.47: Analisis korelasi runtime votes dan profit



Gambar 4.48: Analisis korelasi budget dan profit

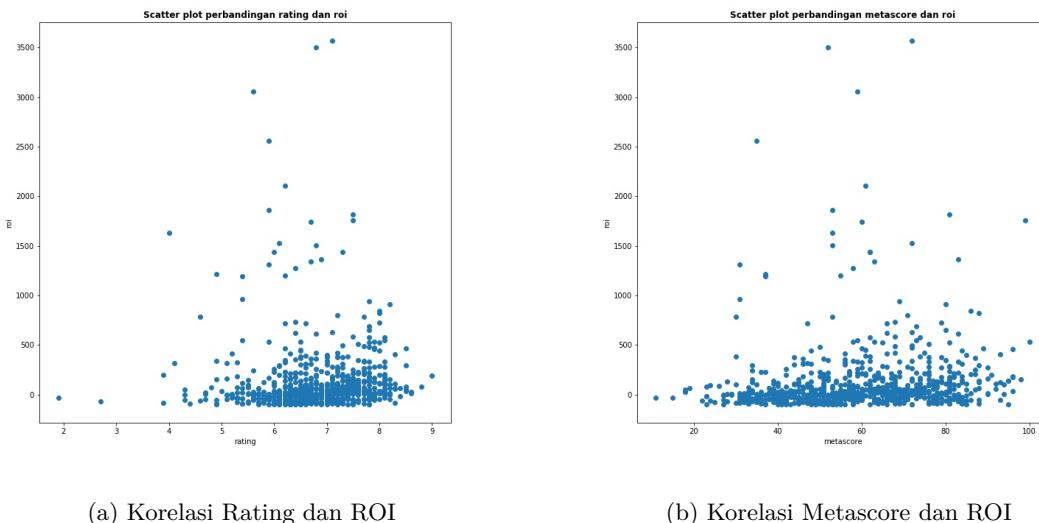
Gambar 4.8, 4.47 dan 4.52 adalah visualisasi korelasi tiap fitur dengan *profit* menggunakan *scatterplot*. Dibanding dengan korelasi *revenue*, korelasi *profit* dengan tiap fitur lebih tidak menun-

juukkan korelasi yang positif. Hal ini disebabkan karena setelah perhitungan *profit* dengan selisih *budget* dan *revenue* banyak film yang menghasilkan kerugian sehingga korelasi positif berkurang. Berikut adalah pengujian korelasi *profit* dengan fitur prediktor lain berdasarkan *pearson correlation* menggunakan *barchart*.



Gambar 4.49: Pengujian perbandingan korelasi tiap fitur dengan profit menggunakan pearson correlation ke-2

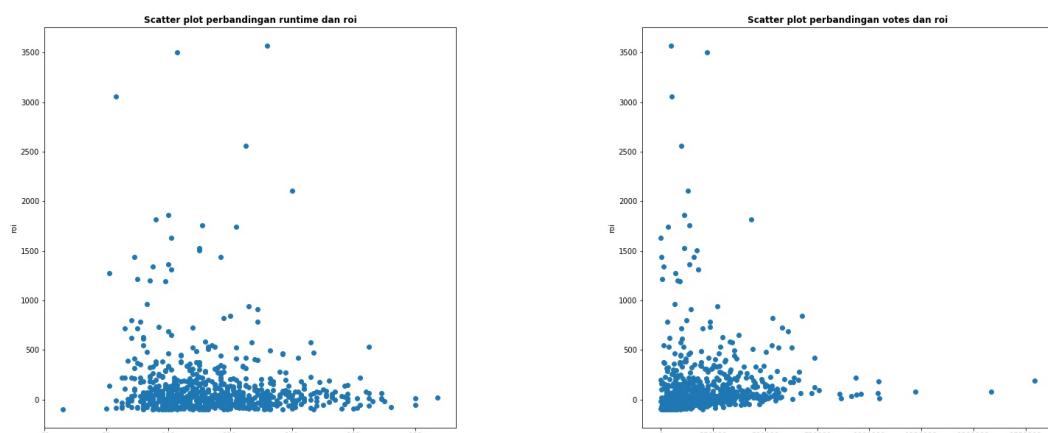
Berdasarkan Gambar 4.49, fitur yang memiliki korelasi positif tertinggi adalah *votes*. Semakin banyak *votes* sebuah film, maka semakin banyak *profit* yang dapat diperoleh. Pengaruh dukungan penonton dapat mempengaruhi peningkatan *profit*. Selain *profit*, dilakukan analisis korelasi *ROI* terhadap fitur prediktor lain menggunakan *scatter plot*.



(a) Korelasi Rating dan ROI

(b) Korelasi Metascore dan ROI

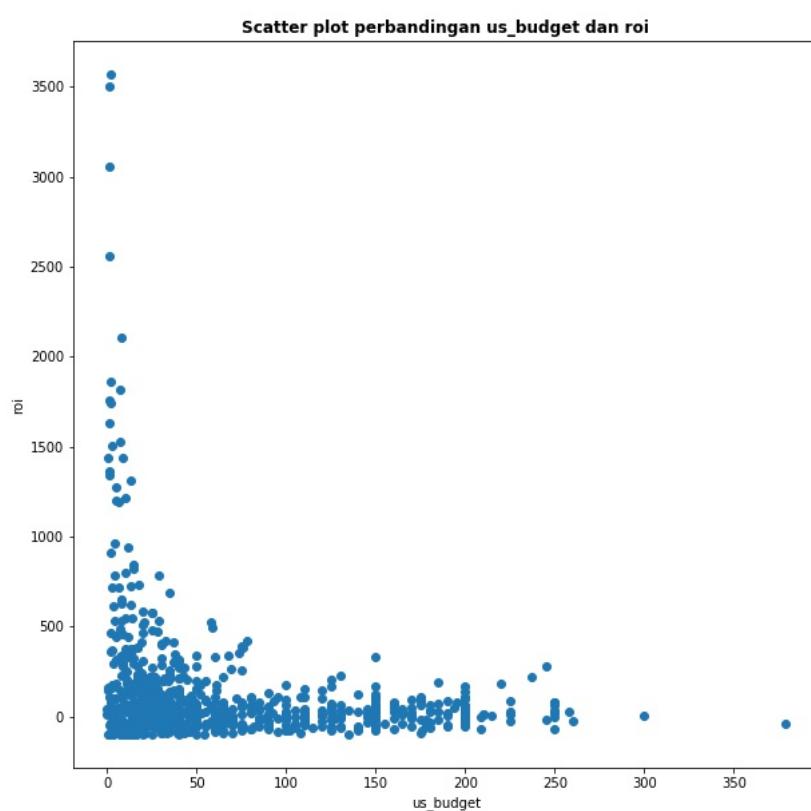
Gambar 4.50: Analisis korelasi situs review dan ROI



(a) Korelasi Runtime dan ROI

(b) Korelasi Votes dan ROI

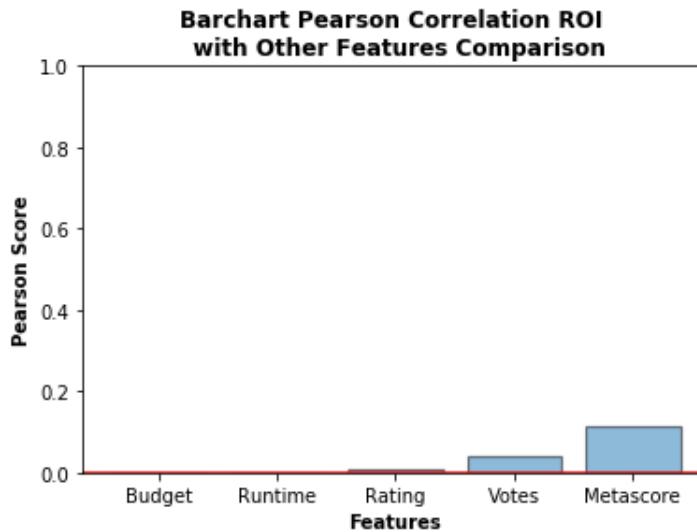
Gambar 4.51: Analisis korelasi runtime votes dan roi



Gambar 4.52: Analisis korelasi budget dan profit

Berdasarkan Gambar 4.50, 4.51 dan 4.52, banyak fitur prediktor yang tidak menunjukkan korelasi positif yang kuat dengan ROI. Berikut adalah pengujian korelasi *ROI* dengan fitur prediktor

berdasarkan *pearson correlation* menggunakan *barchart*.



Gambar 4.53: Pengujian perbandingan korelasi tiap fitur dengan ROI menggunakan pearson correlation ke-2

Berdasarkan Gambar 4.53, tidak ada fitur yang menunjukkan korelasi yang positif dengan *ROI* karena nilai yang dihasilkan sangat kecil. Nilai korelasi yang kecil dan negatif pada beberapa fitur menyebabkan percobaan prediksi *ROI* tidak dapat dilakukan.

#### 4.5.6 Percobaan Prediksi Fitur Data Tambahan

Pada subbab ini dilakukan percobaan membuat model yang dapat memprediksi keuntungan film yaitu *revenue* dan *profit* menggunakan regresi. Percobaan prediksi regresi menggunakan *Linear Regression* dan *Polynomial Regression*. Berdasarkan hasil pengujian *pearson* sebelumnya fitur yang paling berpengaruh adalah *votes* dan *budget* sehingga prediksi memanfaatkan 2 fitur sekaligus untuk membuat model prediksi.

##### Percobaan Prediksi Revenue

Berdasarkan *training model* yang sudah dibuat. Berikut adalah perbandingan nilai R2 prediksi *revenue* menggunakan *votes* dan *budget*.

Tabel 4.15: Tabel Skor R2 Revenue

Fitur Prediktor : votes dan budget	
Skor R2 Linear	Skor R2 Polynom
0.56	0.60

Berdasarkan Tabel 4.15, skor evaluasi R2 pada iterasi ke-2 meningkat dibanding dengan skor R2 iterasi ke-1 pada Tabel 4.8. Pada percobaan kali ini algoritma *Polynomial Regression* memiliki nilai R2 yang lebih tinggi dari *Linear Regression*. Berikut adalah tabel percobaan prediksi *revenue* pada model *votes* dan *budget*.

Tabel 4.16: Tabel sampel percobaan prediksi Revenue menggunakan regresi fitur votes dan budget

Title	Votes	Budget	Revenue Asli	Revenue Linear	Revenue Polynomial
Final Destination 5	88000	40	42.80	46.45	48.61
Twilight	361449	37	191.45	99.61	94.15
Warm Bodies	153579	33	66.36	53.73	58.105
17 Again	152808	20	64.15	41.88	47.00
Spider-Man 3	406219	258	336.53	304	312
...	...	...	...	...	...

### Percobaan Prediksi Profit

Berdasarkan *training model* yang sudah dibuat. Berikut adalah perbandingan nilai R2 prediksi *profit* menggunakan *votes* dan *budget*.

Tabel 4.17: Tabel skor R2 prediksi profit

Fitur Prediktor : votes dan Budget	
Skor R2 Linear	Skor R2 Polynomial
0.20	0.29

Berdasarkan hasil Tabel 4.17, nilai evaluasi R2 untuk memprediksi *profit* memiliki nilai yang kecil sehingga tidak dapat digunakan untuk prediksi. Berikut adalah perbandingan sampel percobaan prediksi *profit* dengan *profit* asli.

Tabel 4.18: Tabel sampel percobaan prediksi profit menggunakan budget dan votes dan perbandingannya dengan profit asli

Title	Votes	Budget	Profit Asli	Profit Linear	Profit Polynomial
Trolls	38552	125	28.98	-13.51	-59.03
Diary of the Wimpy Kid	34184	15	49	-0.9	6.62
Goosebumps	57602	67	13.02	-2.9	-12.29
Finding Dory	157026	200	286.29	-2.0	-57.303
Piranha 3D	75262	24	1	5.59	11.27
...	...	...	...	...	...

Berdasarkan Tabel 4.18, Hasil prediksi yang dihasilkan untuk memprediksi *profit* sangat buruk. Sehingga model yang dibuat dengan fitur yang terbaik tidak dapat digunakan untuk memprediksi *profit*.

## 4.6 Proses Analisis Data Sosial Media

Berdasarkan informasi yang didapatkan pada Subbab 4.5, fitur yang memiliki korelasi yang paling tinggi adalah *votes* dan *budget*. *Votes* adalah jumlah pengguna situs IMDB yang suka / mendukung sebuah *film* tetapi *votes* hanya mencakup pendukung film yang tergabung dalam situs IMDB. Banyak penikmat film yang menyukai film tetapi tidak berkontribusi untuk menilai sebuah film pada situs IMDB. Akan dilakukan pengujian analisis kesuksesan film berdasarkan data sosial media.

Tahap-tahap yang dilakukan selama proses analisis data media sosial adalah :

- Melakukan pengumpulan data media sosial *Youtube* dan *Instagram*
- Melakukan *data integration* untuk menggabungkan data media sosial dengan *dataset*

- Melakukan analisis data media sosial dengan menggunakan visualisasi
- Melakukan *data selection* untuk memilih fitur terbaik prediksi *revenue* dan *profit*
- Melakukan prediksi *revenue* dan *profit* menggunakan fitur media sosial

Penjelasan secara lebih detail proses analisis data media sosial dijelaskan pada subbab berikutnya. Berdasarkan hasil analisis data media sosial yang dilakukan, informasi yang didapat adalah.

- Pengumpulan data menggunakan API *youtube* sangat lambat karena ada batas kuota tiap harinya
- Pengumpulan data *youtube* dan *instagram* dapat menggunakan *Octoparse* dan memanfaatkan *url* dinamis dari halaman yang ingin dikumpulkan
- Berdasarkan analisis *dataset*, Fitur *view count* (*Youtube*) lebih berpengaruh daripada dibanding *hashtag* (*Instagram*)
- Ada kecenderungan semakin banyak jumlah penonton *trailer film* di *youtube* (*viewCount*), maka semakin meningkat kemungkinan *revenue* dan *profit* yang dapat diperoleh. *Trailer* sebuah film yang menarik dapat membuat penonton tertarik untuk menonton di bioskop
- Ada kecenderungan popularitas aktor dalam media sosial khususnya *instagram* memiliki pengaruh terhadap peningkatan *revenue* sebuah film
- *Genre* film yang paling menguntungkan berdasarkan jumlah penonton *trailer* di *youtube* adalah *Action, Adventure*.
- Nilai evaluasi R2 prediksi *revenue* meningkat dari 0.60 menjadi 0.65 setelah menambah data *Youtube* sebagai prediktor sehingga model prediksi sudah mendekati 0.70 yaitu minimum sebuah model yang dapat digunakan untuk model prediksi
- Nilai evaluasi R2 prediksi *profit* meningkat dari 0.30 menjadi 0.38 setelah menambah fitur *Youtube* sebagai prediktor

#### 4.6.1 Data Collection

*Youtube* adalah *platform* berbagi video secara gratis. Setiap pengguna dapat mengunggah video dan mendapatkan jumlah *view* dan *like* yang banyak. *Youtube* dapat menghitung jumlah *view* yang ditonton oleh *user*. Semakin banyak *view* yang diperoleh maka semakin video itu akan direkomendasikan kepada *user* lain.

Sebuah film membutuhkan media untuk mempromosikan film yang dibuat kepada penonton. Biasanya pihak produksi *film* akan menciptakan *trailer* untuk mempromosikan film yang dibuat. *Trailer* adalah cuplikan video tentang promosi film yang dilakukan sebelum film rilis ke bioskop. Pembuat film menciptakan akun *youtube* dan mengunggah *trailer* yang dibuat ke situs *youtube*. Pada proses ini akan diuji apakah jumlah penonton *trailer* film di *youtube* dapat mempengaruhi kesuksesan film.

Selain *youtube*, terdapat situs media sosial lain yang dibuat pembuat film untuk mempromosikan film. *Instagram* adalah sebuah aplikasi berbagi foto dan video untuk para pengguna sosial media. *Instagram* memiliki fitur berupa setiap *user* dapat mengikuti *user* lain (*follow*). Semakin banyak *follower* yang dimiliki, maka akan semakin terkenal sebuah akun. Selain fitur *follow*, *instagram* memiliki fitur *hashtag*. *Hashtag* (#) adalah tagar yang dapat ditambahkan pada sebuah foto yang diunggah sehingga *user* yang tidak saling *follow* tetap dapat berbagi konten.

Biasanya *hashtag* digunakan untuk memberikan makna tema sebuah foto. Pembuat film akan memanfaatkan *instagram* untuk mempromosikan film dengan mengunggah foto poster film dan video *trailer* singkat. Pada proses ini diuji juga apakah sosial media *instagram* memiliki pengaruh pada tingkat kesuksesan film.

## Data Collection pada Youtube

Google menyediakan sebuah media bagi para *developer* untuk mengambil data secara *open source*. Youtube sebagai bagian dari google menyediakan API yang dapat dimanfaatkan untuk mengambil data youtube bernama *YoutubeAPIV3*. Cara mendapatkan data adalah dengan melakukan *request* pada *uniform resource locator (url)* yang disediakan *youtube*. Contoh *url* yang dapat diakses untuk melakukan pencarian data video adalah.

<https://www.googleapis.com/youtube/v3/search?part=snippet&q=<query>&key=<apikey>>

*url* adalah *request* yang dapat dilakukan untuk melakukan pencarian data. Teks "<query>" dan "<apikey>" dapat diubah secara dinamis. *query* adalah kata kunci pencarian video dan *apikey* adalah sebuah *identifier* / penanda *user* mana yang melakukan *request* data. *Apikey* dapat diperoleh dari registrasi *API* dengan menggunakan akun *gmail*. Hasil dari pencarian ini adalah sebuah teks *json*.

```
{
  "kind": "youtube#searchListResponse",
  "etag": "Mnxs_Dh4uTfzeTZ0wgir6cdy1bI",
  "nextPageToken": "CAUQAA",
  "regionCode": "ID",
  "pageInfo": {
    "totalResults": 1000000,
    "resultsPerPage": 5
  },
  "items": [
    {
      "kind": "youtube#searchResult",
      "etag": "F1AHVt5KRtUj0w7GSYKX3v8QEgk",
      "id": {
        "kind": "youtube#video",
        "videoId": "ExeTwQWrcwY"
      },
      "snippet": {
        "publishedAt": "2013-12-03T00:46:23Z",
        "channelId": "UCTCjFFoX1un-i7ni4B6HJ3O",
        "title": "The Dark Knight (2008) Official Trailer #1 - Christopher Nolan Movie HD",
        "description": "The Dark Knight (2008) Official Trailer #1 - Christopher Nolan Movie HD",
        "thumbnails": {
          "medium": {}
        },
        "channelTitle": "Movieclips Classic Trailers",
        "liveBroadcastContent": "none",
        "publishTime": "2013-12-03T00:46:23Z"
      }
    }
  ]
}
```

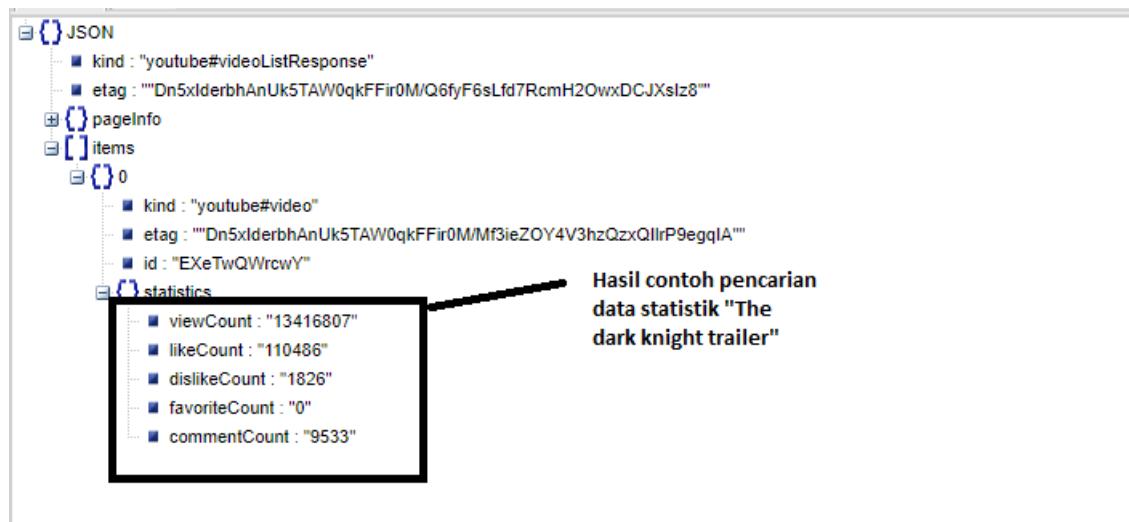
Gambar 4.54: Contoh JSON hasil request search youtube menggunakan API

Id video dari hasil pencarian pada dapat digunakan untuk memanggil *request statistic*. *Request statistic* adalah *request* data video yang berhubungan dengan status video. *Statistics* adalah data yang berhubungan dengan *view*, jumlah *likes* dan *dislikes* sebuah video. *Url* yang dapat diakses untuk mengambil datanya adalah.

[https://www.googleapis.com/youtube/v3/videos?part=statistics&id=<video\\_id>&key=<apikey>](https://www.googleapis.com/youtube/v3/videos?part=statistics&id=<video_id>&key=<apikey>)

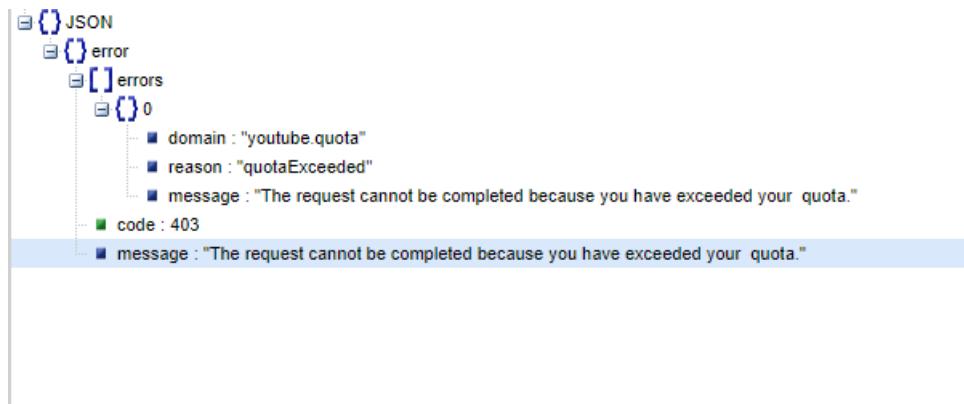
*Url* diatas dapat melakukan untuk mengambil data statistik sebuah video dari *youtube*. *Request* ini menerima parameter berupa *video\_id* dan *apikey*. *Video id* adalah *identifier* dari sebuah video. *Video id* yang sebelumnya diperoleh dari *request* pencarian video dapat dimanfaatkan. Berdasarkan

*video id* yang sudah diperoleh untuk mengambil data "The dark knight trailer" pada Gambar 4.54, berikut adalah hasil pencarian data statistik.



Gambar 4.55: Contoh JSON hasil request seach youtube menggunakan API

Gambar 4.55 adalah contoh hasil data statistik yang diperoleh dari pencarian "The dark knight trailer". Berdasarkan penjelasan API untuk pencarian data *youtube*, pengumpulan data *view* *youtube* dari setiap *trailer film* pada *dataset*.



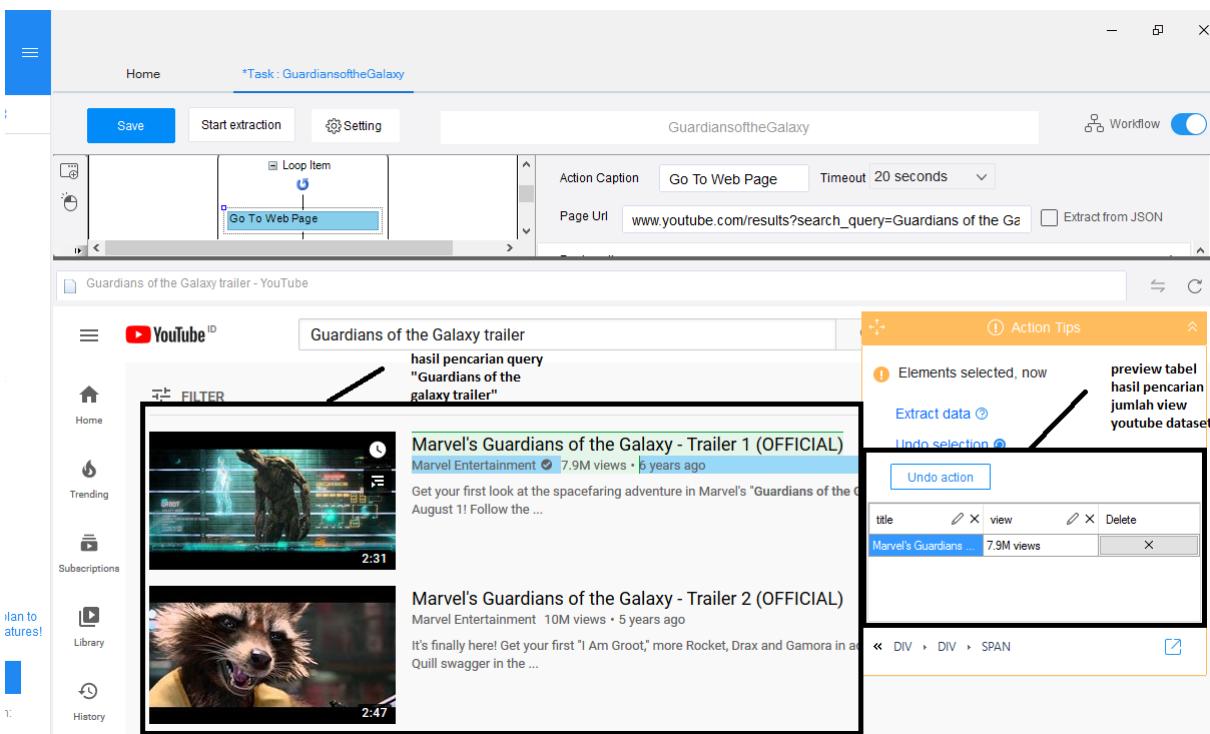
Gambar 4.56: Contoh JSON Error Youtube Quota Exceeded

Pada pencarian jumlah *view trailer* pada *dataset* ke-90, *json* yang dikembalikan adalah *json* pada Gambar 4.56. Hasil *request* tersebut menjelaskan bahwa ternyata terdapat batas dalam melakukan *request* tiap harinya. Maka, pencarian data sosial media menggunakan API tidak dapat dilakukan karena membutuhkan waktu untuk menunggu kuota dapat terisi kembali tiap harinya. Akan dilakukan *scraping* menggunakan *Octoparse* sebagai solusi lain.

Untuk mengakses situs *youtube* menggunakan *browser*, *user* memerlukan untuk memasukkan *url* alamat situs *youtube*. Contoh *url* untuk melakukan pencarian pada film adalah.

```
https://www.youtube.com/results?search_query=<kueripencarian>
```

*Url* diatas dapat dimanfaatkan untuk melakukan pencarian video *youtube*. Bagian "<kueripencarian>" adalah sebuah teks dinamis yang dapat diganti dengan *query* pencarian dari video yang dibutuhkan. Dengan memanfaatkan *url* ini, sehingga *scraping* jumlah *view trailer* video pada *dataset* dapat dilakukan dengan *Octoparse*.



Gambar 4.57: Tampilan octoparse untuk scraping youtube view

Gambar 4.57 adalah tampilan saat *Octoparse* memilih elemen dalam mengambil data. Dengan menggunakan cara ini, akan dilakukan pencarian semua data jumlah *trailer film* pada *dataset* menggunakan *Octoparse*. Berikut adalah 10 sampel *url* yang digenerate dari *dataset*.

Tabel 4.19: 10 sampel url yang digenerate

Url Dataset yang diakses untuk scrapping
www.youtube.com/results?search_query=Prometheus trailer
www.youtube.com/results?search_query=Split trailer
www.youtube.com/results?search_query=Sing trailer
www.youtube.com/results?search_query=Suicide Squad trailer
www.youtube.com/results?search_query=The Great Wall trailer
www.youtube.com/results?search_query=La La Land trailer
www.youtube.com/results?search_query=The Lost City of Z trailer
www.youtube.com/results?search_query=Fantastic Beasts and Where to Find Them trailer
www.youtube.com/results?search_query=Hidden Figures trailer
www.youtube.com/results?search_query=Rogue One trailer

Tabel 4.19 adalah beberapa dari semua *url* yang akan diakses menggunakan *octoparse*. *Octoparse* mengakses semua *url dataset* untuk mengambil data jumlah *view* dari *trailer film*. Dengan memanfaatkan *octoparse* dan semua *url* yang digenerate, berikut adalah contoh 10 sampel hasil pencarian jumlah *view trailer*.

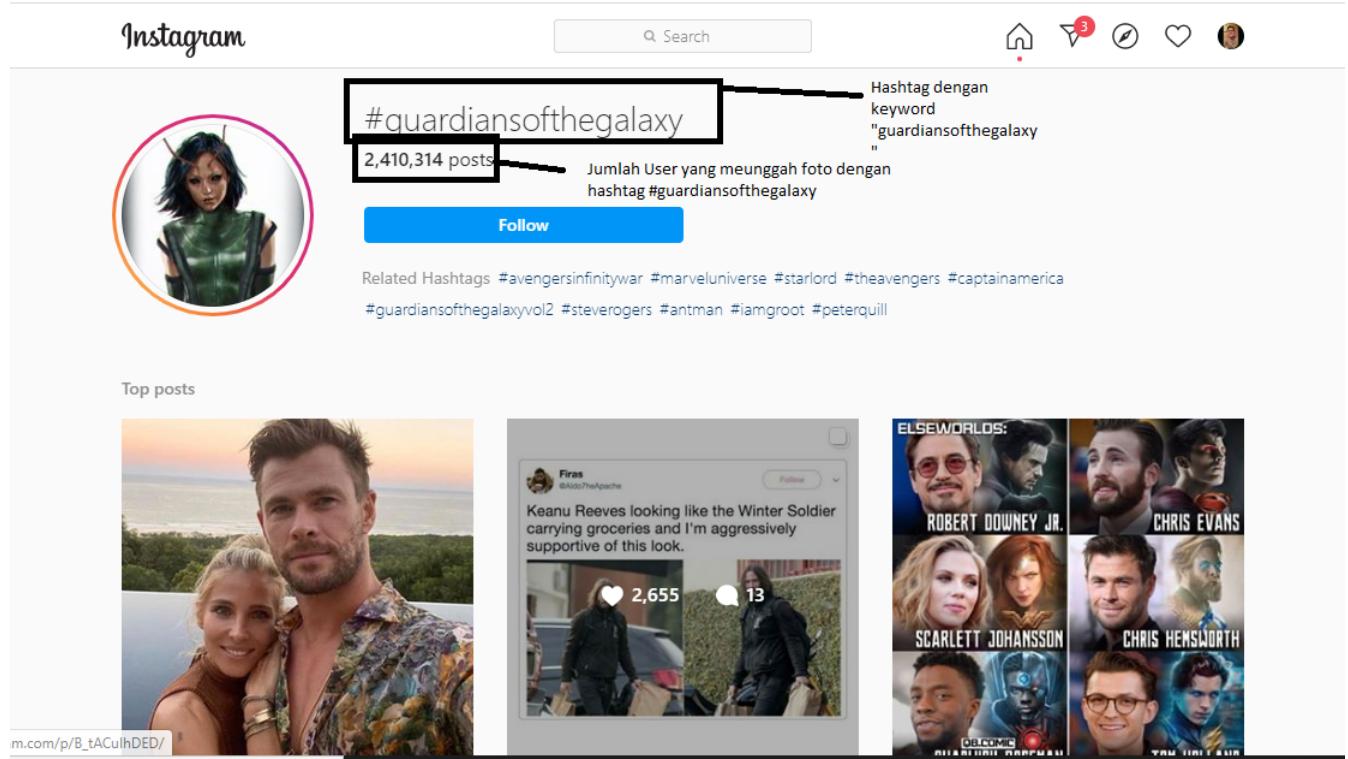
Tabel 4.20: 10 sampel hasil *scrapping youtube view* menggunakan *octoparse*

Youtube title	Views
Prometheus - Official Full Trailer - In Theaters 6/8/12	10M views
Split Official Trailer 1 (2017) - M. Night Shyamalan Movie	27M views
Sing TRAILER 1 (2016) - Scarlett Johansson, Matthew McConaughey	9.1M views
Suicide Squad - Official Trailer 1 [HD]	90M views
The Great Wall - Official Trailer 2 - In Theaters This February	10M views
La La Land (2016 Movie) Official Trailer – 'Dreamers'	37M views
The Lost City of Z International Trailer 1 (2017) Movieclips Trailers	6.3M views
PASSENGERS - Official Trailer (HD)	30M views
Fantastic Beasts and Where to Find Them - Teaser Trailer [HD]	16M views
Hidden Figures Official Trailer [HD] 20th Century FOX	7.5M views

Tabel 4.20 adalah beberapa contoh hasil *scraping* untuk mengambil data *youtube view*. Terdapat komponen lain dalam video *youtube* seperti jumlah *likes* dan *dislikes* tetapi tidak semua video mau untuk ditampilkan jumlah *like* sehingga analisis hanya menggunakan jumlah *view*.

### Data Collection Instagram

Fitur *hashtag Instagram* memungkinkan pengguna yang tidak berhubungan dapat membagikan foto dan video dengan tema yang sama. Tren suatu *keyword hashtag* dapat mempengaruhi popularitas sesuatu hal. Contohnya adalah sebagai berikut.



Gambar 4.58: Contoh fitur Hashtag Instagram

Gambar 4.58 adalah contoh sebuah tren *hashtag* dengan *keyword* sebuah film yaitu "*Guardians of the Galaxy*". Semua unggahan foto dengan *hashtag* "*Guardians of the galaxy*" adalah semua hal yang berhubungan dengan film tersebut seperti aktor, poster dan video unggahan. Pada tahap

ini dilakukan pengumpulan data menggunakan *Octoparse*. *Url instagram* yang diakses dengan *url* tertentu

*https://www.instagram.com/explore/tags/<namahashtag>/*

*url hashtag instagram* dapat dimanfaatkan untuk mengakses sebuah halaman dengan *hashtag* tertentu. Bagian "<namahashtag>" dapat diganti sesuai *hashtag* yang ingin diakses. Dengan menggunakan *url* yang disediakan *instagram*, akan dibuat seluruh *url hashtag* berdasarkan nama film pada *dataset*. Berikut adalah sampel 10 *url instagram hashtag* berdasarkan film *dataset*.

Tabel 4.21: 10 Sampel URL Instagram Hashtag dengan Menggunakan Judul Film

url Hashtag Instagram
<i>https://www.instagram.com/explore/tags/TheComedian/</i>
<i>https://www.instagram.com/explore/tags/SpringBreakers/</i>
<i>https://www.instagram.com/explore/tags/Hairspray/</i>
<i>https://www.instagram.com/explore/tags/TheHappening/</i>
<i>https://www.instagram.com/explore/tags/TheAssassinationofJesseJamesbytheCowardRobertFord/</i>
<i>https://www.instagram.com/explore/tags/Fridaythe13th/</i>
<i>https://www.instagram.com/explore/tags/KickAss/</i>
<i>https://www.instagram.com/explore/tags/SuckerPunch/</i>
<i>https://www.instagram.com/explore/tags/RocknRolla/</i>
<i>https://www.instagram.com/explore/tags/BlueValentine/</i>

Tabel 4.21 adalah beberapa *url* yang diakses untuk mengambil jumlah *hashtag* dari *keyword* judul film dari *dataset*. Semua *url* yang berjumlah 838 *url hashtag* film akan diakses oleh *Octoparse*. Berikut adalah sampel 10 hasil *scraping* yang peroleh *Octoparse*.

Tabel 4.22: 10 Sampel hasil *scraping* data hashtag instagram menggunakan *octoparse*

Hashtag Judul Film	Jumlah Post
# thecomedian	27,932
# springbreakers	268,282
# hairspray	708,045
# thehappening	11,619
# theassassination	1,940
# fridays13th	2,497,489
# kickass	1,228,185
# suckerpunch	109,806
# rocknrolla	137,870
# bluevalentine	47,173

Tabel 4.22 adalah beberapa contoh hasil *scraping* untuk mengambil jumlah *hashtag* dari setiap judul film pada *dataset*. Selain pengumpulan jumlah data *hashtag* judul film, akan dilakukan pengumpulan data *hashtag* dengan *keyword* nama aktor/aktris utama pada sebuah film. Aktor/aktris umumnya sudah memiliki situs sosial media resmi untuk berhubungan dengan penggemarnya. Biasanya aktor akan bekerja sama dengan pembuat film untuk mempromosikan filmnya melalui sosial media. Dengan cara yang sama seperti melakukan *scraping hashtag* judul film, akan dilakukan pengumpulan data *hashtag* nama aktor. Berikut adalah 10 sampel film beserta *hashtag* aktor yang diakses.

Tabel 4.23: 10 Sampel hasil pengumpulan data *hashtag* aktor utama setiap film pada dataset

url hashtag aktor utama
<a href="https://www.instagram.com/explore/tags/robertdeniro/">https://www.instagram.com/explore/tags/robertdeniro/</a>
<a href="https://www.instagram.com/explore/tags/vanessahudgens/">https://www.instagram.com/explore/tags/vanessahudgens/</a>
<a href="https://www.instagram.com/explore/tags/johntravolta/">https://www.instagram.com/explore/tags/johntravolta/</a>
<a href="https://www.instagram.com/explore/tags/markwahlberg/">https://www.instagram.com/explore/tags/markwahlberg/</a>
<a href="https://www.instagram.com/explore/tags/bradpitt/">https://www.instagram.com/explore/tags/bradpitt/</a>
<a href="https://www.instagram.com/explore/tags/jaredpadalecki/">https://www.instagram.com/explore/tags/jaredpadalecki/</a>
<a href="https://www.instagram.com/explore/tags/aarontaylorjohnson/">https://www.instagram.com/explore/tags/aarontaylorjohnson/</a>
<a href="https://www.instagram.com/explore/tags/emilybrowning/">https://www.instagram.com/explore/tags/emilybrowning/</a>
<a href="https://www.instagram.com/explore/tags/gerardbutler/">https://www.instagram.com/explore/tags/gerardbutler/</a>
<a href="https://www.instagram.com/explore/tags/ryangosling/">https://www.instagram.com/explore/tags/ryangosling/</a>

Tabel 4.23 adalah beberapa contoh dari semua *url* yang *digenerate* untuk mengakses semua *hashtag* aktor utama film pada *dataset*. Setelah melakukan *scraping* dengan *Octoparse*, berikut adalah 10 sampel jumlah *hashtag actor*.

Tabel 4.24: 10 Sampel hasil *scraping* data *hashtag* aktor dengan *Octoparse*

Hashtag Actor	Jumlah Post
# robertdeniro	424,802
# vanessahudgens	478,292
# johntravolta	241,526
# markwahlberg	197,163
# bradpitt	1,055,192
# jaredpadalecki	4,068,673
# aarontaylorjohnson	93,604
# emilybrowning	22,936
# gerardbutler	143,321
# ryangosling	774,215

Tabel 4.24 adalah beberapa sampel hasil *scraping* data *hashtag* aktor yang diperoleh. Data yang diperoleh merupakan jumlah unggahan foto dari semua *user* yang menggunakan *hashtag* tersebut.

#### 4.6.2 Data Integration

Terdapat 3 fitur sosial media yang sudah diperoleh selama tahap *data collection* yaitu :

- Jumlah *view count trailer* (*Youtube*)
- Jumlah *hashtag count title* (*Instagram*)
- Jumlah *hashtag count actor name* (*Instagram*)

Data sosial media yang sudah diperoleh sebelumnya digunakan untuk digabungkan dengan *dataset*. Cara penggabungan adalah dengan memanfaatkan judul film pada *dataset* dan nama aktor utama. Berikut adalah contoh beberapa *dataset* yang telah tergabung dengan data sosial media.

Tabel 4.25: Sampel data integrasi antara *dataset* dengan data sosial media

Title	...	Youtube Trailer View	Hashtag title	Hashtag Actor
Kickboxer: Vengeance	...	4.5M Views	3,122	65,443
Spider-Man 3	...	4.5M Views	138,212	205,845
Hercules	...	1.9M Views	1,0737,96	645,185
The Amazing Spider-Man	...	14M Views	414,847	389,582
In the Heart of the Sea	...	12M Views	137,475	1,457,829
The Lost City of Z	...	6.3M Views	94,449	355,403
Turbo Kid	...	668K Views	17,222	20,625
Fool's Gold	...	21M Views	150,990	172,612
Monster Trucks	...	4.4M Views	370,426	23,660
Men in Black 3	...	17M Views	9,118	1,391,216

Tabel 4.25 menunjukkan bahwa *dataset* sudah digabungkan dengan data sosial media.

#### 4.6.3 Data Transformation

Data sosial media yang sudah digabung dengan *dataset* akan dianalisis menggunakan visualisasi, tetapi terdapat beberapa masalah yaitu :

- Data *youtube view* masih berbentuk *String* (contoh: "1.2 M Views")
- Data *youtube view* masih memiliki variasi data yang berbeda (Contoh : "1.2M" untuk satuan jutaan dan "333K" untuk satuan ribuan)
- Data *Instagram Hashtag* masih berbentuk *String* ("120,000")

Berdasarkan permasalahan data sosial media yang dijelaskan sebelumnya, maka dilakukan *data transformation* untuk mengubah data media sosial menjadi bentuk yang lebih sesuai. Berikut adalah tabel konversi 3 kolom sosial media pada *dataset*.

Tabel 4.26: Operasi dan contoh data *transformation* pada data media sosial

Nama Kolom	Operasi	Sebelum Konversi	Sesudah Konversi
Youtube View	Menghilangkan keyword "M,K"	1.4M Views	1.4
	Mengubah dalam bentuk satuan jutaan	300K Views	0.3
Hashtag Title	Mengubah tipe data menjadi numerik	"120,000"	120000
Hashtag Actor	Mengubah tipe data menjadi numerik	"10,000"	10000

Tabel 4.26 adalah detail operasi yang dilakukan untuk mengonversi data sosial media menjadi bentuk yang lebih tepat agar dapat dianalisis secara deskriptif. Perubahan fitur media sosial menjadi numerik agar fitur media sosial dapat dianalisis secara visualisasi. Kolom *youtube view* dibuat menjadi satuan jutaan. Kolom *hashtag Instagram* dirubah dari *String* menjadi numerik. Berikut adalah contoh 5 sampel pada *dataset*.

Tabel 4.27: 5 Sampel *dataset* setelah konversi *data transformation*

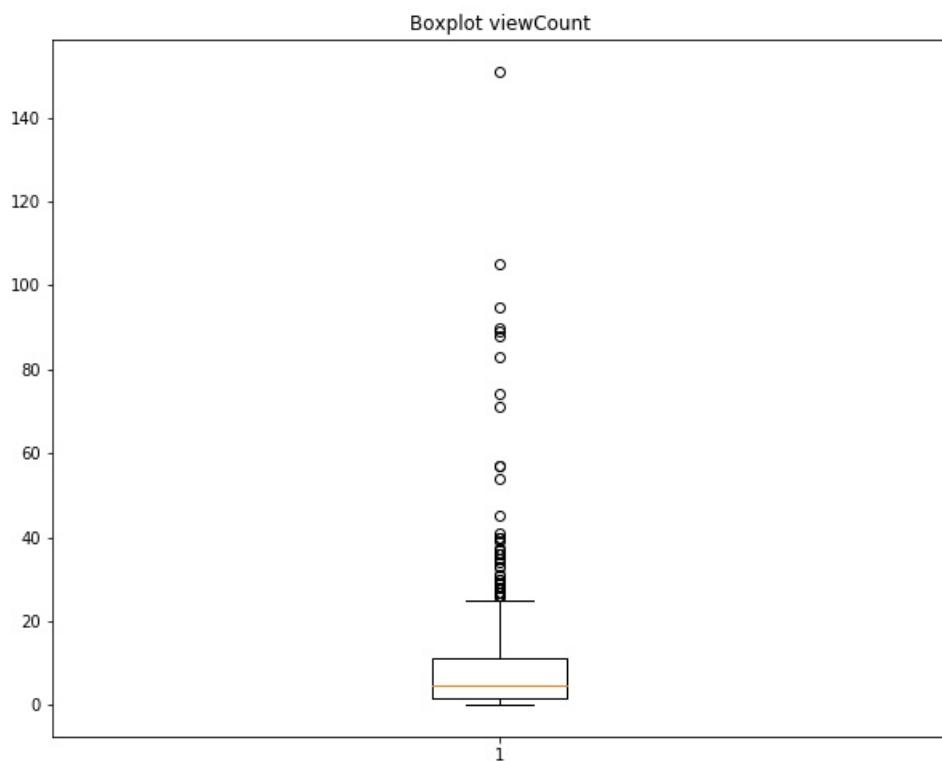
Title	Youtube_view (jutaan)	Hashtag Title	Hashtag Actor
Deadpool	19	5622065	585034
Ant-Man	25	1534586	276859
Sherlock Holmes	2	1948636	2469813
Hands of Stone	0.297	27155	393655
RED	0.962	115604667	280856

Tabel 4.27 adalah contoh beberapa hasil *dataset* yang sudah *ditransform* menjadi bentuk yang lebih sesuai.

#### 4.6.4 Analisis Data Media Sosial Menggunakan Visualisasi

Berdasarkan data sosial media yang sudah diperoleh pada subbab sebelumnya, akan dilakukan analisis data menggunakan visualisasi. Berikut adalah visualisasi distribusi fitur media sosial menggunakan *boxplot*.

##### Analisis Visualisasi Youtube View Count



Gambar 4.59: Boxplot view count

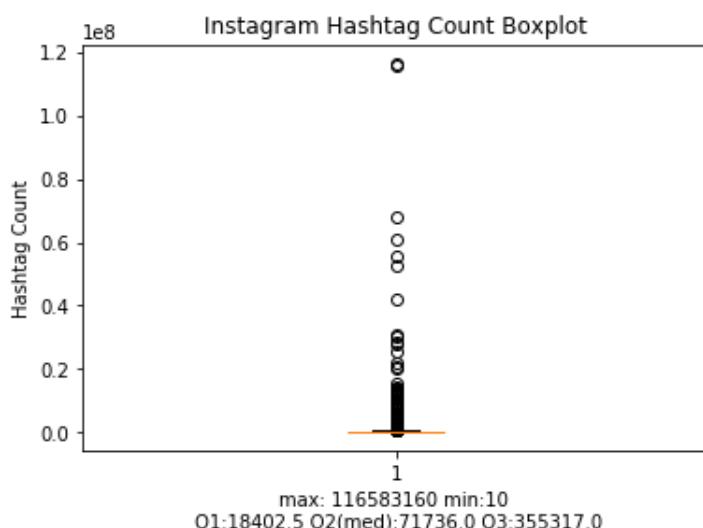
Gambar 4.59 adalah visualisasi distribusi menggunakan *boxplot*. Tiap film memiliki jumlah banyak pengguna yang menonton *trailer* film tersebut di situs *youtube*. Berdasarkan visualisasi terdapat film-film divisualisasikan sebagai *outlier* karena terlalu jauh dari nilai Q2. Berikut adalah 5 film *outlierview count* tertinggi.

Tabel 4.28: 5 Film *outlier* dengan *View Count* Terbanyak

Title	Genre	Revenue (Million)	ROI (%)	View
Star Wars : The Force Awakens	Action,Adventure,Fantasy	936.63	282.298	105
Fifty Shades of Grey	Drama,Romance	166.15	315.375	95
Suicide Squad	Action, Adventure,Fantasy	325.02	85.72	90
Jurrasic World	Action,Adventure,Sci-fi	652.18	334.7867	89
Avengers: Age of Ultron	Action,Adventure,Sci-Fi	408.08	83.596	88

Tabel 4.28 adalah 5 film *outlier* dengan jumlah *view trailer youtube* yang tertinggi. 5 film terbaik berdasarkan *view count* cenderung merupakan film *genre Action* dan *Adventure*. Film terbaik berdasarkan *view* juga meraih *revenue* dan *ROI* yang menguntungkan.

#### Analisis Visualisasi Instagram Hashtag Title



Gambar 4.60: Hashtag Instagram title boxplot

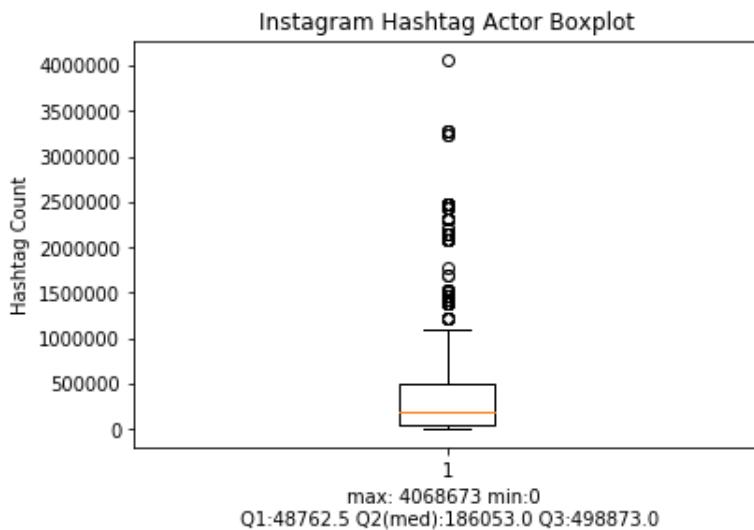
Gambar 4.60 adalah *boxplot* dari *hashtag title Instagram*. Berdasarkan *boxplot* yang diberikan, terdapat film yang memiliki *hashtag* yang jumlahnya sangat tinggi. Berikut adalah 5 film *outlier* dengan *hashtag title* terbanyak.

Tabel 4.29: 5 Film *outlier* berdasarkan Hashtag judul terbanyak

Title	Hashtag Title Instagram	Revenue	ROI
Vacation	116,583,160	58.88	282.533
RED	115,604,667	90.36	55.79
Sunshine	67,897,892	3.68	-29.86
Australia	61,052,897	49.55	-80.45
Gold	55,697,911	7.22	-12.78

Tabel 4.29 adalah 5 film dengan *hashtag* terbanyak berdasarkan judul. Film yang memiliki *hashtag* judul banyak cenderung tidak menunjukkan pola yang menarik. Hal ini disebabkan karena judul film merupakan *term* / kata yang sering muncul dan bukan judul film khusus.

### Analisis Visualisasi Hashtag Actor Instagram



Gambar 4.61: Hashtag Instagram actor boxplot

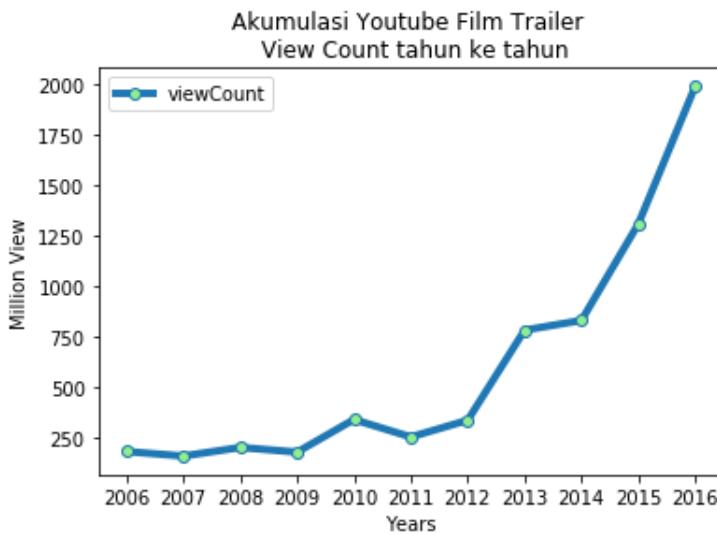
Gambar 4.61 adalah *boxplot* untuk distribusi *hashtag actor* utama dalam setiap film pada *dataset*. *Hashtag actor* menunjukkan popularitas seorang aktor. Berikut adalah 5 film yang aktor utama memiliki *hashtag* terbanyak.

Tabel 4.30: 5 Film *outlier* berdasarkan *Actor Hashtag* terbanyak

Title	Aktor	Jumlah Hashtag	Revenue	ROI
Friday the 13th	Jared Padalecki	4.068.673	65	242,10
The Maze Runner	Dylan O'Brien	3.271.395	102,41	201,20
Fifty Shades of Grey	Dakota Johnson	3.235.549	166,15	315,38
Iron Man	Robert Downey	2.469.813	318,3	127,36
Captain America : Civil War	Chris Evans	2.444.525	408,08	63.232

Tabel 4.30 adalah 5 film *outlier* berdasarkan popularitas aktor di *Instagram* berdasarkan *jumlah hashtag*. Aktor yang populer di *instagram* cenderung film yang dimainkan menghasilkan keuntungan.

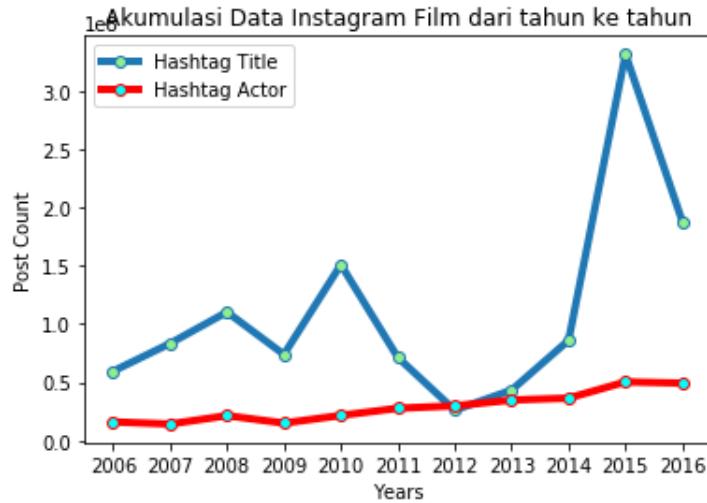
### Tren Youtube View Trailer Dari Tahun Ke Tahun



Gambar 4.62: Analisis tren *youtube view trailer*

Gambar 4.62 adalah visualisasi tren jumlah *view trailer youtube* setiap film pada dataset dari tahun ke tahun. Berdasarkan visualisasi, akumulasi *view* dari tahun ke tahun meningkat. Peminat situs *youtube* untuk menonton *trailer* film yang mereka tunggu-tunggu semakin meningkat. Pihak pembuat film dapat memanfaatkan media sosial untuk dapat mempromosikan film.

### Tren Instagram Hashtag Dari Tahun Ke Tahun

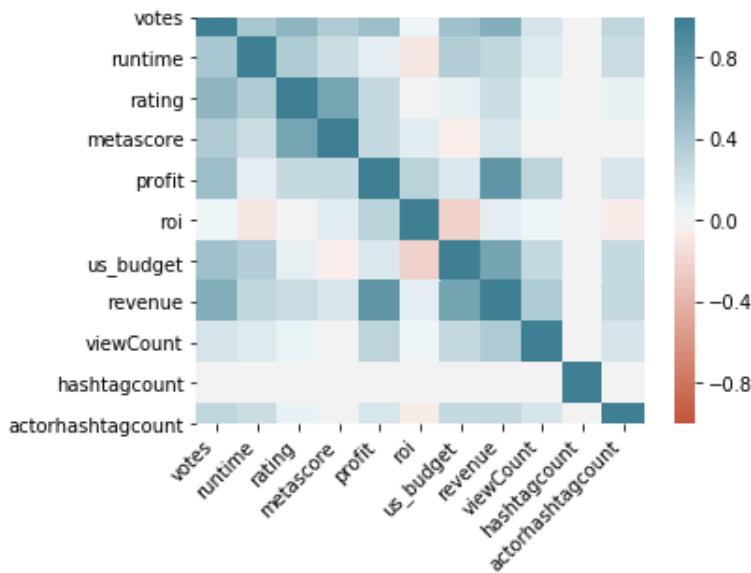


Gambar 4.63: Analisis tren *Instagram Hashtag*

Gambar 4.63 adalah visualisasi tren *instagram hashtag* dari tahun ke tahun. Jumlah akumulasi unggahan foto/video tiap tahunnya meningkat. Meningkatnya tiap tahun dapat dimanfaatkan pihak pembuat film untuk fokus mempromosikan filmnya di *instagram*.

#### 4.6.5 Data Selection

Visualisasi korelasi semua fitur satu sama lain. Visualisasi ditampilkan menggunakan *heatmap*.



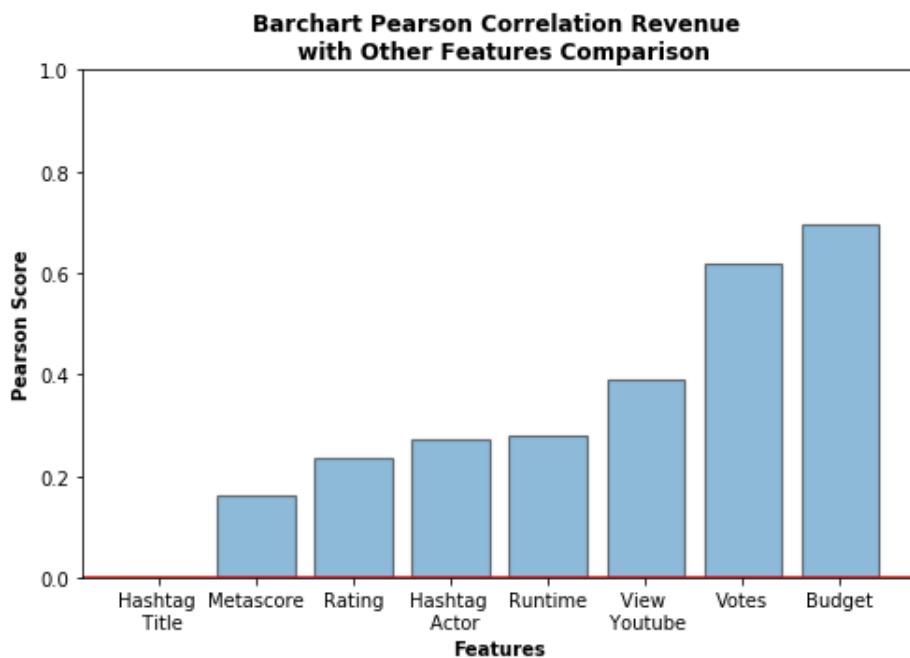
Gambar 4.64: Pearson correlation fitur prediktor terhadap *revenue*

Gambar 4.64 adalah visualisasi korelasi tiap fitur. Warna biru menandakan korelasi positif dan merah menandakan korelasi negatif. Semakin gelap warnanya menandakan korelasi semakin kuat. Berdasarkan hasil visualisasi *heatmap*, informasi yang didapatkan adalah :

- Ada kecenderungan jika nilai *metascore* yang diperoleh sebuah film tinggi, maka semakin tinggi *rating* yang diperoleh
- Jumlah *post* pada *hashtag instagram* yang menggunakan judul film (*hashtagcount* tidak memiliki pengaruh terhadap fitur lain)
- Semakin banyak *budget* yang dikeluarkan, maka ada kemungkinan sebuah film dapat dibuat dengan durasi lebih lama. Hal ini disebabkan oleh proses penyuntingan film yang lebih lama dan membutuhkan biaya lebih
- Semakin banyak *budget* yang dikeluarkan, maka akan kemungkinan *votes* sebuah film meningkat
- Ada kecenderungan semakin banyak *budget* yang dikeluarkan, maka semakin meningkat jumlah penonton *trailer film* di *youtube*. Pembuatan *trailer* yang menarik dapat meningkatkan minat penonton
- Semakin banyak jumlah penonton *trailer film* di *youtube* (*viewCount*), maka semakin meningkat *revenue* dan *profit* yang dapat diperoleh. *Trailer* sebuah film yang menarik dapat membuat penonton tertarik untuk menonton di bioskop
- Semakin banyak *votes* yang diperoleh, maka semakin banyak *revenue* dan *profit* yang diperoleh
- Tidak terdapat fitur yang memiliki korelasi yang kuat terhadap *ROI*
- Ada kecenderungan semakin banyak jumlah *post* dengan menggunakan *hashtag* nama aktor utama film, maka semakin meningkat keuntungan sebuah film. Pembuat film dapat bekerja-sama menggunakan akun resmi sosial media para aktor untuk membantu mempromosikan film

Pada tahap ini dilakukan pengujian korelasi semua fitur prediktor media sosial dengan target kesuksesan yaitu *revenue*, *profit* dan *ROI* menggunakan *pearson correlation*. Berikut adalah pengujian *pearson* fitur prediktor terhadap *revenue* menggunakan visualisasi.

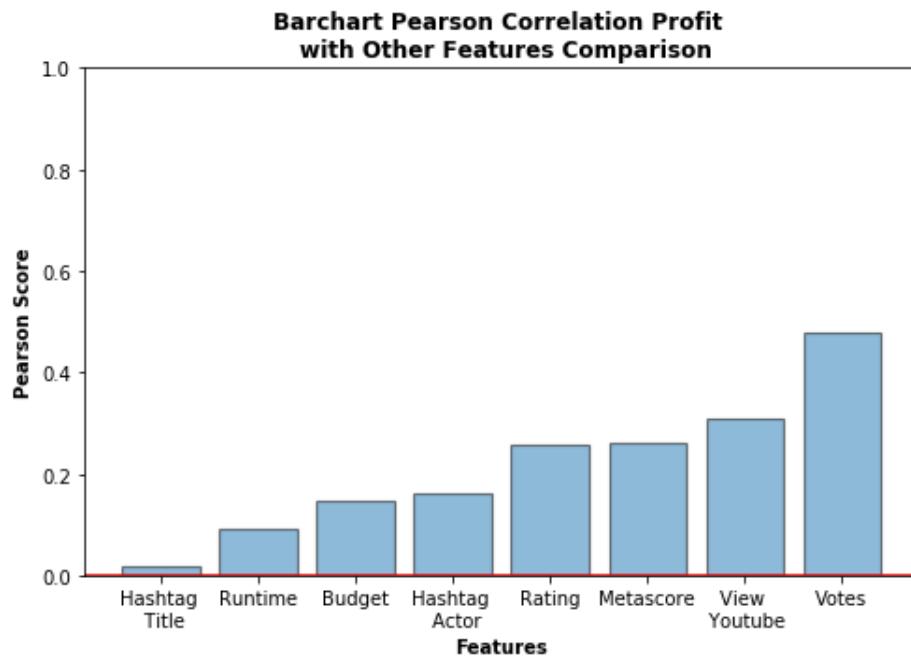
### Pearson Correlation Revenue Dengan Fitur Lain



Gambar 4.65: Pearson correlation fitur prediktor terhadap *revenue*

Gambar 4.65 adalah hasil visualisasi perbandingan perhitungan nilai korelasi *pearson correlation* tiap fitur terhadap *revenue* menggunakan *barchart*. Berdasarkan visualisasi yang ditunjukkan, *budget* / pengeluaran memiliki nilai korelasi yang paling tinggi. Data media sosial seperti *view trailer youtube* juga menjadi salah satu fitur yang mempengaruhi kenaikan *revenue* dibanding fitur lain. *Hashtag* judul film pada *Instagram* dinilai tidak memiliki korelasi dengan *revenue*

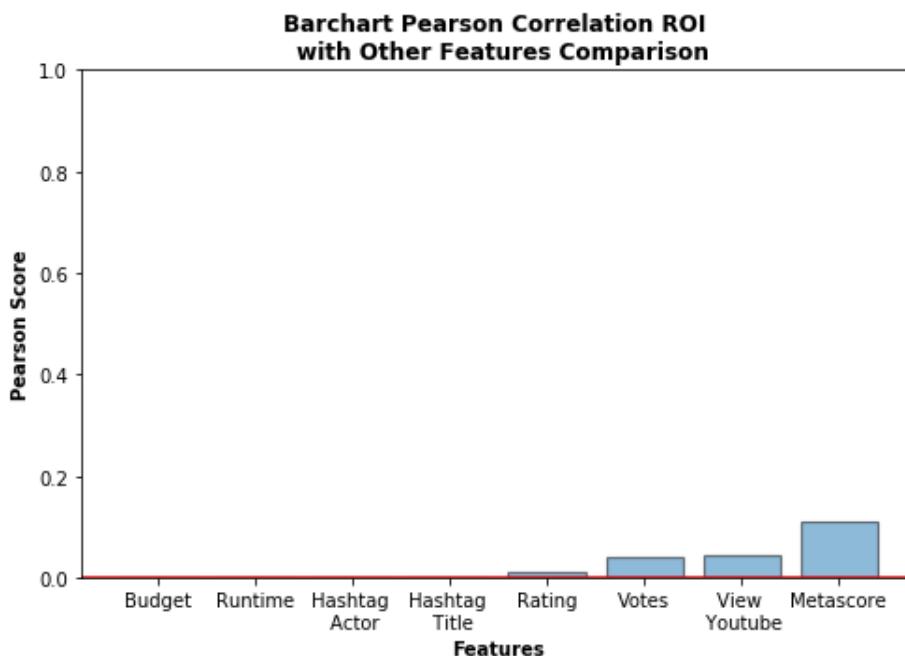
### Pearson Correlation Profit Dengan Fitur Lain



Gambar 4.66: Pearson correlation fitur prediktor terhadap *profit*

Gambar 4.66 adalah hasil visualisasi perbandingan nilai korelasi *pearson correlation* tiap fitur dan *profit* menggunakan *barchart*. Nilai korelasi yang paling tinggi adalah *votes* dan *view count*. Data media sosial yaitu *trailer youtube (viewCount)* menjadi salah satu prediktor yang memiliki korelasi yang kuat dengan *profit*. Data media sosial ternyata dapat dipertimbangkan menjadi fitur yang berpengaruh terhadap *revenue* dan *profit*

### Pearson Correlation ROI Dengan Fitur Lain



Gambar 4.67: Pearson correlation fitur prediktor terhadap *profit*

Gambar 4.67 adalah hasil visualisasi perbandingan nilai korelasi tiap fitur dan *ROI*. Berdasarkan hasil visualisasi, tidak terdapat fitur yang memiliki korelasi yang kuat dengan *ROI*. Nilai tertinggi hanya *metascore* dengan nilai *pearson* yaitu 0.10. Nilai *pearson metascore* dan *ROI* tidak dapat diklasifikasikan sebagai korelasi yang kuat. Terdapat beberapa fitur yang memiliki nilai korelasi negatif dengan *ROI*. Data media sosial tidak memiliki pengaruh terhadap *ROI*.

#### 4.6.6 Percobaan Prediksi Fitur Data Sosial Media

Pada subbab ini dilakukan percobaan membuat model regresi dengan memanfaatkan fitur media sosial *viewCount*. Model regresi dibuat agar dapat memprediksi keuntungan film yaitu *revenue* dan *profit*.

#### Prediksi Revenue Menggunakan Linear Regression Dengan Data Media Sosial

*Dataset* dibagi menjadi 2 bagian *training* dan *test*. *Dataset* dibagi menjadi 80 persen dan 20 persen *test*. Data *training* digunakan untuk menghasilkan fungsi regresi dari prediksi. Dengan menambah *viewCount* sebagai fitur tambahan, fungsi regresi berdasarkan data *training* yang dihasilkan *linear regression* untuk memprediksi *revenue* adalah

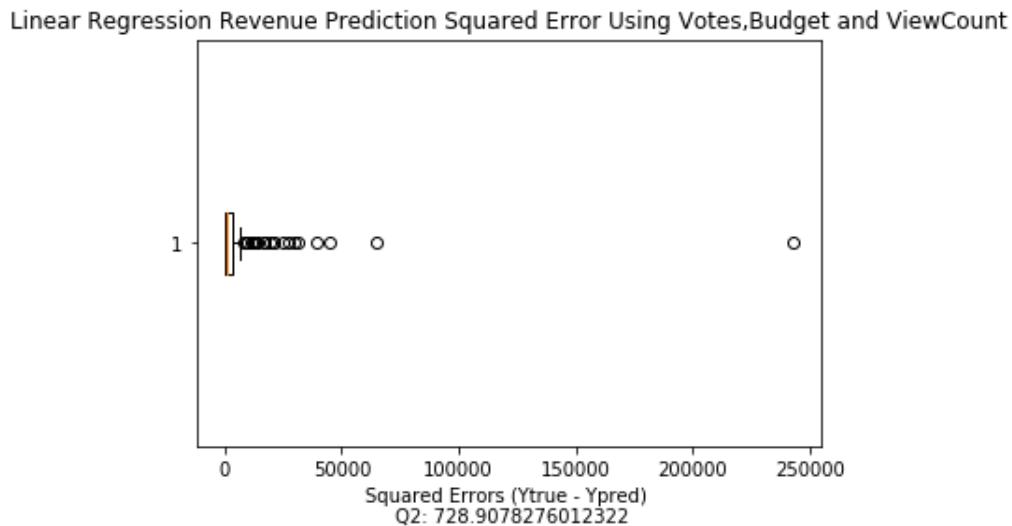
$$\text{revenueLinearRegression} = -11.67 + 0.00019(\text{votes}) + 0.82 * (\text{budget}) + 1.19(\text{viewCount}) \quad (4.5)$$

Persamaan 4.5 adalah fungsi yang dihasilkan untuk melakukan percobaan prdiksi pendapatan kotor / *revenue*. Fungsi ini memanfaatkan 3 fitur yaitu *votes*, *budget* dan *viewCount*. Berikut adalah 5 data film yang diambil dari data *test* dan hasil prediksinya menggunakan fungsi *Linear Regression*.

Tabel 4.31: Tabel percobaan *Linear Regression* untuk prediksi *revenue* menggunakan fitur *votes,budget* dan *viewCount*

Title	Votes	Budget	ViewCount	Revenue Asli	Revenue Prediksi
Sherlock Holmes	357436	123	1.4	186.83	162.234
Victor Frankenstein	37975	65	9.8	5.77	60.89
Pirates of the Caribbean	498821	300	1.8	309.4	334.156
Final Destination 5	88000	40	0.07	42.58	38.40
Grown Ups 2	114482	80	3.8	133.67	80.91

Tabel 4.31 dapat digunakan untuk melihat perbandingan *revenue* asli dan hasil prediksi yang dihasilkan oleh *Linear regression* menggunakan fitur *votes,budget* dan *viewCount*. Berikut adalah distribusi nilai *squared error* dari percobaan semua prediksi data *test*.



Gambar 4.68: Distribusi nilai *squared errors* ( $(y_{\text{true}} - y_{\text{test}})^2$ ) *Linear regression* pada prediksi *revenue* menggunakan fitur *votes,budget* dan *viewCount*

Gambar 4.68 menunjukkan persebaran nilai *squared error* memiliki nilai yang tidak terlalu besar. Nilai Skor akurasi R2 yang diperoleh adalah.

Tabel 4.32: Hasil nilai R2 *linear regression* pada prediksi revenue menggunakan data sosial media

R2 prediksi Revenue Linear Regression data sosial media
0.63

Tabel 4.32 adalah hasil skor akurasi prediksi *revenue* menggunakan algoritma *linear regression*. Terdapat peningkatan skor R2 setelah menambah fitur media sosial yaitu *viewCount*.

### Prediksi Revenue Menggunakan Polynomial Regression Dengan Data Media Sosial

Percobaan prediksi *revenue* menggunakan fitur sosial media juga diuji dengan *polynomial regression*. Fungsi prediksi yang dihasilkan adalah.

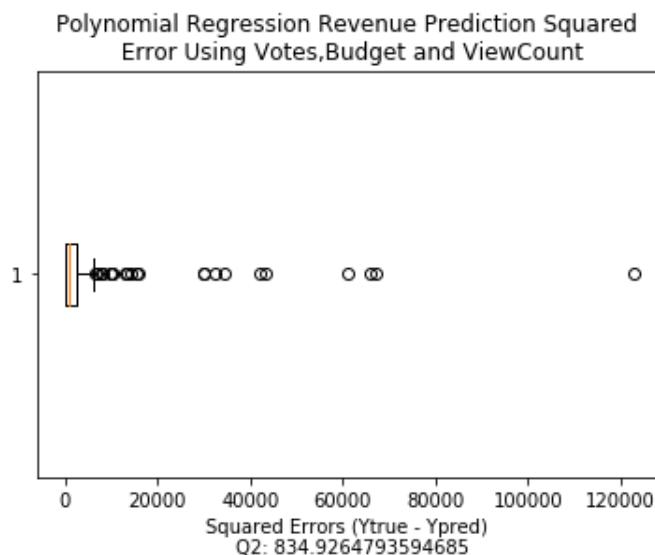
$$\begin{aligned} \text{revenuePolynomialRegression} = & 2.88 + 0.00015 * (\text{votes}) + 0.58 * (\text{budget}) + 0.81 * (\text{viewCount}) + \\ & 0.00000000002 * (\text{votes})^2 + 0.0000023(\text{votes})(\text{budget}) + 0.0000012(\text{votes})(\text{viewCount}) + \\ & 0.0023(\text{budget})^2 + 0.0021(\text{budget})(\text{viewCount}) + 0.0037(\text{viewCount})^2 \end{aligned} \quad (4.6)$$

Persamaan 4.6 adalah fungsi yang dihasilkan untuk melakukan percobaan prediksi pendapatan kotor / *revenue* menggunakan *polynomial regression*. Berikut adalah 5 data film yang diambil dari data *test* dan hasil prediksinya menggunakan fungsi *polynomial regression*

Tabel 4.33: Tabel percobaan *Polynomial Regression* untuk prediksi *revenue* menggunakan fitur *votes*, *budget* dan *viewCount*

Title	Votes	Budget	ViewCount	Revenue Asli	Revenue Prediksi
Sherlock Holmes	357436	123	1.4	186.83	176.784
Victor Frankenstein	37975	65	9.8	5.77	52.077
Pirates of the Caribbean	498821	300	1.8	309.4	356.95
Final Destination	88000	40	0.07	42.58	42.84
Grown Ups 2	11482	80	3.8	133.67	75.87

Tabel 4.33 menampilkan percobaan prediksi pada 5 data *test*. Terdapat prediksi yang sangat akurat dan ada yang perbandingan *revenue* asli dan prediksi yang memiliki selisih yang jauh. Berikut adalah distribusi nilai *squared error* dari percobaan semua prediksi data *test*.



Gambar 4.69: Distribusi nilai *squared errors* ( $(y_{\text{true}} - y_{\text{pred}})^2$ ) *Polynomial regression* pada prediksi *revenue* menggunakan fitur *votes*, *budget* dan *viewCount*

Gambar 4.69 adalah distribusi nilai *squared error* yang diperoleh dari percobaan prediksi *revenue polynomial regression* menggunakan fitur data sosial media. Berdasarkan nilai Q2, percobaan pada *linear regression* memiliki *squared error* yang lebih kecil. Nilai akurasi R2 yang diperoleh adalah.

Tabel 4.34: Hasil nilai R2 *polynomial regression* pada prediksi revenue menggunakan data sosial media

R2 prediksi Revenue Polynomial Regression data sosial media
0.65

Tabel 4.34 adalah hasil skor akurasi prediksi *revenue* menggunakan algoritma *polynomial regression*. Hasil R2 yang diperoleh meningkat lebih tinggi dibanding *linear regression* dan prediksi pada Data Utama pada iterasi sebelumnya. Data sosial media terbukti memiliki pengaruh dalam kesuksesan film

### Prediksi Profit Menggunakan Linear Regression Dengan Data Media Sosial

Selain *revenue*, dilakukan pengujian prediksi *profit* menggunakan *linear regression* dan *polynomial regression*. Fitur prediktor yang digunakan adalah 3 fitur terbaik yaitu *votes*, *budget* dan *viewCount*. Fungsi prediksi yang dihasilkan *linear regression* untuk memprediksi *profit* adalah.

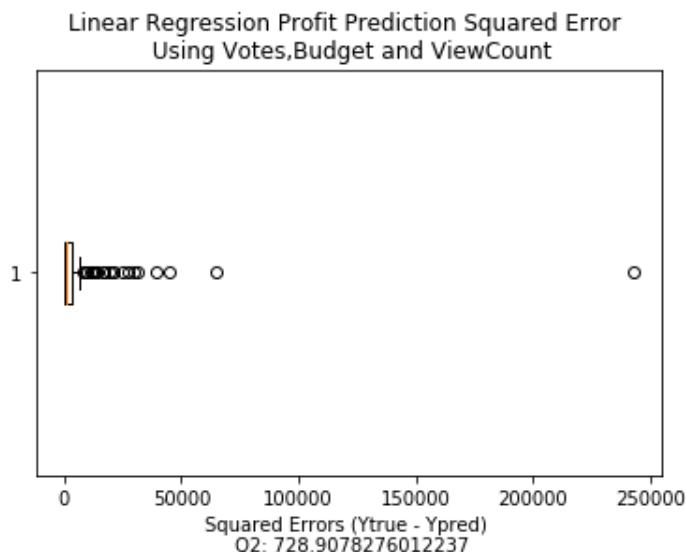
$$\text{profitLinearRegression} = -11.67 + 0.00019(\text{votes}) + (-0.17)(\text{budget}) + 1.19 * (\text{viewCount}) \quad (4.7)$$

Persamaan 4.7 adalah fungsi yang dihasilkan untuk menghitung prediksi *profit*. Fitur yang digunakan adalah *votes*, *budget* dan *viewCount*. Berikut adalah 5 data film yang diambil dari data *test* dan perbandingan prediksinya menggunakan *Linear Regression*.

Tabel 4.35: Tabel percobaan *Linear Regression* untuk prediksi *profit* menggunakan fitur *votes*, *budget* dan *viewCount*

Title	Votes	Budget	viewCount	Profit Asli	Profit Prediksi
Sherlock Holmes	357436	125	1.4	61.83	37.23
Victor Frankenstein	37975	65	9.8	-59.23	-4.10
Pirates of the Caribbean	498821	300	1.8	9.4	34.15
Final Destination	88000	40	0.07	2.58	-1.59
Grown Ups 2	114482	80	3.8	53.67	0.91

Tabel 4.35 merupakan data *test* dan hasil perbandingan *profit* asli dan prediksi menggunakan *linear regression*. Hasil prediksi yang diperoleh masih kurang baik dibanding saat memprediksi *revenue*. Berikut adalah distribusi *squared error* yang diperoleh pada Gambar 4.70.



Gambar 4.70: Distribusi nilai *squared errors* ( $(y_{pred} - y_{test})^2$ ) *Linear regression* pada prediksi *profit* menggunakan fitur *votes*, *budget* dan *viewCount*

Berikut adalah nilai R<sup>2</sup> dari prediksi *profit* menggunakan *Linear Regression*.

Tabel 4.36: Hasil nilai R<sup>2</sup> *linear regression* pada prediksi *profit* menggunakan data sosial media

R<sup>2</sup> prediksi Profit Linear Regression data sosial media  
0.34

Nilai yang diperoleh 4.36 masih cukup kecil untuk disebut sebagai prediksi yang akurat karena masih lebih kecil dari 0.7.

## Prediksi Profit Menggunakan Polynomial Regression Dengan Data Media Sosial

Percobaan prediksi *profit* juga diuji dengan *polynomial regression*. Fungsi prediksi yang dihasilkan adalah,

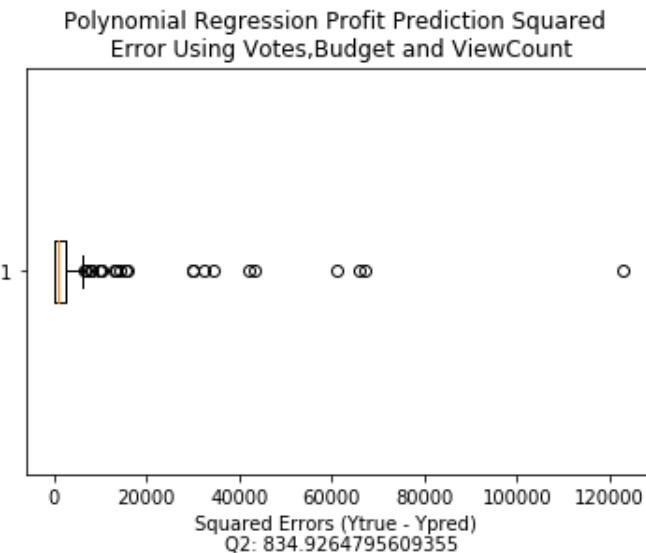
$$\begin{aligned}
& profitPolynomialRegression = 2.88 + 0.00015(votes) + \\
& (-0.41)(budget) + 0.81(viewCount) + (-0.00000000021)(votes)^2 + \\
& 0.0000024(votes)(budget) + 0.0000012(votes)(viewCount) + (-0.0023)(budget)^2 + \\
& 0.0021(budget)(viewCount) + (-0.0037)(viewCount)^2
\end{aligned} \tag{4.8}$$

Persamaan 4.8 adalah fungsi yang dihasilkan untuk melakukan percobaan prediksi *profit* menggunakan *polynomial regression*. Berikut adalah percobaan 5 data *test* menggunakan *Polynomial Regression*

Tabel 4.37: Tabel percobaan *Polynomial Regression* untuk prediksi *profit* menggunakan fitur *votes,budget* dan *viewCount*

Title	Votes	Budget	View Count	Profit Asli	Profit Prediksi
Sherlock Holmes	357436	125	1.4	61.83	51.78
Victor Frankenstein	37975	65	9,8	-59.23	-12.9225
Pirates of the Caribbean	498821	300	1.8	9.4	56.95
Final Destination	88000	40	0.07	2.58	2.94
Grown Ups 2	114482	80	3.8	54.67	-4.12

Tabel 4.37 menunjukkan hasil percobaan prediksi *profit* menggunakan fungsi prediksi yang dihasilkan. Hasil yang dimiliki polinom lebih baik dari linear. Berikut adalah nilai distribusi *squared error* dari percobaan fungsi prediksi data *test* pada Gambar 4.71 menggunakan *boxplot*.



Gambar 4.71: Distribusi nilai *squared errors* ( $(y_{pred} - y_{test})^2$ ) *Polynomial regression* pada prediksi *profit* menggunakan fitur *votes,budget* dan *viewCount*

Selain distribusi nilai *squared error*, berikut adalah skor akurasi R2 dari *polynomial regression* menggunakan data media sosial.

R2 prediksi Profit Polynomial Regression data sosial media
0.38

## 4.7 Analisis Clustering

Pada subbab ini dilakukan implementasi *clustering* pada *dataset*. Hasil *cluster* dianalisis untuk mengambil informasi seperti sifat *cluster* yang dihasilkan dan interpretasinya. *Clustering* dilakukan berdasarkan 2 fitur yaitu *genre* dan *aktor*.

### 4.7.1 Clustering Berdasarkan Aktor

*Clustering* berdasarkan aktor dilakukan untuk mengetahui sebuah kelompok dengan pasangan aktor yang sering bermain bersama. Fitur aktor pada *dataset* merupakan tipe data *String* berisi nama aktor yang dipisah berdasarkan koma.

Tabel 4.38: Contoh Aktor pada *Dataset*

Title	Aktor
Guardians of The Galaxy	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe Saldana
Star Trek Beyond	Chris Pine, Zachary Quinto, Karl Urban, Zoe Saldana

Berdasarkan Tabel 4.38, data aktor masih dalam bentuk *String* sehingga perlu untuk diubah dalam bentuk numerik agar *clustering* dapat dilakukan. Berdasarkan Subbab 2.6, *one hot encoding* dapat digunakan untuk mengubah data kategori menjadi numerik sehingga dapat menghitung kemunculan aktor pada suatu film.

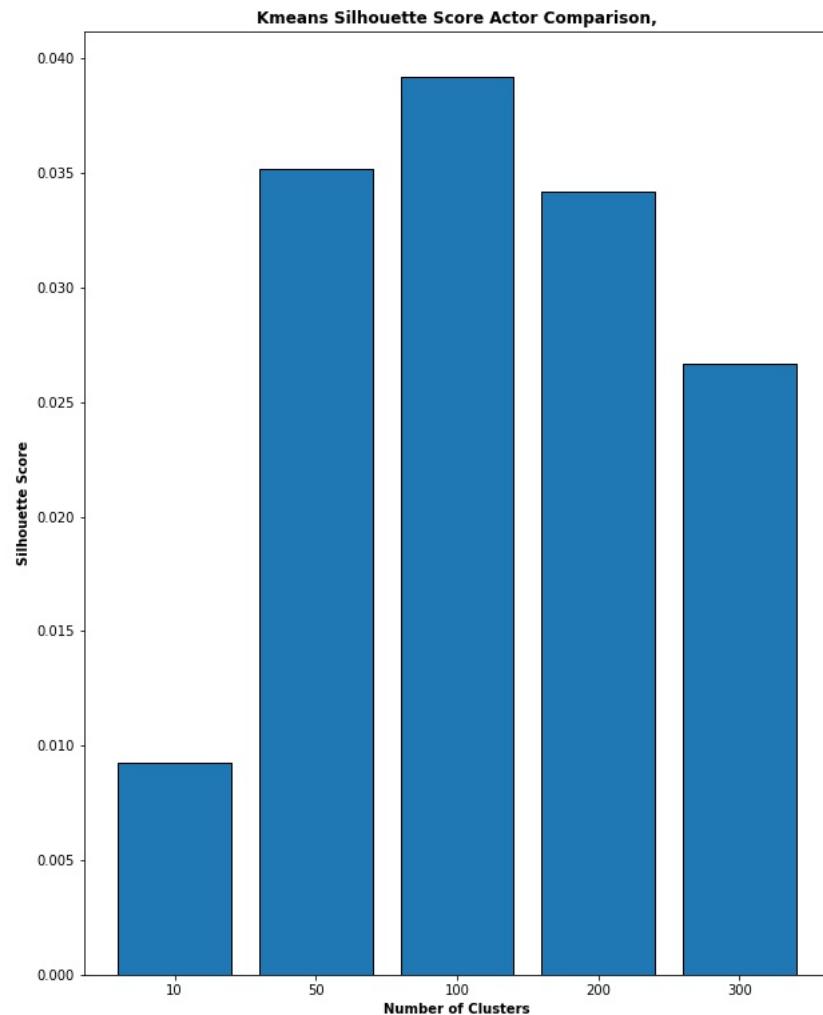
Tabel 4.39: Contoh Hasil One Hot Encoding Aktor

Title	Chris Pratt	Vin Diesel	Bradley Cooper	Zoe Saldana	Chris Pine	Karl Urban	Zachary Quinto
Guardians of The Galaxy	1	1	1	1	0	0	0
Star Trek Beyond	0	0	0	1	1	1	1

Tabel 4.39 adalah contoh hasil konversi fitur aktor menjadi numerik menggunakan *one hot encoding*. *One hot encoding* dilakukan untuk semua film pada *dataset* sehingga semua aktor dapat dihitung kemunculannya masing-masing. Untuk tiap baris film, nilai 0 menandakan bahwa aktor tersebut tidak bermain pada film sedangkan 1 menandakan bahwa aktor tersebut bermain terlibat dalam film. *Hasil one hot encoding* aktor digunakan untuk melakukan *clustering* menggunakan *K-Means* dan *Agglomerative*.

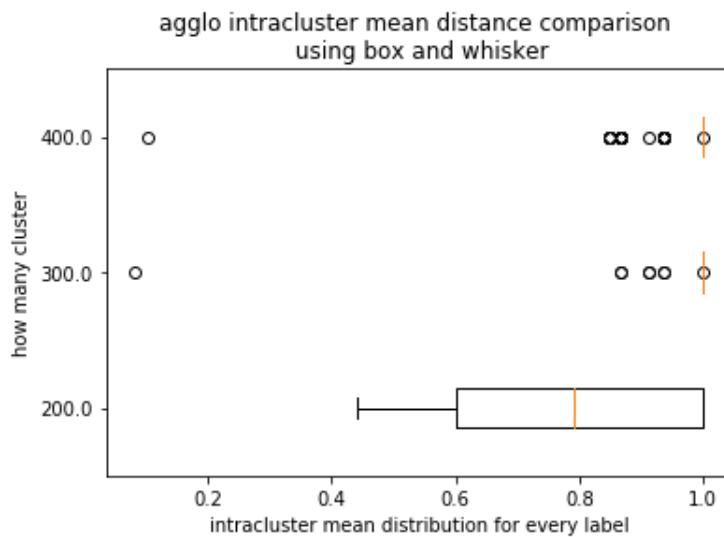
Perhitungan jarak tiap data objek / data film akan menggunakan *cosine distance*. *Cosine* digunakan karena ingin memenuhi syarat jika 2 film terdapat aktor yang sama, maka nilai jarak akan meningkat / semakin mirip.

Algoritma *K-Means* dan *Agglomeratives* menerima *input* berupa K / jumlah kelompok yang ingin dibuat. Jumlah nilai K yang dipilih mempengaruhi hasil *cluster* / seberapa baik kemiripan antar anggota pada *cluster*. Pada tahap ini dilakukan pengujian untuk menentukan nilai K terbaik menggunakan *Silhouette score*.



Gambar 4.72: Kualitas Clustering Aktor berdasarkan jumlah menggunakan *Silhouette Score*

Pengujian menentukan nilai K berdasarkan *Silhouette score* pada *barchart* Gambar 4.72 menunjukkan bahwa nilai kualitas *cluster* sangat buruk. Tiap bar merepresentasikan percobaan nilai K dan hasil uji yang diperoleh. Hal yang menyebabkan buruknya hasil *cluster* adalah *silhouette score* merupakan rata-rata dan penggabungan dari kualitas inter dan intra pada semua *cluster*. Kualitas *cluster* yang buruk disebabkan oleh sebuah film yang memiliki masing-masing 2-4 aktor dan jarang terjadi kemungkinan 2 film yang semua aktornya sama persis. Yang dibutuhkan pada saat ini adalah tiap *cluster* setidaknya terdapat satu aktor yang sama / berpasangan sehingga dilakukan pengujian *intraclass*. Berikut adalah distribusi nilai *intraclass* pada setiap percobaan nilai K pada *cluster* aktor.



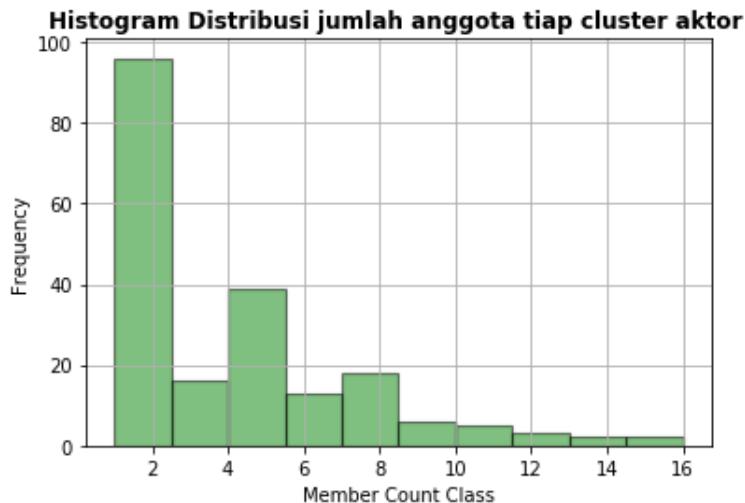
Gambar 4.73: Distribusi nilai *intracluster* pada beberapa percobaan clustering actor

Berdasarkan *boxplot* Gambar 4.73, nilai K yang paling tepat untuk menghasilkan *cluster* sesuai kebutuhan adalah K = 200. Distribusi pada K =300 dan K=400 tidak sesuai karena jika dilihat pada distribusinya 2 daerah yaitu yang nilai *intraclusternya* sangat buruk (*intracluster* < 0.2) dan sisanya yang sangat bagus ( 0.9 <= *intracluster* <= 1). Sifat distribusi tersebut akan menghasilkan satu *cluster* yang besar tetapi anggotanya tidak memiliki keunikan dan sisa *cluster* lain yang anggotanya hanya 1. Jika memilih K = 200 akan memenuhi persyaratan dimana setidaknya salah satu aktor yang bermain 2 film akan tergabung dalam satu *cluster*. Selain pemilihan jumlah K, berikut adalah perbandingan performa *clustering K-Means* dan *Agglomeratives*.

Tabel 4.40: Perbandingan waktu *clustering* menggunakan KMeans dan Agglomeratives

Waktu KMeans	Waktu Agglomeratives
752 detik	9 detik

Berdasarkan Tabel 4.40 , *clustering* aktor dengan menggunakan *Agglomerative* lebih cepat dari *KMeans*. *Agglomerative clustering* lebih cepat karena setiap iterasi penggabungan anggota mengurangi jumlah anggota yang harus digabungkan sedangkan *K-Means* setiap iterasi tetap menghitung jarak tiap data dengan *centroid* sesuai jumlah data. Hasil *clustering* memberikan sebuah label pada tiap data film yang berupa numerik dari 0-199. Berikut adalah distribusi jumlah anggota tiap kelompok dari hasil *clustering* aktor.



Gambar 4.74: Distribusi jumlah anggota tiap *cluster* aktor

Barchart Gambar 4.74 adalah distribusi jumlah anggota tiap *cluster* aktor. Sumbu y adalah frekuensi / jumlah kelompok pada suatu batang dan sumbu x adalah kelas jumlah anggota. Terdapat banyak kelompok yang anggotanya masih sendiri disebabkan tidak ada pasangan aktor yang mirip.

Tabel 4.41: Contoh Anggota *Cluster* 13,28 dan 161 hasil *cluster Actor*

intracluster	label	title	actors	runtime	votes	rating	metascore	revenue	profit	genre
0.61	13	Iron Man	Robert Downey Jr., Gwyneth Paltrow, Terrence Howard, Jeff Bridges	126	737719	7.9	79	318.8	178	Action, Adventure, Sci-Fi
0.61	13	Iron Man 2	Robert Downey Jr., Mickey Rourke, Gwyneth Paltrow, Don Cheadle	124	556666	7	57	312.06	112.06	Action, Adventure, Sci-fi
0.61	13	Iron Man 3	Robert Downey Jr., Guy Pearce, Gwyneth Paltrow, Don Cheadle	130	591023	7.2	62	408.99	208	Action, Adventure, Sci-fi
0.51	161	Hotel Transylvania 2	Adam Sandler, Andy Samberg, Selena Gomez, Kevin James	89	69157	6.7	44	169	89	Animation, Comedy, Family
0.51	161	Just Go with It	Adam Sandler, Jennifer Aniston, Brooklyn Decker, Nicole Kidman	117	182069	6.4	33	103.33	23	Comedy, Romance
0.51	161	Grown Ups 2	Adam Sandler, Kevin James, Chris Rock, David Spade	101	114482	5.4	19	133	53	Comedy
0.79	28	Trolls	Anna Kendrick, Justin Timberlake, Zooey Deschanel, Christopher Mintz-Plasse	92	38552	6.5	56	153.69	28.69	Animation, Adventure, Comedy
0.79	28	Gabor Csopo	Josh Hutcherson, AnnaSophia Robb, Zooey Deschanel, Robert Patrick	96	117297	7.2	74	82.23	65.23	Adventure, Drama, Family

Tabel 4.41 adalah contoh 3 dari 200 kelompok hasil *cluster* berdasarkan aktor yaitu kelompok

13,161 dan 28. Masing-masing *cluster* terbukti memiliki aktor dominan contohnya *cluster* 13 memiliki aktor dominan yang sering muncul adalah Robert Downey Jr. , *cluster* 161 adalah Adam Sandler dan *cluster* 28 adalah Zooey Deschanel. Selain mengetahui aktor dominan untuk tiap *cluster*, dapat diketahui juga *genre* favorit dari tiap aktor seperti Robert Downey Jr. paling banyak bermain di *genre* Action, Adventure dan Sci-fi . Adam Sandler paling sering bermain di *genre* Comedy dan Zooey Deschanel paling sering bermain *genre* Adventure.

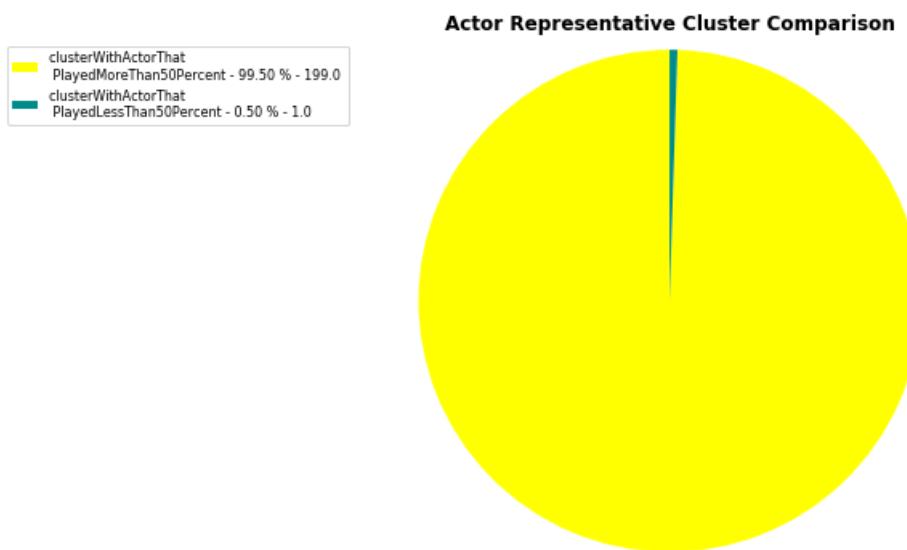
Hasil *cluster* aktor perlu diinterpretasi untuk mengetahui makna dari tiap *cluster*. Terdapat beberapa temuan interpretasi dari hasil analisis *cluster* aktor.

### Tiap Cluster Merepresentasikan 1 Aktor Representatif

Pengujian pertama adalah memastikan jika 1 *cluster* merepresentasikan 1 aktor dominan. Aktor dominan dihitung berdasarkan kontribusi jumlah film dimana suatu aktor tergabung dalam suatu *cluster*. Contoh jika pada sebuah *cluster* terdapat 4 film dan Aktor "A" terlibat dalam satu film, maka kontribusi jumlah "A" dalam *cluster* tersebut adalah 25 persen.

$$Kontribusi(actor, cluster) = \frac{Jumlah\ filmyang\ aktormain\ kandicluster}{Jumlah\ semua\ filmpadacluster} * 100 \quad (4.9)$$

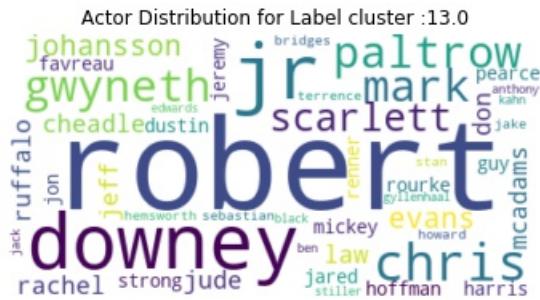
Persamaan 4.9 adalah cara menghitung kontribusi seorang aktor pada *cluster* tersebut Batas yang ditentukan jika seorang aktor adalah aktor dominan dalam sebuah *cluster* jika kontribusinya lebih dari 50 persen. Akan dihitung perbandingan jumlah *cluster* yang memenuhi syarat dan tidak, untuk menguji kebenaran temuan.



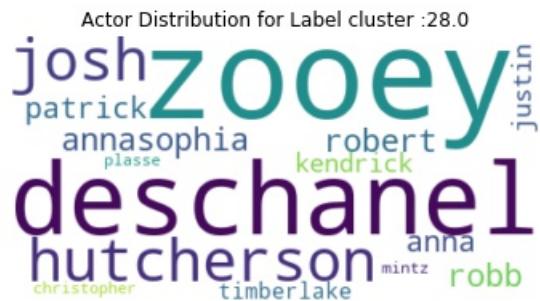
Gambar 4.75: Pengujian sebuah cluster merepresentasikan satu aktor dominan

Gambar 4.75 adalah hasil visualisasi pengujian tiap *cluster* merepresentasikan aktor utama menggunakan piechart. Dari 200 *cluster* aktor, 99.5 persen (199 *cluster*) memiliki seorang aktor yang kontribusi jumlah filmnya lebih dari 50 persen. Dapat disimpulkan *cluster* dapat digunakan untuk menemukan aktor dominan. Seorang aktor dominan lebih sering bermain film dibanding yang lain sehingga metode ini dapat digunakan untuk menemukan pasangan aktor dominan yang bagus. Ada kemungkinan berbeda pasangan main akan mempengaruhi kualitas dan pendapatan film.

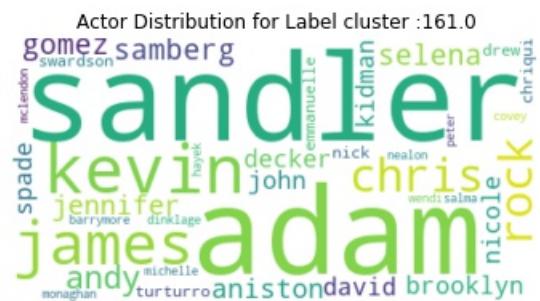
Visualisasi word cloud dapat membantu menemukan aktor dominan pada sebuah *cluster*. Semakin besar nama seorang aktor maka artinya aktor tersebut adalah aktor utama dari *cluster*.



Gambar 4.76: Wordcloud distribusi aktor cluster 13



Gambar 4.77: Wordcloud distribusi aktor cluster 28



Gambar 4.78: Wordcloud distribusi aktor cluster 161

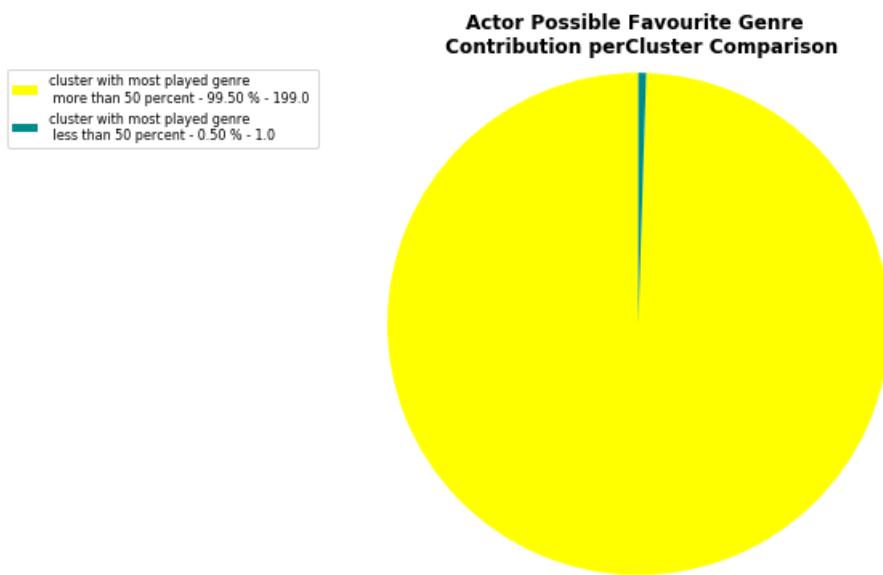
#### Aktor Representatif Memiliki Genre Favorit

Berdasarkan temuan sebelumnya, sebuah *cluster* memiliki aktor representatif. Pada tahap ini diuji apakah aktor dominan memiliki *genre* favorit. *Genre* favorit adalah jumlah kontribusi sebuah *genre*

pada suatu *cluster* lebih dari 50 persen. Kontribusi dapat dihitung dengan rumus sebagai berikut.

$$Kontribusi(G, cluster) = \frac{JumlahFilmyangterdapatgenreGpadacluster}{jumlahsemuafilmpadacluster} * 100 \quad (4.10)$$

Persamaan 4.10 digunakan untuk menghitung kontribusi *genre* pada sebuah film. Pada tahap ini dilakukan perhitungan kontribusi tiap *genre* pada masing-masing *cluster*. *Genre* yang paling muncul dan kontribusi diatas 50 persen maka disebut sebagai *genre* favorit dalam sebuah *cluster*. Dengan mengkombinasikan aktor dominan dengan *genre* yang memiliki kontribusi di atas 50 persen, maka sebuah aktor dominan memiliki *genre* favorit. Pada tahap ini juga akan dihitung jumlah *cluster* yang memenuhi syarat kontribusi *genre* di atas 50 persen. Berikut adalah visualisasi distribusi kontribusi *genre* favorit dari tiap *cluster* untuk pengujian setiap aktor memiliki *genre* favorit.



Gambar 4.79: Pengujian aktor dominan memiliki *genre* favorit pada tiap *cluster*

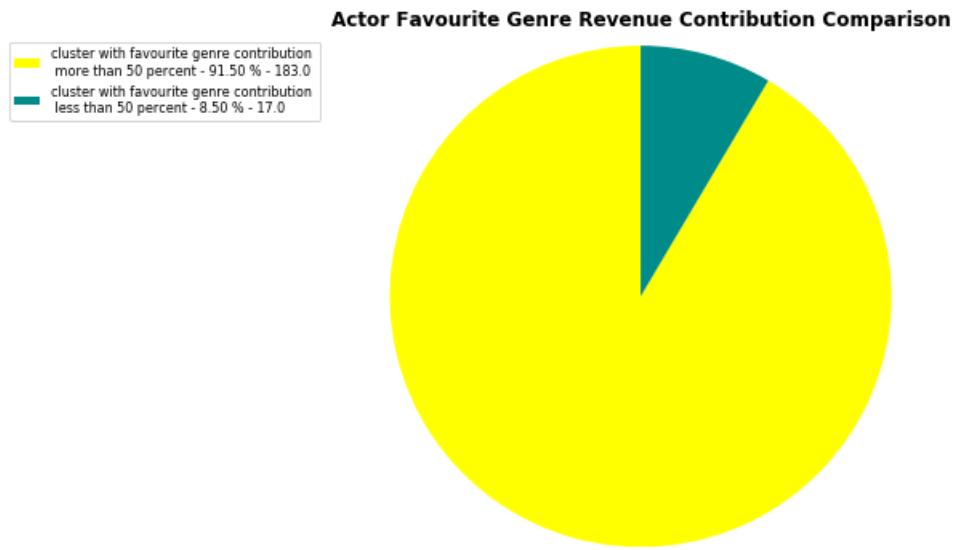
Berdasarkan *piechart* pada Gambar 4.79, hampir semua aktor dominan pada tiap *cluster* memiliki *genre* favorit. Dapat disimpulkan tiap aktor memiliki *genre* favorit. Mengetahui *Genre* favorit tiap aktor akan membantu meningkatkan performa seorang aktor dan membantu memperoleh kesuksesan.

### Genre Favorit Berkontribusi Menguntungkan

Tiap Aktor merepresentasikan seorang aktor dominan/ representatif. Aktor representatif memiliki *genre* favorit yaitu jumlah *genre* film yang dimainkan berkontribusi banyak tetapi seorang aktor yang sering bermain suatu *genre* belum tentu menguntungkan. Pada tahap ini diuji apakah *genre* favorit tiap aktor representatif juga berkontribusi dalam memperoleh *revenue*.

$$KontribusiRevenue(Genre, Cluster) = \frac{JumlahRevenueGenrepadaCluster}{JumlahRevenuepadaCluster} * 100 \quad (4.11)$$

Persamaan 4.11 digunakan untuk menghitung kontribusi *revenue* jenis *genre* sebuah *cluster*. Akan dilakukan pengujian untuk menghitung jumlah *cluster* yang *genre* favorit dari aktor tidak hanya berkontribusi dari jumlah film tetapi juga menguntungkan. Syarat sebuah *genre* pada *cluster* menguntungkan adalah kontribusi *revenue* dari total perolehan *revenue*. Berikut adalah visualisasi jumlah perbandingan *cluster* yang memenuhi syarat sebuah *genre* menguntungkan.



Gambar 4.80: Pengujian *genre* favorit pada tiap *cluster* juga berkontribusi tinggi untuk keuntungan

Berdasarkan *piechart* pada Gambar 4.80, mayoritas dari semua *cluster* terbukti *genre* favorit aktor memiliki kontribusi keuntungan paling tinggi. Berdasarkan analisis *cluster* yang diberikan penting untuk mengetahui *genre* apa yang diminati aktor sehingga menambah kemungkinan untuk meningkatkan kesuksesan.

Tabel 4.42: Contoh jumlah kontribusi aktor utama pada *cluster* 13 , 28 dan 161

Label / Cluster	Aktor Utama	Kontribusi Keterlibatan film	Genre Favorit	Kontribusi Revenue Genre Favorit
13	Robert Downey Jr.	100 %	Action	97 %
161	Adam Sandler	100 %	Comedy	100%
28	Zooey Deschanel	100 %	Adventure	100 %

Berdasarkan hasil analisis *cluster* berdasarkan aktor, informasi yang diperoleh adalah :

- Algoritma *Agglomerative* lebih cepat dari algoritma *K-Means* dalam melakukan *clustering*
- Tiap aktor mempunyai *genre* favorit berdasarkan seberapa banyak aktor itu bermain di *genre* tertentu
- Penting untuk mengetahui *genre* favorit aktor karena dapat meningkatkan kemungkinan keuntungan yang diperoleh

#### 4.7.2 Clustering Berdasarkan Genre

Selain *clustering* berdasarkan aktor, pada tahap ini dilakukan berdasarkan fitur *genre*. Sebuah film memiliki lebih dari 1 *genre* sehingga *clustering* dilakukan untuk mengelompokkan anggota film yang memiliki 1 kombinasi *genre* yang sama. Hasil *cluster* dianalisis untuk mengambil informasi seperti sifat *cluster* yang dihasilkan dan interpretasinya. Fitur *genre* pada *dataset* merupakan tipe data *String* berisi *genre* yang dipisah berdasarkan koma.

Tabel 4.43: Contoh Genre pada *Dataset*

Title	Genre
Prometheus	Adventure,Mystery,Sci-Fi
The Hunger Games	Adventure,Sci-Fi,Thriller
Minions	Animation,Action,Adventure

Berdasarkan Tabel 4.43, data *genre* masih dalam bentuk *String* sehingga perlu untuk diubah dalam bentuk numerik agar *clustering* dapat dilakukan. *One hot encoding* dapat digunakan untuk mengubah data kategori menjadi numerik sehingga dapat menghitung kemunculan *genre* pada suatu film.

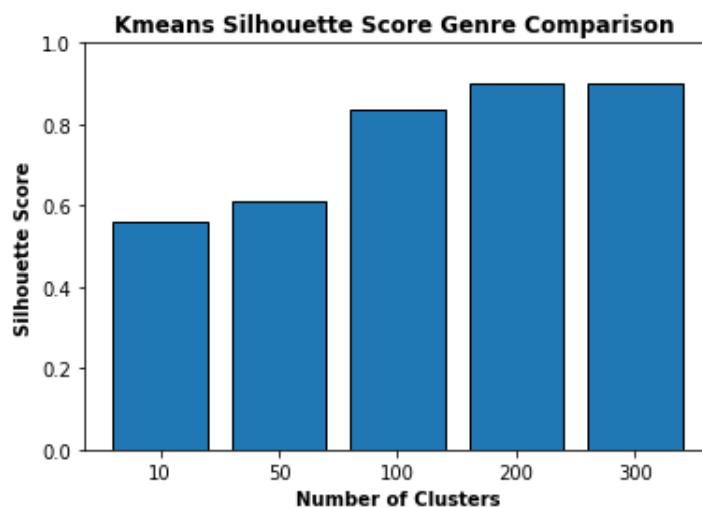
Tabel 4.44: Contoh Hasil One Hot Encoding Genre

Title	Adventure	Mystery	Sci-fi	Thriller	Animation	Action
Prometheus	1	1	1	0	0	0
The Hunger Games	1	0	1	1	0	0
Minions	1	0	0	0	1	1

Tabel 4.44 adalah hasil konversi fitur *genre* menjadi numerik menggunakan *one hot encoding*. *One hot encoding* dilakukan untuk semua film pada *dataset* sehingga tiap *genre* dapat dihitung kemunculannya. Untuk tiap baris film, nilai 1 menandakan bahwa film mengandung suatu *genre* dan 0 tidak mengandung suatu *genre*. Hasil *one hot encoding genre* digunakan untuk melakukan *clustering* menggunakan *K-Means* dan *Agglomerative*.

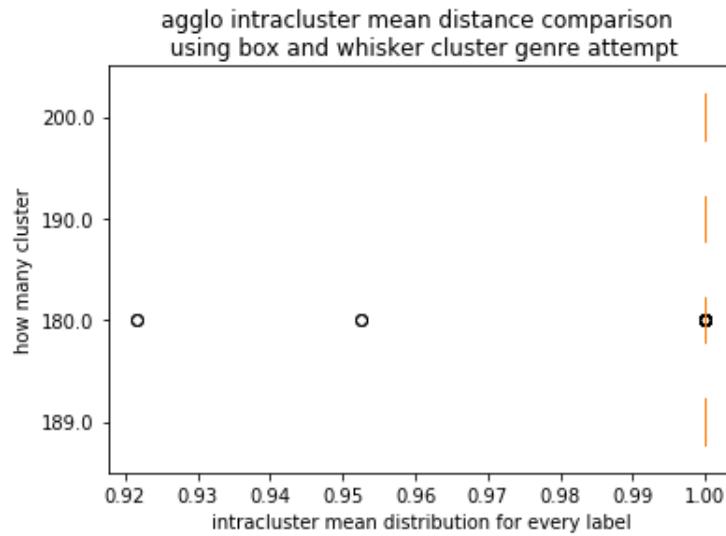
Perhitungan jarak tiap data objek / data film menggunakan *cosine distance*. *Cosine* semakin besar jika dua film mengandung *genre* yang mirip.

Algoritma *clustering* menerima *input* berupa K / jumlah kelompok yang ingin dibuat. Jika dihitung, terdapat 186 kombinasi *genre* yang berbeda. Pada tahap ini dilakukan pengujian untuk menentukan jumlah K terbaik menggunakan *Silhouette score*.

Gambar 4.81: Kualitas Clustering Genre berdasarkan jumlah menggunakan *Silhouette Score*

Pengujian menentukan nilai K untuk *clustering genre* menggunakan *Silhouette score* pada Gambar 4.81 menunjukkan bahwa K=200 sudah mendapatkan nilai *silhouette* yang hampir sempurna. Pengujian ini sesuai dengan tujuan utama yaitu mengelompokkan tiap kombinasi *genre* yang sama. Jika menggunakan K=300 akan menyebabkan *cluster-cluster* yang seharusnya sekelompok menjadi terpisah. Nilai K di bawah 200 menyebabkan terdapat *cluster* yang menggabungkan kombinasi

*genre* yang berbeda. Selain *silhouette score*, maka akan dilakukan pengujian pemilihan K terbaik menggunakan *intraclass*.



Gambar 4.82: Distribusi nilai *intraclass* pada beberapa percobaan clustering *genre*

Berdasarkan *boxplot* pada Gambar 4.82, nilai K yang paling tepat untuk menghasilkan *cluster* yang sesuai adalah K = 189. Dapat dilihat bahwa K = 189 menghasilkan distribusi yang hanya ada pada nilai *intraclass* hampir = 1.0. Artinya semua anggota pada tiap *cluster* memiliki jarak yang sangat dekat / sama persis. Berdasarkan kesimpulan sebelumnya pada analisis *clustering* aktor, *Agglomerative clustering* lebih cepat daripada *K-Means* sehingga *clustering genre* akan menggunakan *Agglomerative*.

Hasil *clustering genre* memberikan sebuah label pada tiap data film yang berupa numerik dari 0-188. Berikut adalah contoh beberapa kelompok dari hasil *clustering genre*.

Tabel 4.45: Contoh Anggota Cluster 13,28 dan 62 hasil cluster Genre

intraclass	label	title	actors	runtime	votes	rating	metascore	revenue	profit	genre
1	13	Seven Psychopath	Colin Farrell, Woody Harrelson, Sam Rockwell, Christopher Walken	110	196652	7.2	66	14.99	-0.01	Comedy, Crime
1	13	Horrible Bosses	Jason Bateman, Charlie Day, Jason Sudeikis, Steve Wible	98	368556	6.9	57	117.53	82	Comedy, Crime
1	13	Horrible Bosses 2	Jason Bateman, Jason Sudeikis, Charlie Day, Jennifer Aniston	108	125190	6.3	40	54.43	12	Comedy, Crime
1	28	A United Kingdom	David Oyelowo, Rosamund Pike, Tom Felton, Jack Davenport	111	4771	6.8	65	3.9	-10.1	Biography, Drama, Romance
1	28	The Danish Girl	Eddie Redmayne, Alicia Vikander, Amber Heard, Ben Whishaw	119	110773	7	66	12.71	-2.29	Biography, Drama, Romance
1	28	The Theory of Everything	Eddie Redmayne, Felicity Jones, Tom Prior, Sophie Perry	123	299718	7.7	72	35	20	Biography, Drama, Romance
1	62	Spotlight	Mark Ruffalo, Michael Keaton, Rachel McAdams, Liev Schreiber	128	268282	8.1	93	44.93	23.99	Crime, Drama, History
1	62	Zodiac	Jake Gyllenhaal, Robert Downey Jr., Mark Ruffalo, Anthony Edwards	157	329683	7.7	78	33.05	-31.95	Crime, Drama, History

Tabel 4.45 adalah contoh 3 dari 189 kelompok hasil cluster berdasarkan genre yaitu kelompok 13,28 dan 62. Dapat lihat nomor label yang sama menandakan film sekelompok dan kombinasi genre sudah sama. Berdasarkan hasil cluster dapat lihat terdapat aktor yang sering muncul pada sebuah kombinasi genre. Hasil cluster membantu melihat apakah seorang aktor cocok untuk bermain pada sebuah kombinasi genre.

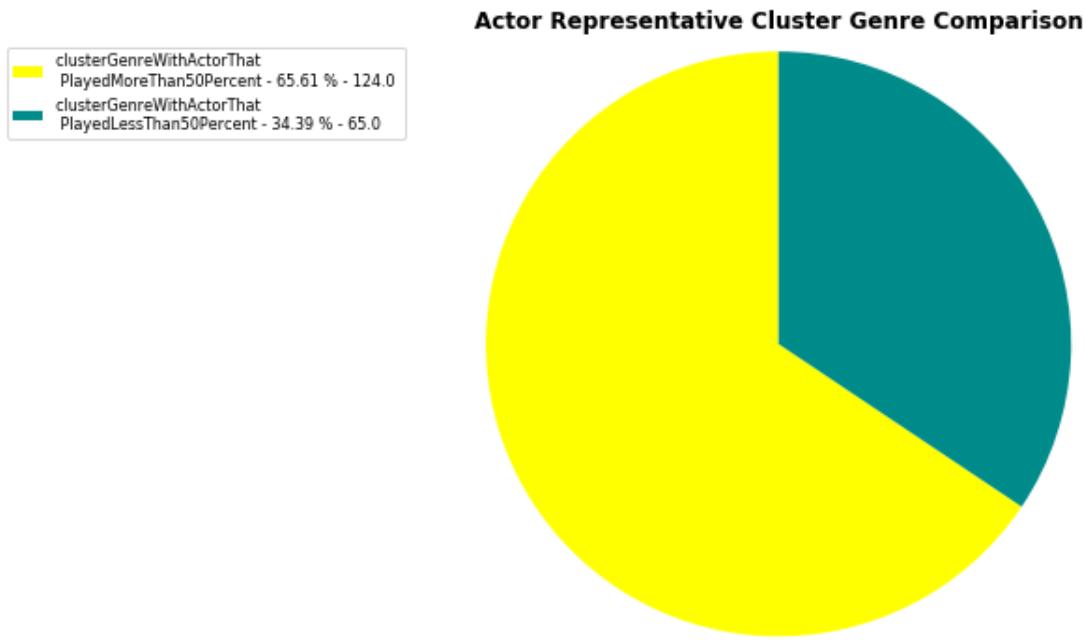
Hasil cluster genre perlu diinterpretasi untuk mengetahui pola yang menarik. Pada tahap selanjutnya dilakukan pengujian beberapa temuan yang dibuat.

### Tiap Kombinasi Genre Mempunyai Aktor Utama

Pengujian pertama adalah menentukan jika sebuah kombinasi genre memiliki 1 aktor utama. Aktor utama dihitung berdasarkan kontribusi jumlah film dimana suatu aktor sering bermain kombinasi genre tersebut. Contoh jika pada sebuah cluster dengan kombinasi genre "Action, Adventure" terdapat 5 film dan *Robert Downey Jr.* bermain di 3 film maka kontribusi *Robert Downey Jr.* bermain genre Action, Adventure adalah 60 persen.

$$Kontribusi(Actor, ClusterGenre) = \frac{Jumlah filmyang actor mainkan di klasifikasi ClusterGenre}{Jumlah film pada klasifikasi ClusterGenre} * 100 \quad (4.12)$$

Persamaan 4.12 adalah cara menghitung kontribusi seorang aktor pada suatu kombinasi genre / sebuah cluster genre. Batas yang ditentukan jika seorang aktor adalah aktor utama dalam sebuah cluster jika kontribusinya di atas 50 persen. Pada tahap ini dihitung perbandingan jumlah cluster yang memenuhi syarat dan tidak. Berikut adalah visualisasi dari pengujian jumlah aktor utama.



Gambar 4.83: Perbandingan jumlah *cluster* yang memiliki aktor utama dan tidak

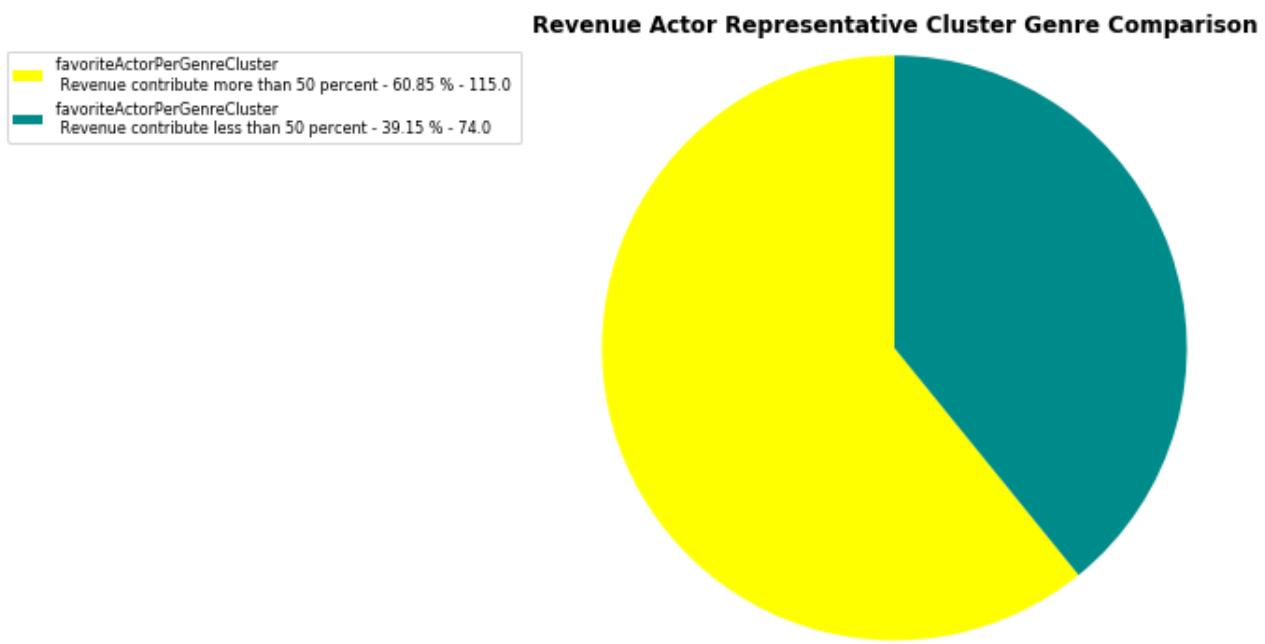
Gambar 4.83 adalah hasil visualisasi pengujian tiap kombinasi *genre* mempunyai aktor utama menggunakan *piechart*. Dari 189 *cluster*, 124 *cluster* memiliki aktor utama yang kontribusi bermain pada suatu kombinasi *genre* lebih dari 50 persen. Dapat disimpulkan *cluster genre* dapat digunakan untuk menemukan aktor utama dari suatu kombinasi *genre* sehingga metode ini dapat digunakan untuk mencari aktor-aktor yang meningkatkan kemungkinan suatu film berhasil.

#### Revenue yang dihasilkan Aktor Utama Berkontribusi Untuk Kombinasi Genre

Berdasarkan pengujian sebelumnya, sebuah *cluster genre* / kombinasi *genre* memiliki aktor utama. Pada tahap ini diuji apakah aktor utama yang berkontribusi secara jumlah film juga berkontribusi tinggi dalam meraih *revenue*. Pada pengujian ini, kontribusi *revenue* suatu *genre* yang lebih dari 50 persen dianggap sebagai yang menguntungkan

$$KontribusiRevenue(actor, clusterGenre) = \frac{JumlahRevenueFilmAktorPadaClusterGenre}{JumlahRevenueSemuaFilmPadaClusterGenre} * 100 \quad (4.13)$$

Persamaan 4.13 dapat digunakan untuk menghitung kontribusi *revenue* aktor utama pada sebuah *cluster genre*. Pada tahap ini diuji apakah aktor utama pada *cluster genre* juga mempengaruhi *revenue*. Berikut adalah perbandingan jumlah *cluster* yang memenuhi dan tidak.



Gambar 4.84: Perbandingan jumlah *cluster* yang aktor utama berkontribusi berdasarkan *revenue* pada kombinasi *genre*

Berdasarkan *piechart* pada Gambar 4.84, mayoritas dari semua *cluster* aktor utama memiliki kontribusi *revenue* paling tinggi. Mengetahui aktor-aktor utama yang berkontribusi pada suatu kombinasi *genre* dapat meningkatkan kemungkinan pendapatan *revenue*.

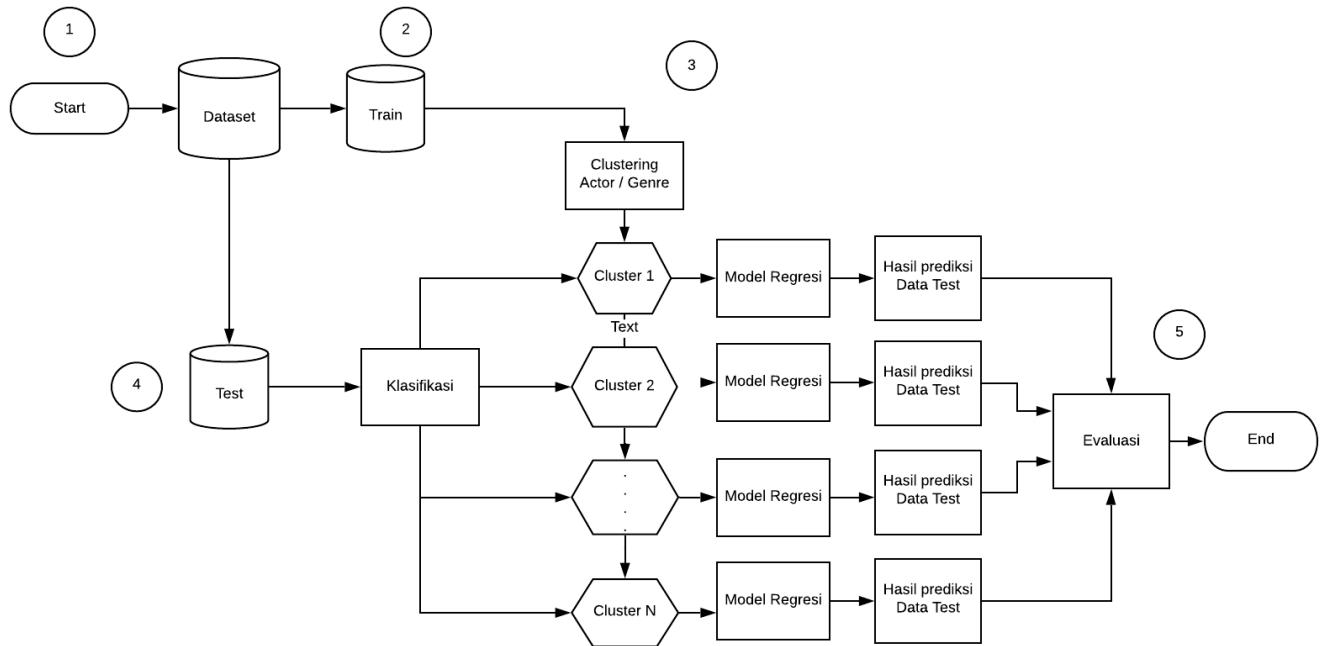
Berdasarkan hasil analisis *cluster* berdasarkan *genre*, informasi yang diperoleh adalah :

- Tiap kombinasi *genre* memiliki sifat yang berbeda
- Kecocokan suatu kombinasi *genre* dan aktor yang bermain berpengaruh terhadap kualitas sebuah film
- Perlu mengetahui aktor utama suatu kombinasi *genre* untuk meningkatkan kemungkinan kesuksesan kualitas film

## 4.8 Prediksi Berdasarkan Cluster

Pada tahap ini dilakukan percobaan prediksi data *test* menggunakan *segmented cluster*. *Segmented cluster* adalah proses untuk memprediksi tiap *data test* dengan model regresi yang berbeda-beda. Sebuah data *test* diklasifikasi terlebih dahulu berdasarkan *genre* dan aktor. Setelah data *test* menemukan kelompok *genre* / aktor yang sama, maka anggota kelompok tersebut dijadikan *data train* untuk membuat model regresi.

Ilustrasi *flowchart* proses *segmented cluster* pada Gambar 4.85.



Gambar 4.85: Segmented Cluster Flowchart

Tahap proses yang dilakukan *segmented cluster* adalah :

1. Membagi *dataset* menjadi 80 persen *train* dan 20 persen *test*
2. Data *train* dicluster berdasarkan aktor dan *genre* secara terpisah. Tiap *cluster* akan membuat model regresi *Linear Regression* dan *Polynomial Regression* masing-masing berdasarkan *data train*.
3. Sebuah film / Data *test* akan diklasifikasi ke kelompok Data *train* yang tepat berdasarkan aktor / *genre*. Proses ini diulang untuk semua data *test*
4. Data *test* yang sudah diklasifikasi akan menggunakan model regresi sesuai label yang diperoleh setelah klasifikasi untuk melakukan prediksi *revenue*
5. Tiap prediksi masing-masing data *test* akan dievaluasi

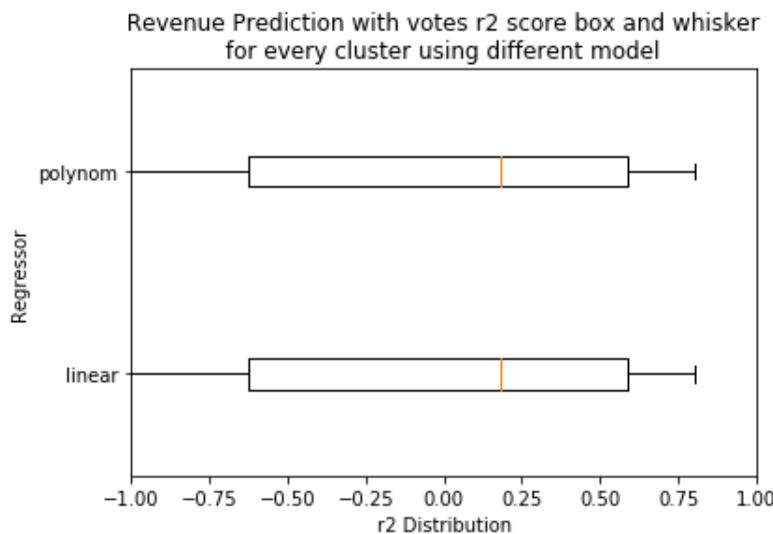
#### Hasil Evaluasi Prediksi Berdasarkan Cluster *Genre*

Pada tahap ini pengujian *segmented cluster* melakukan prediksi *revenue* berdasarkan *clustering genre*. Berikut adalah tabel hasil R2.

Tabel 4.46: Hasil evaluasi prediksi Revenue menggunakan *Votes* dengan R2 Clustering *Genre*

Nilai R2 Segmented Cluster <i>Genre</i> Prediksi Revenue Menggunakan <i>Votes</i>	
Linear	Polynomial
0.27	-3.1

Berdasarkan Tabel 4.46, hasil R2 yang diperoleh dari pengujian *segmented cluster* lebih kecil dari hasil prediksi Hasil Analisis Data Utama pada Subbab 4.4. Berikut adalah *boxplot* distribusi R2 hasil prediksi *revenue segmented cluster* menggunakan *genre* secara detail.



Gambar 4.86: Distribusi R2 hasil prediksi revenue berdasarkan *cluster genre*

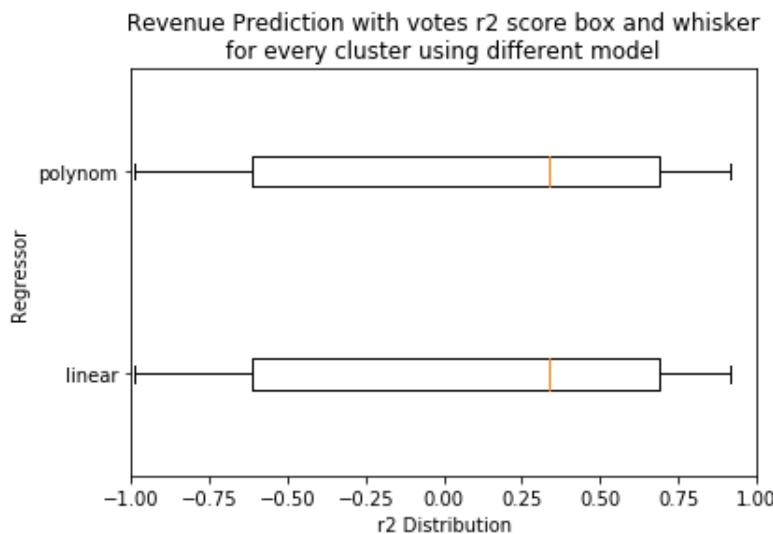
#### Hasil Evaluasi Prediksi Berdasarkan Cluster Actor

Pada tahap ini pengujian *segmented cluster* melakukan prediksi *revenue* berdasarkan *clustering actor*. Berikut adalah tabel hasil R2.

Tabel 4.47: Hasil evaluasi prediksi Revenue menggunakan *Votes* dengan R2 Clustering Actor

Nilai R2 Segmented Cluster Actor Prediksi Revenue Menggunakan Votes	
Linear	Polynomial
0.028	-365.1

Berdasarkan Tabel 4.47, hasil R2 yang diperoleh dari pengujian *segmented cluster* lebih kecil dari hasil prediksi *cluster aktor* dan Hasil Analisis Data Utama pada Subbab 4.4. Berikut adalah piechart distribusi R2 hasil prediksi *revenue segmenred cluster* menggunakan aktor secara detail.



Gambar 4.87: Distribusi R2 hasil prediksi revenue berdasarkan *cluster aktor*

Berdasarkan hasil pengujian *segmented cluster* berdasarkan aktor dan *genre*, percobaan prediksi *revenue* menghasilkan nilai akurasi yang lebih buruk dari prediksi regresi tanpa *segmented cluster*.

Hal ini disebabkan tiap *cluster* memiliki data *train* yang jauh lebih sedikit dari percobaan regresi biasa yang menggunakan semua data *train* tanpa dipisah.



## BAB 5

### KESIMPULAN DAN SARAN

Bab ini akan membahas kesimpulan yang didapat dari hasil penelitian ini dan saran yang dapat diberikan untuk pengembangan penelitian ini lebih lanjut.

#### 5.1 Kesimpulan

Kesimpulan yang dihasilkan dari penelitian menggunakan *dataset* yang digunakan adalah :

- Faktor yang paling berpengaruh dalam menentukan kesuksesan film berdasarkan *dataset* yang digunakan antara lain adalah *votes*, *budget* dan jumlah *view trailer* Youtube. Evaluasi akurasi prediksi *revenue* menggunakan R2 dapat mencapai 0.63 berdasarkan penggunaan fitur tersebut. *Votes*, *Budget* dan *Youtube* adalah 3 fitur yang memiliki korelasi tertinggi dengan *revenue* dibanding fitur lain berdasarkan pengujian *pearson*.
- Berdasarkan *dataset* yang dianalisis, selera penonton berbeda dengan selera kritikus *review* dalam menilai bagus tidaknya sebuah film. Selera penonton (*votes*) memiliki hubungan korelasi positif dengan *revenue* yang lebih tinggi yaitu 0.6 . Nilai korelasi *pearson votes* memiliki nilai yang lebih tinggi dibanding selera kritikus (*review*) yaitu 0.2.
- Grafik tren nilai akumulasi *revenue* , *profit* dan *budget* dari tahun ke tahun meningkat berdasarkan *dataset* yang dianalisis. Pada tahun 2011, terjadi penurunan yaitu nilai akumulasi *revenue* yang menurun dibanding tahun sebelumnya. Hal ini disebabkan oleh film-film tahun 2011 yang lebih banyak menghasilkan *revenue* yang lebih kecil dari tahun 2010. Terjadi peningkatan jumlah film yang dibuat tiap tahunnya.
- Berdasarkan *dataset* yang dianalisis, tiap kombinasi *genre* film memiliki rentang pendapatan yang berbeda. Visualisasi distribusi *revenue* *boxplot* tiap kombinasi *genre* seperti contoh kombinasi *Action, Adventure, Mystery* memiliki nilai Q2 yang lebih besar dari *Action, Drama, Fantasy*. Pemilihan *genre* pada pembuatan film mempengaruhi rentang *revenue* yang dapat diperoleh.
- Berdasarkan *dataset* yang dianalisis, *budget* yang besar tidak menjamin *profit* yang diperoleh akan besar. Visualisasi *barchart* perbandingan 10 *profit* tertinggi pada tiap kombinasi *genre* menunjukkan film dengan kombinasi *genre Horror, Mystery, Thriller* memiliki *budget* yang sangat kecil tetapi mendapatkan keuntungan yang besar.
- Tiap aktor memiliki *genre* favorit. *Genre* favorit aktor adalah *genre* yang paling sering dimainkan seorang aktor dan memiliki kontribusi jumlah film paling banyak. *Genre* favorit aktor cenderung berkontribusi menghasilkan *revenue* yang tinggi dibanding *genre* lain berdasarkan pengujian *clustering* aktor.
- Algoritma *Agglomerative* lebih cepat dibandingkan dengan algoritma *K-Means* dalam melakukannya *clustering*. Berdasarkan pengujian *clustering* pada *dataset*, waktu yang dibutuhkan *Agglomerative* adalah 9 detik sedangkan *K-Means* adalah 752 detik. *Agglomerative* lebih cepat dari *K-Means* karena *Agglomerative* tiap iterasinya akan menggabungkan 2 data objek dan

mengurangi jumlah *cluster* terpisah sedangkan *K-Means* tiap iterasi akan menghitung jarak data objek dengan *centroid*.

- *Hashtag* Instagram pada *dataset* yang dianalisis tidak memiliki korelasi positif yang kuat dengan *revenue*. *Hashtag* Instagram mengandung kata-kata yang orang sering gunakan seperti 'Red' dan 'Vacation'. Hal ini menyebabkan data *Hashtag* mengandung noise
- Hubungan korelasi positif Youtube dengan *Revenue* lebih tinggi dari Instagram berdasarkan pengujian korelasi dengan *pearson*. Hal ini disebabkan oleh pengaruh kesalahan data *hashtag* judul film di Instagram pada kesimpulan sebelumnya.
- Pengujian prediksi *revenue* berdasarkan *cluster* menghasilkan nilai evaluasi akurasi R2 yang sangat kecil yaitu 0.23 . Hal ini disebabkan oleh jumlah data *train* yang sangat sedikit setelah *dicluster* sehingga model prediksi tidak valid untuk diuji.

## 5.2 Saran

Saran yang dapat dilakukan untuk memperbaiki dan mengembangkan penelitian ini lebih lanjut :

- Merubah metode prediksi dengan mengubah model regresi menjadi model klasifikasi. Film-film pada *dataset* dapat dikelompokkan berdasarkan *revenue* / *profit*.
- Menambah ukuran *dataset* yang dianalisis. Penambahan jumlah film pada *dataset* akan membantu mengatasi kendala ketika *dataset* sudah *dicluster* tidak mengalami kekurangan data *train*.

## **DAFTAR REFERENSI**

- [1] Han, J., Kamber, M., dan Pei, J. (2012) Data mining concepts and techniques, third edition.
- [2] Tan, P.-N., Steinbach, M., Karpatne, A., dan Kumar, V. (2020) *Introduction to data mining*. Pearson.
- [3] Draper, N. R. dan Smith, H. (1998) *Applied regression analysis*. John Wiley & Sons.
- [4] Manning, C. D., Raghavan, P., dan Schutze, H. (2018) *Introduction to information retrieval*. Cambridge University Press.



# LAMPIRAN A

## KODE PROGRAM

Listing A.1: aggregationAdditionalExperiment.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Apr  4 17:49:17 2020
4
5 @author: lenovo
6 """
7
8
9 import pandas as pd
10 import matplotlib.pyplot as plt
11 import plotModule
12 import numpy as np
13 import os
14 import shutil
15
16 dataset = pd.read_csv('hasilEksperimen_FinalForm.csv')
17
18
19 all_genre_combination = dataset.genre
20 all_genre_combination = all_genre_combination.drop_duplicates()
21
22
23 # genre terpisah yang unik menggunakan pie chart
24 split_genre_dataset = plotModule.split_column_genre(dataset)
25 # count groupby genre
26 count_groupbygenre = split_genre_dataset[['genre', 'title']].groupby('genre').count()
27 plotModule.pieChart('Genre_Title_Movies_Distribution' , count_groupbygenre.index , count_groupbygenre.title)
28
29
30 dataset_combination_count = pd.DataFrame({'genre':[], 'count':[]})
31 for genreNow in all_genre_combination:
32     this_genre_data = dataset[dataset['genre'] == genreNow]
33     count          = this_genre_data.shape[0]
34
35     dataset_combination_count = dataset_combination_count.append({'genre':genreNow , 'count':count}, ignore_index = True)
36
37
38 dataset_combination_count = dataset_combination_count.sort_values('count', ascending = False)
39
40 q2 = dataset_combination_count.quantile(0.5)
41 q1 = dataset_combination_count.quantile(0.25)
42 q3 = dataset_combination_count.quantile(0.75)
43
44 # How Many member in every genre combination
45 plt.title("Genre_Combination_Distribution_Box_and_Whisker")
46 plt.boxplot(dataset_combination_count['count'] , vert=True)
47 plt.show()
48 plt.clf()
49
50
51 # filter dataset
52 filtered_dataset = pd.DataFrame({})
53 for genreNow in dataset_combination_count.genre:
54     countNow = dataset[dataset['genre'] == genreNow].shape[0]
55     if(countNow>2):
56         dataGenreNow = dataset[dataset['genre'] == genreNow]
57         filtered_dataset = filtered_dataset.append(dataGenreNow)
58
59
60 dataset = filtered_dataset
61 # top N revenue tiap kombinasi genre
62 all_genre_combination = dataset.genre
63
64 genre_allRevenueBoxPlotDataFrame = pd.DataFrame({'genre':[], 'revenueRange':[], 'revenueMean':[]})
65 all_genre_combination = all_genre_combination.drop_duplicates()
66 for genreNow in all_genre_combination:
67     this_genre_data = dataset[dataset['genre'] == genreNow]
68
69     this_mean = this_genre_data.revenue.mean()
70     genre_allRevenueBoxPlotDataFrame = genre_allRevenueBoxPlotDataFrame.append({'genre':genreNow,
71                                     'revenueRange': this_genre_data.revenue,
72                                     'revenueMean':this_mean}, ignore_index = True)
73
74 genre_allRevenueBoxPlotDataFrame = genre_allRevenueBoxPlotDataFrame.sort_values('revenueMean', ascending = False)
75
```

```

76| fig,ax = plt.subplots(figsize=(10,30))
77| plt.title('All_Genre_Revenue_Combination_\n')
78| plt.ylabel('Genre_Label')
79| plt.xlabel('Revenue_Distribution_(Million_Dollars)')
80| ax.set_yticklabels(genre_allRevenueBoxPlotDataFrame.genre)
81| ax.boxplot(genre_allRevenueBoxPlotDataFrame.revenueRange, vert=False)
82|
83# top 10 Genre Revenue
84fig,ax = plt.subplots(figsize=(10,10))
85plt.title('Top_10_All_Genre_Revenue_Combination_\n')
86plt.ylabel('Genre_Label')
87plt.xlabel('Revenue_Distribution_(Million_Dollars)')
88ax.set_yticklabels(genre_allRevenueBoxPlotDataFrame.head(10).genre)
89ax.boxplot(genre_allRevenueBoxPlotDataFrame.head(10).revenueRange, vert=False)
90|
91|
92## Top N jumlah genre yang paling banyak dibuat terpisah
93countTitle_Genre = dataset[['genre', 'title']].groupby('genre').count()
94countTitle_Genre = countTitle_Genre.sort_values('title', ascending = False)
95|
96|
97## Semua multi barchart
98## ada dataframe rata rata revenue , rata rata budget , sama selisih rata rata nya
99## bikin yang gede satu sama yang top 10 satu
100all_genre_combination = dataset.genre
101all_genre_combination = all_genre_combination.drop_duplicates()
102profitableDataFrame = pd.DataFrame({'genre':[], 'meanRevenue':[], 'meanBudget':[], 'difference':[]})
103for genreNow in all_genre_combination:
104    this_Genre_Data = dataset[dataset['genre'] == genreNow]
105
106    thisMeanRevenue = np.mean(this_Genre_Data.revenue)
107    thisMeanBudget = np.mean(this_Genre_Data.us_budget)
108
109    difference = thisMeanRevenue - thisMeanBudget
110    profitableDataFrame.append({'genre':genreNow, 'meanRevenue':thisMeanRevenue, 'meanBudget':thisMeanBudget,
111                                'difference':difference}, ignore_index = True)
112
113profitableDataFrame = profitableDataFrame.sort_values('difference' , ascending = False)
114
115topN_profitableDataFrame = profitableDataFrame.head(10)
116barLabels = ['revenue', 'budget']
117Xlabels = topN_profitableDataFrame.genre
118MultiYData = [topN_profitableDataFrame.meanRevenue,topN_profitableDataFrame.meanBudget]
119
120plotModule.multiBarchart(Xlabels , MultiYData, barLabels,'Genre', 'Million_Dollars'
121                           , 'Top_10_Genre_Movie_with_High_profit')
122# =====
123#
124# =====
125## Top 10 all Genre with high Votes
126all_genre_combination = dataset.genre
127all_genre_combination = all_genre_combination.drop_duplicates()
128rankedVotesGenreDataFrame = pd.DataFrame({'genre':[], 'votesRange':[], 'meanVotes':[]})
129for genreNow in all_genre_combination:
130    this_genreData = dataset[dataset['genre'] == genreNow]
131    meanVotes = np.mean(this_genreData.votes)
132    rankedVotesGenreDataFrame.append({'genre':genreNow, 'votesRange':this_genreData.votes, 'meanVotes':
133                                    :meanVotes},ignore_index = True)
134
135
136rankedVotesGenreDataFrame=rankedVotesGenreDataFrame.sort_values('meanVotes' , ascending = False)
137
138fig,ax = plt.subplots(figsize=(10,30))
139plt.title('All_Genre_Votes_Combination_\n')
140plt.ylabel('Genre_Label')
141plt.xlabel('Votes_Distribution')
142ax.set_yticklabels(rankedVotesGenreDataFrame.genre)
143ax.boxplot(rankedVotesGenreDataFrame.votesRange, vert=False)
144
145plt.clf()
146# top 10 Genre Revenue
147fig,ax = plt.subplots(figsize=(10,10))
148plt.title('Top_10_All_Genre_Votes_Combination_\n')
149plt.ylabel('Genre_Label')
150plt.xlabel('Votes_Distribution')
151ax.set_yticklabels(rankedVotesGenreDataFrame.head(10).genre)
152ax.boxplot(rankedVotesGenreDataFrame.head(10).votesRange, vert=False)
153
154
155# US_BUDGET FOR ALL MOVIE TREND
156sumBudget_GroupYear = dataset[['year', 'us_budget']].groupby('year').sum()
157
158plt.bar(sumBudget_GroupYear.index , sumBudget_GroupYear['us_budget'],edgecolor='black')
159plt.xlabel('Year',fontweight='bold')
160plt.ylabel('Us_Budget',fontweight='bold')
161plt.title('Budget_Invested_Year_by_Year' , fontweight='bold')
162#set axis if needed
163#plt.ylim(0,1)
164plt.xticks(rotation=90)
165
166#draw line
167#plt.axhline(y=0.0, color='r' , linestyle='--')
168plt.show()
169plt.clf()
170
171
172
173

```

```

174 ## Bar chart Top Genre Film tiap tahun
175 ## kombinasi genre paling banyak dibuat dari tahun ke tahun
176
177 ## kombinasi genre dengan
178 all_year = dataset.year
179 all_year = all_year.drop_duplicates()
180 mostTitleGenreCombinationPerYearDataFrame = pd.DataFrame({'year':[], 'yeargenre':[], 'count':[]})
181 for yearNow in all_year:
182     this_yearData = dataset[dataset['year'] == yearNow]
183     countTitleByGenre = this_yearData[['genre', 'title']].groupby('genre').count()
184     countTitleByGenre = countTitleByGenre.sort_values('title', ascending= False)
185
186     topGenreThisYear = countTitleByGenre.index[0]
187     topGenreCount = countTitleByGenre.title[0]
188     mostTitleGenreCombinationPerYearDataFrame = mostTitleGenreCombinationPerYearDataFrame.append({'year':yearNow
189         , 'yeargenre':topGenreThisYear+'\n'+str(yearNow), 'count':topGenreCount}, ignore_index = True)
190
191 mostTitleGenreCombinationPerYearDataFrame = mostTitleGenreCombinationPerYearDataFrame.sort_values('year')
192
193
194 plt.figure(figsize=(27,8))
195 plt.bar(mostTitleGenreCombinationPerYearDataFrame['yeargenre'] , mostTitleGenreCombinationPerYearDataFrame['count'], edgecolor='
196     black')
197 plt.xlabel('Year_and_Genre',fontweight='bold')
198 plt.ylabel('How_Many_Movie_Created',fontweight='bold')
199 plt.title('Most_Movie_Created_year_by_year' , fontweight='bold')
200 plt.show()
201 plt.clf()
202
203 ## kombinasi genre dengan rata rata revenue dari tahun ke tahun
204 highestRevenueMeanGenreCombinationPerYearDataFrame = pd.DataFrame({'year':[], 'yeargenre':[], 'meanRevenue':[]})
205 for yearNow in all_year:
206     this_yearData = dataset[dataset['year'] == yearNow]
207     meanRevenueByGenre = this_yearData[['genre', 'revenue']].groupby('genre').mean()
208     meanRevenueByGenre = meanRevenueByGenre.sort_values('revenue', ascending = False)
209
210     topGenreThisYear = meanRevenueByGenre.index[0]
211     topGenreMeanRevenue = meanRevenueByGenre.revenue[0]
212
213     highestRevenueMeanGenreCombinationPerYearDataFrame = highestRevenueMeanGenreCombinationPerYearDataFrame.append({'year':yearNow
214         , 'yeargenre':topGenreThisYear+'\n'+str(yearNow)
215             , 'meanRevenue':
216                 topGenreMeanRevenue
217             },
218             ignore_index = True)
219
220 highestRevenueMeanGenreCombinationPerYearDataFrame = highestRevenueMeanGenreCombinationPerYearDataFrame.sort_values('year')
221
222 plt.figure(figsize=(29,8))
223 plt.bar(highestRevenueMeanGenreCombinationPerYearDataFrame['yeargenre'] ,highestRevenueMeanGenreCombinationPerYearDataFrame['
224     meanRevenue'],edgecolor='black')
225 plt.xlabel('Year_and_Genre',fontweight='bold')
226 plt.ylabel('Mean_Revenue',fontweight='bold')
227 plt.title('Most_High_Mean_Revenue_Genre_Combination_Year_by_Year' , fontweight='bold')
228 plt.show()
229 plt.clf()
230
231 # =====
232 #
233 # =====
234
235 ## kombinasi genre dengan ratarata profit tertinggi dari tahun ke tahun
236 highestProfitMeanGenreCombinationPerYearDataFrame = pd.DataFrame({'year':[], 'yeargenre':[], 'meanProfit':[]})
237 for yearNow in all_year:
238     this_yearData = dataset[dataset['year']== yearNow]
239     meanProfitByGenre = this_yearData[['genre', 'profit']].groupby('genre').mean()
240     meanProfitByGenre = meanProfitByGenre.sort_values('profit', ascending = False)
241
242     topGenreThisYear= meanProfitByGenre.index[0]
243     topGenreMeanProfit = meanProfitByGenre.profit[0]
244
245     highestProfitMeanGenreCombinationPerYearDataFrame = highestProfitMeanGenreCombinationPerYearDataFrame.append({'year':yearNow
246         , 'yeargenre':
247             topGenreThisYear
248             +'\n'+str(
249                 yearNow),
250             'meanProfit':
251                 topGenreMeanProfit
252             },
253             ignore_index = True)
254
255 highestProfitMeanGenreCombinationPerYearDataFrame = highestProfitMeanGenreCombinationPerYearDataFrame.sort_values('year')
256
257 plt.figure(figsize=(29,8))
258 plt.bar(highestProfitMeanGenreCombinationPerYearDataFrame['yeargenre'] ,highestProfitMeanGenreCombinationPerYearDataFrame['
259     meanProfit'],edgecolor='black')
260 plt.xlabel('Year_and_Genre',fontweight='bold')
261 plt.ylabel('Mean_Profit',fontweight='bold')
262 plt.title('Most_High_Mean_Profit_Genre_Combination_Year_by_Year' , fontweight='bold')
263 plt.show()
264 plt.clf()

```

```

254 # =====
255 # ## kombinasi genre dengan ratarata roi tertinggi dari tahun ke tahun
256 # =====
257 highestRoiMeanGenreCombinationPerYearDataFrame = pd.DataFrame({'year':[], 'yeargenre':[], 'meanRoi':[]})
258 for yearNow in all_year:
259     this_yearData = dataset[dataset['year']==yearNow]
260     meanRoiByGenre = this_yearData[['genre', 'roi']].groupby('genre').mean()
261     meanRoiByGenre = meanRoiByGenre.sort_values('roi', ascending = False)
262
263     topGenreThisYear = meanRoiByGenre.index[0]
264     topGenreMeanRoi = meanRoiByGenre.roi[0]
265
266
267     highestRoiMeanGenreCombinationPerYearDataFrame = highestRoiMeanGenreCombinationPerYearDataFrame.append({'year':yearNow,
268                                         'yeargenre':topGenreThisYear+'\n'+str(yearNow),
269                                         'meanRoi':topGenreMeanRoi},
270                                         ignore_index = True)
271
272
273 highestRoiMeanGenreCombinationPerYearDataFrame = highestRoiMeanGenreCombinationPerYearDataFrame.sort_values('year')
274
275 plt.figure(figsize=(29,8))
276 plt.bar(highestRoiMeanGenreCombinationPerYearDataFrame['yeargenre'] ,highestRoiMeanGenreCombinationPerYearDataFrame['meanRoi'],
277          edgecolor='black')
278 plt.xlabel('Year_and_Genre',fontweight='bold')
279 plt.ylabel('Mean_Roi_(%)',fontweight='bold')
280 plt.title('Most_High_Mean_Roi_Genre_Combination_Year_by_Year' , fontweight='bold')
281 plt.show()
282 plt.clf()
283
284 # =====
285 # TOP 3 EVERY YEAR QUANTILE REVENUE
286 # =====
287 # top 3 revenue year by quantile
288 top_3CombinationGenreDataFrame = pd.DataFrame({'year':[], 'genre':[], 'quantileRevenue':[]})
289 for yearNow in all_year:
290     this_yearData = dataset[dataset['year']== yearNow]
291     quantileRevenueByGenre = this_yearData[['genre', 'revenue']].groupby('genre').quantile()
292     quantileRevenueByGenre = quantileRevenueByGenre.sort_values('revenue', ascending = False)
293
294     top1_genre = quantileRevenueByGenre.index[0]
295     top1_revenue = quantileRevenueByGenre.revenue[0]
296
297     top2_genre = quantileRevenueByGenre.index[1]
298     top2_revenue = quantileRevenueByGenre.revenue[1]
299
300     top3_genre = quantileRevenueByGenre.index[2]
301     top3_revenue = quantileRevenueByGenre.revenue[2]
302
303     top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
304                                         , 'genre':top1_genre
305                                         , 'quantileRevenue':top1_revenue},ignore_index = True)
306     top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
307                                         , 'genre':top2_genre
308                                         , 'quantileRevenue':top2_revenue},ignore_index = True)
309
310     top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
311                                         , 'genre':top3_genre
312                                         , 'quantileRevenue':top3_revenue},ignore_index = True)
313
314 fig, ax = plt.subplots()
315 fig.set_figheight(10)
316 fig.set_figwidth(15)
317 plt.xlabel('year')
318 plt.ylabel('quantileRevenue')
319 plt.title("Top_3_Trend_Top_Genre_With_Most_Revenue_")
320 distinct_genre = top_3CombinationGenreDataFrame.genre.drop_duplicates()
321
322 markers = [".", " ", "o", "v", "^", "<", ">", "1", "2",
323             "3", "4", "8", "s", "p", "P", "*", "h", "H",
324             "+", "x", "X", "D", "d"]
325
326 marker_i = 0
327 for genreNow in distinct_genre:
328     n= []
329     dataNow = top_3CombinationGenreDataFrame[top_3CombinationGenreDataFrame['genre']==genreNow]
330     x = np.array(dataNow['year'])
331     y = np.array(dataNow['quantileRevenue'])
332     ax.scatter(x,y, label=genreNow , marker = markers[marker_i], s=150)
333     marker_i = marker_i + 1
334
335     for genre in dataNow.genre:
336         arr_genre = genre.split(',')
337         new_genre = ''
338         for text in arr_genre:
339             new_genre = new_genre + ' ' + text[0:2] + ','
340
341     n.append(new_genre)
342
343 ax.legend(loc='upper_center' , ncol=4, fancybox = True)
344
345 # =====
346 # TOP 3 EVERY YEAR QUANTILE PROFIT
347 # =====
348 # top 3 profit year by quantile
349 top_3CombinationGenreDataFrame = pd.DataFrame({'year':[], 'genre':[], 'quantileProfit':[]})
350 for yearNow in all_year:

```

```

350 this_yearData = dataset[dataset['year']== yearNow]
351 quantileProfitByGenre = this_yearData[['genre', 'profit']].groupby('genre').quantile()
352 quantileProfitByGenre = quantileProfitByGenre.sort_values('profit', ascending = False)
353
354 top1_genre = quantileProfitByGenre.index[0]
355 top1_profit = quantileProfitByGenre.profit[0]
356
357 top2_genre = quantileProfitByGenre.index[1]
358 top2_profit = quantileProfitByGenre.profit[1]
359
360 top3_genre = quantileProfitByGenre.index[2]
361 top3_profit = quantileProfitByGenre.profit[2]
362
363 top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
364                                         , 'genre':top1_genre
365                                         , 'quantileProfit':top1_profit},ignore_index = True)
366 top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
367                                         , 'genre':top2_genre
368                                         , 'quantileProfit':top2_profit},ignore_index = True)
369
370 top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
371                                         , 'genre':top3_genre
372                                         , 'quantileProfit':top3_profit},ignore_index = True)
373
374 fig, ax = plt.subplots()
375 fig.set_figheight(10)
376 fig.set_figwidth(15)
377 plt.xlabel('year')
378 plt.ylabel('quantileProfit')
379 plt.title("Top_3_Trend_Top_Genre_With_Most_Profit")
380 distinct_genre = top_3CombinationGenreDataFrame.genre.drop_duplicates()
381
382 markers = [".", ",", "o", "v", "^", "<", ">", "1", "2",
383             "3", "4", "8", "s", "p", "P", "*", "h", "H",
384             "+", "x", "X", "D", "d", ".", ",", "o", "v", "^", "<"]
385
386 marker_i = 0
387 for genreNow in distinct_genre:
388     n= []
389     dataNow = top_3CombinationGenreDataFrame[top_3CombinationGenreDataFrame['genre']==genreNow]
390     x = np.array(dataNow['year'])
391     y = np.array(dataNow['quantileProfit'])
392     ax.scatter(x,y, label=genreNow , marker = markers[marker_i], s=150, zorder=2)
393     marker_i = marker_i + 1
394
395 for genre in dataNow.genre:
396     arr_genre = genre.split(',')
397     new_genre = ''
398     for text in arr_genre:
399         new_genre = new_genre + ' ' + text[0:2] + ','
400
401     n.append(new_genre)
402
403
404 ax.legend(loc='upper_center' , ncol=4, fancybox = True)
405
406
407 # =====
408 # TOP 3 EVERY YEAR QUANTILE ROI
409 # =====
410
411 top_3CombinationGenreDataFrame = pd.DataFrame({'year':[],'genre':[], 'quantileRoi':[]})
412 for yearNow in all_year:
413     this_yearData = dataset[dataset['year']== yearNow]
414     quantileRoiByGenre = this_yearData[['genre', 'roi']].groupby('genre').quantile()
415     quantileRoiByGenre = quantileRoiByGenre.sort_values('roi', ascending = False)
416
417     top1_genre = quantileRoiByGenre.index[0]
418     top1_roi = quantileRoiByGenre.roi[0]
419
420     top2_genre = quantileRoiByGenre.index[1]
421     top2_roi = quantileRoiByGenre.roi[1]
422
423     top3_genre = quantileRoiByGenre.index[2]
424     top3_roi = quantileRoiByGenre.roi[2]
425
426     top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
427                                         , 'genre':top1_genre
428                                         , 'quantileRoi':top1_roi},ignore_index = True)
429     top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
430                                         , 'genre':top2_genre
431                                         , 'quantileRoi':top2_roi},ignore_index = True)
432
433     top_3CombinationGenreDataFrame =top_3CombinationGenreDataFrame.append({'year':yearNow
434                                         , 'genre':top3_genre
435                                         , 'quantileRoi':top3_roi},ignore_index = True)
436
437
438 fig, ax = plt.subplots()
439 fig.set_figheight(10)
440 fig.set_figwidth(15)
441 plt.xlabel('year')
442 plt.ylabel('quantileProfit')
443 plt.title("Top_3_Trend_Top_Genre_With_Most_Roi")
444 distinct_genre = top_3CombinationGenreDataFrame.genre.drop_duplicates()
445
446 markers = [".", ",", "o", "v", "^", "<", ">", "1", "2",
447             "3", "4", "8", "s", "p", "P", "*", "h", "H",
448             "+"]

```

```

449     "+", "x", "X", "D", "d"]
450
451 marker_i = 0
452 for genreNow in distinct_genre:
453     n= []
454     dataNow = top_3CombinationGenreDataFrame[top_3CombinationGenreDataFrame['genre']==genreNow]
455     x = np.array(dataNow['year'])
456     y = np.array(dataNow['quantileRoi'])
457     ax.scatter(x,y, label=genreNow , marker = markers[marker_i], s=250)
458     marker_i = marker_i + 1
459
460 for genre in dataNow.genre:
461     arr_genre = genre.split(',')
462     new_genre = ''
463     for text in arr_genre:
464         new_genre = new_genre + ' ' + text[0:2] + ','
465
466     n.append(new_genre)
467
468
469 top_3CombinationGenreDataFrame = top_3CombinationGenreDataFrame.sort_values(['genre', 'year'])
470 arr = top_3CombinationGenreDataFrame.index
471 color = ['b', 'g', 'r', 'cyan', 'm', 'y', 'b']
472 coloridx = 0
473 i = 0
474 while(i < len(arr)-1):
475     a_genre = top_3CombinationGenreDataFrame.genre[arr[i]]
476     b_genre = top_3CombinationGenreDataFrame.genre[arr[i+1]]
477
478     a_x = top_3CombinationGenreDataFrame.year[arr[i]]
479     a_y = top_3CombinationGenreDataFrame.quantileRoi[arr[i]]
480
481     b_x = top_3CombinationGenreDataFrame.year[arr[i+1]]
482     b_y = top_3CombinationGenreDataFrame.quantileRoi[arr[i+1]]
483
484     if(a_genre == b_genre and abs(a_x-b_x)==1):
485         ax.plot([a_x,b_x] , [a_y,b_y], color =color[coloridx], zorder=0, linewidth = 3)
486     else:
487         coloridx = coloridx + 1
488         if(coloridx > len(color)-1):
489             coloridx = 0
490
491     i = i+1
492
493 ax.legend(loc='upper_center' , ncol=4, fancybox = True)
495
496
497
498
499 # GenerateAllCombinationGenre Trend Per Year :
500 folder= 'allGenreTrend'
501 if os.path.exists(folder):
502     shutil.rmtree(folder)
503 os.makedirs(folder)
504
505 for genreNow in all_genre_combination:
506     thisGenreDir = folder + '\\\\' + genreNow
507     if os.path.exists(thisGenreDir):
508         shutil.rmtree(thisGenreDir)
509     os.makedirs(thisGenreDir)
510     thisGenreDataNow = dataset[dataset['genre'] == genreNow]
511     thisGenreDataNow = thisGenreDataNow.sort_values('year')
512     # get all Year
513     allYear = thisGenreDataNow.year
514     allYear = allYear.drop_duplicates()
515
516
517 thisGenreSumAllFeatureDataFrame = pd.DataFrame({'year':[], 'sumRevenue':[], 'sumRoi':[], 'sumProfit':[], 'sumBudget':[] })
518 for year in allYear:
519     thisYearDataNow = thisGenreDataNow[thisGenreDataNow['year'] == year]
520
521     sumBudget = thisYearDataNow.us_budget.quantile()
522     sumRevenue = thisYearDataNow.revenue.quantile()
523     sumProfit = thisYearDataNow.profit.quantile()
524     sumRoi = thisYearDataNow.roi.sum()
525
526     thisGenreSumAllFeatureDataFrame= thisGenreSumAllFeatureDataFrame.append({'year':year
527                                         , 'sumRevenue':sumRevenue
528                                         , 'sumRoi':sumRoi
529                                         , 'sumProfit':sumProfit
530                                         , 'sumBudget':sumBudget }, ignore_index = True)
531
532 thisGenreFileDir = thisGenreDir + '\\\\' + 'yearTrendAnalysis.jpg'
533 thisGenreDataFrameDir = thisGenreDir + '\\\\' + genreNow+'DataFrame.csv'
534 multiYLabel = ['Revenue', 'Profit']
535 multiYData = [
536     thisGenreSumAllFeatureDataFrame['sumRevenue']
537     ,thisGenreSumAllFeatureDataFrame['sumProfit']]
538 plotModule.saveMultiBarchart(thisGenreSumAllFeatureDataFrame['year'],
539                               multiYData , multiYLabel , 'Year' , 'Sum', genreNow+'_YearTrendAnalysis', thisGenreFileDir)
540
541 thisGenreSumAllFeatureDataFrame.to_csv(thisGenreDataFrameDir)
542

```

Listing A.2: budget-IMDB-Merger.py

```

2 """
3 Created on Mon Mar  2 08:52:42 2020
4
5 @author: Teuku Hashrul
6 """
7
8 import pandas as pd
9
10
11 #read the original cleaned dataset
12 dataset=pd.read_csv('hasileksperimen2.csv')
13 budget_imdbapi = pd.read_excel('scrappedBudgetFromIMDB-API_cleaned.xlsx')
14 merged_dataset = pd.merge(dataset , budget_imdbapi , how='inner' , on='title')
15 merged_dataset.to_csv('hasileksperimen-withBudget.csv')

```

Listing A.3: clusterActorAnalysis.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Feb 28 07:50:15 2020
4
5 @author: lenovo
6 """
7
8
9 import shutil
10 import pandas as pd
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pickle
14 from sklearn.cluster import AgglomerativeClustering, KMeans
15 import os
16 from sklearn.metrics.pairwise import cosine_similarity , cosine_distances
17 import sys
18 np.set_printoptions(threshold=sys.maxsize)
19 import plotModule
20 from scipy.stats.stats import pearsonr
21 # =====
22 # #import plotModuleFromAnotherDirectory (minggu 3)
23 # # some_file.py
24 # import sys
25 # # insert at 1, 0 is the script path (or '' in REPL)
26 # sys.path.insert(1, 'D:\backup\Semester 8\Skripsi 2\Minggu 3')
27 # =====
28
29 # ReadTheClusteredActorDataset
30 dataset = pd.read_csv('clusteredactor200_dataset.csv')
31
32 #count every label from the clustered data
33 countEveryLabelMember = dataset[['label','title']].groupby('label').count()
34
35 countEveryLabelMember = countEveryLabelMember.sort_values('title' , ascending=False)
36 #visualize with barchart
37 # =====
38 # plotModule.barcharth(countEveryLabelMember.index, countEveryLabelMember.title , 'label' , 'count',
39 #                         '200 n Agglomerative Clustered Label Member Distribution' , '200_NAggoClusteredMemberLabelDistribution.jpg'
40 #                         )
41 # =====
42 # ambil 2 kelompok yang anggota actornya mirip dari data 200 clustered actor
43 possible1 = dataset[dataset['label'] == 1]
44 possible2 = dataset[dataset['label'] == 21]
45
46
47
48 # ambil salah kelompok
49 split_genre = plotModule.split_column_actor(possible1)
50 genreonly = split_genre['actors']
51 genreonly = genreonly.astype(str)
52 np_genre = np.array(genreonly).astype(str)
53 text_1= " ".join(np_genre).lower()
54
55 split_genre = plotModule.split_column_actor(possible2)
56 genreonly = split_genre['actors']
57 genreonly = genreonly.astype(str)
58 np_genre = np.array(genreonly).astype(str)
59 text_2= " ".join(np_genre).lower()
60
61 # visualize clustered actor
62 plotModule.generateWordCloud(text_1 , 'Label_1' , 'Actor_that_featured_in_the_cluster_1_-IntraCluster_=0.58')
63 plotModule.generateWordCloud(text_2 , 'label_21' , 'Actor_that_featured_in_the_cluster_21_-IntraCluster_=0.51')
64
65 # visualize the revenue range if we hire the those actor
66 plotModule.boxplot(possible1['revenue'] ,False , 'Cluster-1_revenue_range' , 'first200cluster-Label1RevenueRange.jpg')
67 plotModule.boxplot(possible2['revenue'] ,False , 'Cluster-21_revenue_range' , 'first200cluster-Label21RevenueRange')
68
69 countEveryLabelMember['label_2'] = countEveryLabelMember.index
70 countEveryLabelMember = countEveryLabelMember.rename(columns={"title":"count"})
71
72
73 # =====
74 # Percobaan buang outlier
75 # =====
76
77 #take the cluster that have more than one film / removing possible outlier
78 removedOutlier = countEveryLabelMember[countEveryLabelMember['count']> 1]
79
80 #join the Data with the count

```

```

81| data_removedOutlier = pd.merge(dataset , removedOutlier, how='right', left_on = 'label' , right_on = 'label_2')
82|
83# copy removed possible outlier
84secondCluster = data_removedOutlier
85# removed clusterLabel because we want to cluster again
86del secondCluster['label']
87del secondCluster['intraclusterdistance']
88del secondCluster['count']
89del secondCluster['label_2']
90|
91|
92#split by , to make actors
93splitted_actors = (secondCluster.set_index(secondCluster.columns.drop('actors',1).tolist()))
94    .actors.str.split(',', expand=True)
95    .stack()
96    .reset_index()
97    .rename(columns={0:'actors'})
98    .loc[:, secondCluster.columns]
99    )
100#
101#preprocessing removing front space
102splitted_actors['actors'] = splitted_actors['actors'].str.lstrip()
103#preprocessing lowercase
104splitted_actors['actors'] = splitted_actors['actors'].str.lower()
105#
106# to convert actor into one hot style
107onehot_actors = pd.crosstab(splitted_actors['title'],splitted_actors['actors']).rename_axis(None, axis=1).add_prefix('`')
108#
109merged_inner = pd.merge(left=secondCluster,right=onehot_actors, left_on='title', right_on='title')
110#
111cluster = pd.merge(left=dataset[['title' , 'revenue']],right=onehot_actors, left_on='title', right_on='title')
112#
113merged_inner.to_csv('IMDBMOVIE_withonehotactor.csv')
114cluster_del = cluster
115del cluster_del['title']
116del cluster_del['revenue']
117#
118# fulldata : dataset with all the column
119# features : column used from the fullDataset (sub dataframe from full Data)
120# methods : 'kmeans' or 'agglo' or any other methods
121def generateCluster(fullDataset,features , method , n, titlefeatures):
122    # iterate every label
123    allLabelMean = pd.DataFrame({"n":+ str(n): []})
124    # create dataframe consist of label and the meancluster distance to join with the real dataset
125    labelMeanDistance = pd.DataFrame({'label':[],'intraclusterdistance':[]})
126    if method == 'kmeans':
127        kmeans = KMeans(n_clusters = n)
128        kmeans.fit(features)
129        cluster_del['label'] = kmeans.labels_
130        fullDataset['label'] = kmeans.labels_
131        centroid = pd.DataFrame(kmeans.cluster_centers_)
132        #remove label for countable
133        centroid = centroid.iloc[:, :-1]
134        for index in range(0,n):
135            # take the only label in idx:
136            rowLabelNow = cluster_del[cluster_del['label'] == index]
137            del rowLabelNow['label']
138            #locate the centroid
139            centroidNow = centroid.iloc[index]
140
141            #count all the euclidean distance between every row in row labelnow and the centroidNow and put it in the label Now
142            dist = (rowLabelNow - np.array(centroidNow)).pow(2).sum(1).pow(0.5)
143            meanDistNow = np.mean(dist)
144
145            allLabelMean = allLabelMean.append({"n":+str(n):meanDistNow}, ignore_index = True)
146            labelMeanDistance = labelMeanDistance.append({'label':index,'intraclusterdistance':meanDistNow}, ignore_index = True)
147    elif method == 'agglo':
148        agglo = AgglomerativeClustering(n_clusters = n , affinity='cosine', linkage='average')
149        agglo.fit(features)
150
151        cluster_del['label'] = agglo.labels_
152        fullDataset['label'] = agglo.labels_
153
154        print('experiment_` + str(n))
155        for index in range(0,n):
156            # take the only label in idx:
157            rowLabelNow = cluster_del[cluster_del['label'] == index]
158            del rowLabelNow['label']
159            #create centroid from mean of all the member of the cluster
160
161
162            centroidNow = np.mean(rowLabelNow)
163
164            if index == 8:
165                print(rowLabelNow)
166                print(fullDataset[fullDataset['label'] == 8])
167
168            #count all the cosine similarity distance between every row in row labelnow and the centroidNow and put it in the
169            #label Now
170            #dist = (rowLabelNow - np.array(centroidNow)).pow(2).sum(1).pow(0.5)
171
172            dist = 0
173            lenn = 0
174            for idx, row in rowLabelNow.iterrows():
175                cosine = cosine_similarity([row],[np.array(centroidNow)])
176                dist = dist + cosine[0][0]
177                lenn = lenn + 1
178            meanDistNow = dist / lenn

```

```

179
180     allLabelMean = allLabelMean.append({"n":str(n):meanDistNow}, ignore_index = True)
181     print('label:' + str(index) + '_mean_distance:' + str(meanDistNow))
182     labelMeanDistance = labelMeanDistance.append({'label':index,'intraclusterdistance':meanDistNow}, ignore_index = True)
183
184 #leftouter join labelMeanDistance and the real dataset clustered
185 dataset_clusteredmean = pd.merge(labelMeanDistance,fullDataset,how='left', on = ['label'])
186
187 folder = ''
188 if method == 'kmeans':
189     folder = 'kmeans\\kmeans_n'+str(n)+'\\'
190 elif method == 'agglo':
191     folder = 'agglo\\agglo_n' +str(n)+ '\\'
192
193 if os.path.exists(folder):
194     shutil.rmtree(folder)
195 os.makedirs(folder)
196 #save dataset with label and the mean distance
197 dataset_clusteredmean.to_csv(folder+'clustered'+titlefeatures+str(n) +'_dataset.csv')
198 # save model using pickle
199 model = None
200 if method == 'kmeans':
201     model = kmeans
202 elif method == 'agglo':
203     model = agglo
204 pickle.dump(model, open(folder+'model'+ '_clustered' +str(n)+titlefeatures, 'wb'))
205 return allLabelMean
206
207
208 # =====
209 # second actor cluster attempt
210 # =====
211
212
213 #also need to count timer
214 import timeit
215 start = timeit.default_timer()
216 #switchable 'kmeans' or 'agglo'
217 method = 'agglo'
218 allExperimentMean = pd.DataFrame({'n':[],'meanintracluster':[]})
219 experiment =[100,150 ,200 , 250 , 300]
220 for n_experiment in experiment:
221     allLabelMean = generateCluster(secondCluster,cluster_del ,method ,n_experiment, 'actorsecondattempt')
222     allExperimentMean = allExperimentMean.append({'n':n_experiment, 'meanintracluster':[allLabelMean['n']+str(n_experiment)]}), ignore_index=True)
223
224 stop = timeit.default_timer()
225 estimatedTime = stop - start
226
227 fig,ax = plt.subplots()
228 plt.title(method+'_intracluster_mean_distance_comparison_\nusing_box_and_whisker_actor_second_attempt')
229 plt.ylabel('how_many_cluster')
230 plt.xlabel('intracluster_mean_distribution_for_every_label')
231 ax.set_yticklabels(allExperimentMean['n'])
232 ax.boxplot(allExperimentMean['meanintracluster'] , vert=False)
233
234 # =====
235 # March ,1 experiment only run the library
236 # =====
237
238
239 # since we choose 200 as a best optimum cluster actor
240 # read the generated data
241 actorClusteredFixedDataset = pd.read_csv("agglo/agglo_n200/clusteredactorssecondattempt200_dataset.csv")
242
243 # get label only
244 labelOnly = actorClusteredFixedDataset[['label']]
245
246 # distinct label for looping purpose
247 labelOnly = labelOnly.drop_duplicates()
248
249 # HYPOTHESIS 1 . create dataframe for actor representatives hypothesis
250 actorContributionPerCluster = pd.DataFrame({'actorcluster':[], "clustercontribution":[]})
251
252 # HYPOTHESIS 1.1 : there are multiple actor representatives
253 multipleActorRepresentativesPerCluster = pd.DataFrame({'cluster_actor_label':[], "howmanyactor":[] , "actornames":[]})
254
255
256 #HYPOTHESIS 2. create dataframe -> every actor representatives have their own genre specialty
257 actorGenreMostPerCluster = pd.DataFrame({'genrecluster':[], "clustercontribution":[]})
258
259
260 #HYPOTHESIS 3. create dataframe to count their favourite genre revenue contribution
261 actorFavGenreRevenuePerCluster = pd.DataFrame({'revenueFavGenre':[], "clustercontribution":[]})
262
263
264 #HYPOTHESIS 3.2 : CREATE DATAFRAME THAT CONSISTENT OR NOT
265 actorFavGenre_RevenueConsistencyPerCluster = pd.DataFrame({'cluster_actor_label':[],'actor':[],'favgenre':[], 'favgenrecontribution':[], 'toprevenuegenre':[] , 'toprevenuecontribution':[] , 'status':[]})
266
267
268
269
270 #HIPOTHESES PROFIT 1 : PROFIT FOR EVERY GENRE CONTRIBUTE TO ABOVE 50 PERCENT
271 actorFavGenre_ProfitConsistencyPerCluster = pd.DataFrame({'cluster_actor_label':[] , 'actor':[] , 'favgenre':[], 'favgenrecontribution':[]})
272
273 # HIPOTHESES ROI 1 : ROI FOR EVERY GENRE FAVOURITE CONTRIBUTE TO ABOVE 50 PERCENT

```

```

275 actorFavGenre_RoiConsistencyPerCluster = pd.DataFrame({'cluster_actor_label':[], 'actor':[], 'favgenre':[], 'favgenrecontribution':[]})
276
277 ##Barchart every cluster mean revenue
278 youtubeMeanView_PerCluster = pd.DataFrame({'cluster_actor_label':[], 'mean_youtube_view':[]})
279
280 #Barchart every cluster pearson revenue profit roi
281 youtubeViewPearson_perCluster = pd.DataFrame({'cluster_actor_label':[], 'pearson_revenue':[],'pearson_profit':[],'pearson_roi':[]})
282
283 #Barchart every cluster actor hashtag count pearson
284 instagramHashtagPearson_perCluster = pd.DataFrame({'cluster_actor_label':[], 'pearson_revenue':[],'pearson_profit':[],'pearson_roi':[]})
285
286
287
288 # create super directory for saving the actor experiment
289 folderName = 'actorClusterAnalysis'
290 if os.path.exists(folderName):
291     shutil.rmtree(folderName)
292 os.makedirs(folderName)
293 for index,label in labelOnly.iterrows():
294     #get label now
295     labelNow = label[0]
296     print(labelNow)
297
298 # create directory for this label for every purpose
299 thisLabelFolderName = 'actor-Label-' + str(labelNow)
300 thisLabelDirPath = folderName+'\\'+thisLabelFolderName
301 if os.path.exists(thisLabelDirPath):
302     shutil.rmtree(thisLabelDirPath)
303 os.makedirs(thisLabelDirPath)
304
305 #select the data for label Now
306 dataLabelNow = actorClusteredFixedDataset[actorClusteredFixedDataset['label'] == labelNow]
307 #save the data
308 name_DataLabelNow = 'actor-'+label+'_'+str(labelNow)+'.dataset.csv'
309 dataLabelNow.to_csv(thisLabelDirPath+'\\'+name_DataLabelNow)
310
311 #generate WordCloud Actor from labelNow
312 split_column_actor = plotModule.split_column_actor(dataLabelNow)
313 #trim actor cause there are 'kirsten dunst' and 'kirsten dunst'
314 split_column_actor['actors'] = split_column_actor['actors'].str.strip()
315
316 actoronly = split_column_actor['actors']
317 actoronly = actoronly.astype(str)
318 np_actor = np.array(actoronly).astype(str)
319 text_actor= " ".join(np_actor).lower()
320
321 wordCloudFileName = thisLabelDirPath + '\\wordcloud-actorlabel-' +str(labelNow) + '.jpg'
322 plotModule.generateWordCloud(text_actor , wordCloudFileName , 'Actor_Distribution_for_Label_cluster-' + str(labelNow))
323
324
325
326 #boxPlot revenue for this cluster revenue estimation
327 boxplot_revenueName = thisLabelDirPath + '\\boxandwhiskerRevenue-actorlabel-' +str(labelNow) + '.jpg'
328 plotModule.boxplot(dataLabelNow['revenue'],False , 'Box_and_Whisker_Revenue_Distribution_actor_cluster-' +str(labelNow) , boxplot_revenueName)
329
330 #boxplot rating for this cluster rating estimation
331 boxplot_ratingName = thisLabelDirPath + '\\boxandwhiskerRating-actorlabel-' +str(labelNow) + '.jpg'
332 plotModule.boxplot(dataLabelNow['rating'],False , 'Box_and_Whisker_Rating_Distribution_actor_cluster-' +str(labelNow) , boxplot_ratingName)
333
334 #boxPlot profit for this cluster estimation
335 boxplot_profitName = thisLabelDirPath + '\\boxandwhisker-Profit-actorlabel' +str(labelNow) + '.jpg'
336 plotModule.boxplot(dataLabelNow['profit'],False , 'Box_and_Whisker_Profit_Distribution_actor_cluster-' +str(labelNow) , boxplot_profitName)
337
338 #boxplot roi for this cluster estimation
339 boxplot_roiName = thisLabelDirPath + '\\boxandwhisker-Roi-actorlabel' +str(labelNow) + '.jpg'
340 plotModule.boxplot(dataLabelNow['roi'], False , 'Box_and_Whisker_Roi_Distribution_actor_cluster-' +str(labelNow) , boxplot_roiName)
341
342 #boxplot youtubeview for this cluster estimation
343 boxplot_youtubeViewName = thisLabelDirPath + '\\boxandwhisker-YoutubeView-actorlabel' +str(labelNow) + '.jpg'
344 plotModule.boxplot(dataLabelNow['viewCount'], False , 'Box_and_Whisker_YoutubeView_Distribution_actor_cluster-' +str(labelNow) , boxplot_youtubeViewName)
345
346 #boxplot instagramhashtagtitle count for this cluster estimation
347 boxplot_instagramhashtagCountName = thisLabelDirPath + '\\boxandwhisker-InstagramHashtagCount-actorlabel' +str(labelNow) + '.jpg'
348 plotModule.boxplot(dataLabelNow['hashtagcount'], False , 'Box_and_Whisker_Instagram_Hashtag_Count_Distribution_actor_cluster-' +str(labelNow) , boxplot_youtubeViewName)
349
350
351
352
353
354 split_genre_actor = plotModule.split_column_genre(dataLabelNow)
355 #bar chart for every genre distribution
356 genre_revenue = split_genre_actor[['genre', 'revenue']]
357 meanRevenuePerGenre = genre_revenue.groupby('genre').mean()
358 meanRevenuePerGenre = meanRevenuePerGenre.sort_values('revenue',ascending=True)
359 #path to save and visualize
360 barchartPerGenreRevenueMean_Name = thisLabelDirPath + '\\barchartMeanGenreRevenue-actorlabel-' +str(labelNow) + '.jpg'
361 plotModule.barcharth(meanRevenuePerGenre.index,meanRevenuePerGenre.revenue,'Genre','Mean_Revenue','Barchart_Mean_Revenue_Per_Genre_Distribution_actor_cluster-' +str(labelNow),barchartPerGenreRevenueMean_Name)
362
363

```

```

364 #bar chart for every rating distribution
365 genre_rating = split_genre_actor[['genre', 'rating']]
366 meanRatingPerGenre = genre_rating.groupby('genre').mean()
367 meanRatingPerGenre = meanRatingPerGenre.sort_values('rating', ascending=True)
368 #path to save and visualize
369 barchartPerGenreRatingMean_Name = thisLabelDirPath + '\\barchartMeanGenreRating-actorlabel-' + str(labelNow) + '.jpg'
370 plotModule.barcharth(meanRatingPerGenre.index, meanRatingPerGenre.rating, 'Genre', 'Mean_Rating', 'Bchart_Mean_Rating_Per_
371   Genre_Distribution_actor_cluster-' + str(labelNow), barchartPerGenreRatingMean_Name)
372
373
374 # count every actor play how many title
375 actor_title = split_column_actor[['actors', 'title']]
376 countAllActorInThisCluster = actor_title.groupby('actors').count()
377 # howManyFilmActorPlaysInCluster / howManyTitlesInTheCluster * 100
378 countAllActorInThisCluster['contribution'] = countAllActorInThisCluster['title'] / dataLabelNow.shape[0] * 100
379 actorContributionDataFrameName = thisLabelDirPath + '\\cluster-' + str(labelNow) + '-actorTitleContribution.csv'
380 #saveThisClusterActorContribution
381 countAllActorInThisCluster.to_csv(actorContributionDataFrameName)
382 #sort to get most contributing play in this cluster
383 countAllActorInThisCluster = countAllActorInThisCluster.sort_values('contribution', ascending=False)
384 mostPlayedActor = countAllActorInThisCluster.index[0] + '-' + str(labelNow)
385 mostPlayedContribution = countAllActorInThisCluster.contribution[0]
386
387 #add to The Global DataFrame
388 actorContributionPerCluster = actorContributionPerCluster.append({'actorcluster':mostPlayedActor, "clustercontribution":_
389   mostPlayedContribution}, ignore_index=True)
390
391 #count every movie
392 genre_title = split_genre_actor[['genre', 'title']]
393 countAllGenreInThisCluster = genre_title.groupby('genre').count()
394 # howManyGenreTitle / allGenreTitle * 100
395 countAllGenreInThisCluster['contribution'] = countAllGenreInThisCluster['title'] / dataLabelNow.shape[0]*100
396 # sort by contribution descending to determine the biggest actor
397 countAllGenreInThisCluster = countAllGenreInThisCluster.sort_values('contribution', ascending = False )
398 genreContributionDataFrameName = thisLabelDirPath + '\\cluster-' + str(labelNow) + '-genreManyTitleContribution.csv'
399 countAllGenreInThisCluster.to_csv(genreContributionDataFrameName)
400 mostPlayedGenreByActor = mostPlayedActor + '-' + countAllGenreInThisCluster.index[0] + '-' + str(labelNow)
401 mostPlayedGenreContribution = countAllActorInThisCluster.contribution[0]
402 actorGenreMostPerCluster = actorGenreMostPerCluster.append({'genrecluster':mostPlayedGenreByActor , "clustercontribution":_
403   mostPlayedGenreContribution}, ignore_index = True)
404
405 # HYPOTHESIS NUMBER 3
406 genre_revenue= split_genre_actor[['genre', 'revenue']]
407 sumAllRevenuePerGenreInThisCluster = genre_revenue.groupby('genre').sum()
408 sumAllRevenuePerGenreInThisCluster['contribution'] = sumAllRevenuePerGenreInThisCluster['revenue'] / dataLabelNow['revenue']._
409   sum() * 100
410 sumAllRevenuePerGenreInThisCluster = sumAllRevenuePerGenreInThisCluster.sort_values('contribution', ascending=False)
411 sumActorFavGenreDataName = thisLabelDirPath + '\\cluster-' + str(labelNow) + '-actorFavGenreRevenue.csv'
412 sumAllRevenuePerGenreInThisCluster.to_csv(sumActorFavGenreDataName)
413 #take the fav genre from hypothesis number two
414 favGenreKey = countAllGenreInThisCluster.index[0]
415 favGenreRevenueContribution = sumAllRevenuePerGenreInThisCluster.contribution[favGenreKey]
416 actorFavGenreRevenueName = mostPlayedActor+favGenreKey
417 actorFavGenreRevenuePerCluster = actorFavGenreRevenuePerCluster.append({'revenueFavGenre':actorFavGenreRevenueName, "
418   clustercontribution":favGenreRevenueContribution}, ignore_index= True)
419
420 #HYPOTHESIS NUMBER 1.2 MULTIPLE ACTORS REPRESENTATIVES
421 aboveContributionActorsCount = 0
422 aboveContributionActorNames = ""
423 #using hypothesis 1 aggregation how many actor contribute in title
424 multipleAbove50Data = countAllActorInThisCluster[countAllActorInThisCluster['contribution'] >= 50]
425 for index, row in multipleAbove50Data.iterrows():
426     #for first index
427     if aboveContributionActorsCount == 0:
428         aboveContributionActorNames = aboveContributionActorNames + index
429     else:
430         aboveContributionActorNames = aboveContributionActorNames + ',' + index
431     aboveContributionActorsCount = aboveContributionActorsCount + 1
432
433 multipleActorRepresentativesPerCluster = multipleActorRepresentativesPerCluster.append({"cluster_actor_label":labelNow, "_
434   howmanyactor":aboveContributionActorsCount , "actornames":aboveContributionActorNames}, ignore_index= True)
435
436 #HYPOTHESIS NUMBER 3.2 : CONSISTENT OR NOT
437 isConsistent = ''
438 # fav genre same or not and
439 if sumAllRevenuePerGenreInThisCluster.contribution[favGenreKey] > 50:
440     isConsistent = 'consistent'
441 else:
442     isConsistent = 'not_consistent'
443
444 mostPlayedActor = mostPlayedActor
445 favGenreKey = countAllGenreInThisCluster.index[0]
446 favGenreContribution = countAllActorInThisCluster.contribution[0]
447
448 topRevenueGenre = sumAllRevenuePerGenreInThisCluster.index[0]
449 topRevenueContribution = sumAllRevenuePerGenreInThisCluster.contribution[0]
450
451 actorFavGenre_RevenueConsistencyPerCluster = actorFavGenre_RevenueConsistencyPerCluster.append({'cluster_actor_label':labelNow
452   , 'actor':mostPlayedActor,'favgenre':favGenreKey, 'favgenrecontribution':favGenreContribution, 'toprevenuegenre':_
453   topRevenueGenre , 'toprevenuecontribution':topRevenueContribution , 'status':isConsistent}, ignore_index = True)
454
455 # HYPOTHESIS PROFIT : FAVOURITE GENRE contribute more than 50 percent
456 genre_profit = split_genre_actor[['genre', 'profit']]
457 sumAllProfitPerGenreInThisCluster = genre_profit.groupby('genre').sum()

```

```

455 sumAllProfitPerGenreInThisCluster['contribution'] = sumAllProfitPerGenreInThisCluster['profit'] / dataLabelNow['profit'].sum()
456 * 100
457 favGenreKey = favGenreKey
458 favGenreProfitContribution = sumAllProfitPerGenreInThisCluster.contribution[favGenreKey]
459 actorFavGenre_ProfitConsistencyPerCluster = actorFavGenre_ProfitConsistencyPerCluster.append({'cluster_actor_label':labelNow,
460 'actor':mostPlayedActor, 'favgenre':favGenreKey, 'favgenrerecontribution':favGenreProfitContribution}, ignore_index =
461 True)
462
463 # HYPOTHESIS ROI :
464 genre_roi = split_genre_actor[['genre', 'roi']]
465 sumAllRoiPerGenreInThisCluster = genre_roi.groupby('genre').sum()
466 sumAllRoiPerGenreInThisCluster['contribution'] = sumAllRoiPerGenreInThisCluster['roi'] / dataLabelNow['roi'].sum() * 100
467 favGenreRoiContribution = sumAllRoiPerGenreInThisCluster.contribution[favGenreKey]
468 actorFavGenre_RoiConsistencyPerCluster = actorFavGenre_RoiConsistencyPerCluster.append({'cluster_actor_label':labelNow, 'actor':
469 ':mostPlayedActor, 'favgenre':favGenreKey, 'favgenrerecontribution':favGenreRoiContribution}, ignore_index = True)
470
471 # BARCHART PEARSON YOUTUBEVIEW and InstagramHashtagCount
472 if dataLabelNow.shape[0] > 1:
473     pearson_youtubecount_roi = pearsonr(dataLabelNow['roi'], dataLabelNow['viewCount'])[0]
474     pearson_youtubecount_revenue = pearsonr(dataLabelNow['revenue'], dataLabelNow['viewCount'])[0]
475     pearson_youtubecount_profit = pearsonr(dataLabelNow['profit'], dataLabelNow['viewCount'])[0]
476
477     pearson_instagramcounthashtag_roi = pearsonr(dataLabelNow['roi'], dataLabelNow['hashtagcount'])[0]
478     pearson_instagramcounthashtag_revenue = pearsonr(dataLabelNow['revenue'], dataLabelNow['hashtagcount'])[0]
479     pearson_instagramcounthashtag_profit = pearsonr(dataLabelNow['profit'], dataLabelNow['hashtagcount'])[0]
480
481     youtubeViewPearson_perCluster = youtubeViewPearson_perCluster.append({'cluster_actor_label':mostPlayedActor, 'pearson_revenue':pearson_youtubecount_revenue,
482 , 'pearson_profit':pearson_youtubecount_profit, 'pearson_roi':pearson_youtubecount_roi}, ignore_index =True)
483
484     instagramHashtagPearson_perCluster =instagramHashtagPearson_perCluster.append({'cluster_actor_label':mostPlayedActor, 'pearson_revenue':pearson_instagramcounthashtag_revenue,
485 , 'pearson_profit':pearson_instagramcounthashtag_profit, 'pearson_roi':pearson_instagramcounthashtag_roi}, ignore_index = True)
486
487
488 # saved all The most Played actor representatives
489 actorContributionPerCluster.to_csv('allClusterActor-mostPlayed.csv')
490 actorContributionPerCluster = actorContributionPerCluster.sort_values('clustercontribution', ascending = False)
491 #visualize
492 plotModule.barcharth(actorContributionPerCluster['actorcluster'], actorContributionPerCluster['clustercontribution'], 'actor_
493 representatives', 'contribution_(Percent)', 'Bchart_Every_Cluster_Most_Played_Actor', 'bchartMostPlayedActor.jpg')
494
495 biggerThan50 = actorContributionPerCluster[actorContributionPerCluster['clustercontribution'] >= 50]
496 lowerThan50 = actorContributionPerCluster[actorContributionPerCluster['clustercontribution'] < 50]
497
498 pieChartActorTitleContribution = pd.DataFrame({'label':[], 'count':[]})
499 pieChartActorTitleContribution = pieChartActorTitleContribution.append({'label':'clusterWithActorThat_\n_PlayedMoreThan50Percent',
500 'count':biggerThan50.shape[0]}, ignore_index=True)
501 pieChartActorTitleContribution = pieChartActorTitleContribution.append({'label':'clusterWithActorThat_\n_PlayedLessThan50Percent',
502 'count':lowerThan50.shape[0]}, ignore_index=True)
503 # plot the Distribution for comparison
504 plotModule.pieChart('Actor_Representative_Cluster_Comparison',pieChartActorTitleContribution['label'],
505 pieChartActorTitleContribution['count'])
506
507 #HYPOTHEISNUMBERTWO : EVERY ACTOR HAVE THEIR OWN FAVOURITE GENRE
508 actorGenreMostPerCluster.to_csv('allClusterActor-mostFavGenre.csv')
509 actorGenreMostPerCluster = actorGenreMostPerCluster.sort_values('clustercontribution', ascending = False)
510 #visualize
511 plotModule.barcharth(actorGenreMostPerCluster['genreccluster'], actorGenreMostPerCluster['clustercontribution'], 'Most_Favourite_
512 actor_genre', 'genre_contribution_in_cluster_(percent)', 'Bchart_Every_Cluster_Favourite_Genre', 'bchartMostFavGenreInCluster.jpg')
513
514 biggerGenreThan50 = actorGenreMostPerCluster[actorGenreMostPerCluster['clustercontribution'] >= 50]
515 lowerGenreThan50 = actorGenreMostPerCluster[actorGenreMostPerCluster['clustercontribution'] < 50]
516
517 pieChartFavGenreTitleContribution = pd.DataFrame({'label':[], 'count':[]})
518 pieChartFavGenreTitleContribution = pieChartFavGenreTitleContribution.append({'label':'cluster_with_most_played_genre_\n_more_than
519 .50_percent', 'count':biggerGenreThan50.shape[0]}, ignore_index = True)
520 pieChartFavGenreTitleContribution = pieChartFavGenreTitleContribution.append({'label':'cluster_with_most_played_genre_\n_less_than
521 .50_percent', 'count':lowerGenreThan50.shape[0]}, ignore_index=True)
522 plotModule.pieChart('Actor_Possible_Favourite_Genre_\n_Contribution_perCluster_Comparison',pieChartFavGenreTitleContribution['
523 label'], pieChartFavGenreTitleContribution['count'])
524
525 #HYPOTHEISNUMBERTHREE : DO THEIR FAVOURITE GENRE MOST CONTRIBUTED TO THEIR REVENUE
526 actorFavGenreRevenuePerCluster.to_csv('allClusterActor-FavGenreRevenue.csv')
527 actorFavGenreRevenuePerCluster = actorFavGenreRevenuePerCluster.sort_values('clustercontribution', ascending = False)
528 plotModule.barcharth(actorFavGenreRevenuePerCluster['revenueFavGenre'],actorFavGenreRevenuePerCluster['clustercontribution'], 'cluster-
529 -actor-Favgenre', 'Revenue_Contribution_\n_Per_Cluster_(Percent)', 'BarChart_Every_Cluster_Most_Fav_Genre_\n_Revenue_
530 _Contribution', 'bchartFavGenreRevenueInCluster.jpg')
531 pieChartFavGenreRevenueContribution = pd.DataFrame({'label':[], 'count':[]})
532 biggerRevenueThan50 = actorFavGenreRevenuePerCluster[actorFavGenreRevenuePerCluster['clustercontribution'] >= 50]
533 lowerRevenueThan50 = actorFavGenreRevenuePerCluster[actorFavGenreRevenuePerCluster['clustercontribution'] < 50]
534
535 pieChartFavGenreRevenueContribution = pieChartFavGenreRevenueContribution.append({'label':'cluster_with_favourite_genre_
536 _contribution_\n_more_than_50_percent', 'count':biggerRevenueThan50.shape[0]}, ignore_index=True)
537 pieChartFavGenreRevenueContribution = pieChartFavGenreRevenueContribution.append({'label':'cluster_with_favourite_genre_

```

```

533     contribution_\n_less_than_50_percent', 'count':lowerRevenueThan50.shape[0]} , ignore_index=True),
534     pieChartFavGenreRevenueContribution['label']),
535 #HYPOTHESIS NUMBER 1.2 : NOT ONLY LEAD ACTOR BUT A CLUSTER REPRESENTS A PAIR OF ACTOR POSSIBLE ACTOR THAT CONTRIBUTES ABOVE 50% OF
536     THE FILM
537 multipleActorRepresentativesPerCluster.to_csv('allCluster-possiblePairActorRepresentatives.csv')
538 multipleActorRepresentativesCountGroupBy = multipleActorRepresentativesPerCluster[['cluster_actor_label', 'howmanyactor']].groupby(
539     ('howmanyactor')).count()
540 multipleActorRepresentativesCountGroupBy['howmanyactor'] = multipleActorRepresentativesCountGroupBy.index
541 multipleActorRepresentativesCountGroupBy['howmanyactor'] = 'many_lead_actor_:' + multipleActorRepresentativesCountGroupBy['
542     howmanyactor'].astype(str)
543 # create pieChart
544 labels = np.array(multipleActorRepresentativesCountGroupBy['howmanyactor'])
545 sizes = np.array(multipleActorRepresentativesCountGroupBy['cluster_actor_label'])
546 plt.figure(figsize=(6,6))
547 plt.title('Every_Actor_Cluster_many_\nActor_Representatives', fontweight = 'bold')
548 patches, texts = plt.pie(sizes , startangle = 90)
549 plt.legend(patches , labels , loc='lower_left')
550 plt.axis('equal')
551 plt.tight_layout()
552 plt.show()
553
554 # plot pieChart yang consistent dan tidak
555 actorFavGenre_RevenueConsistencyPerCluster.to_csv('allActorCluster_favGenre_Revenue_ConsistencyDataFrame.csv')
556 consistentGroupBy = actorFavGenre_RevenueConsistencyPerCluster[['actor', 'status']].groupby('status').count()
557 plotModule.pieChart('Actor_Cluster_Consistent_Comparison', consistentGroupBy.index,consistentGroupBy['actor'])
558
559 # HYPOTHESIS PROFIT : ACTOR FAV GENRE PROFIT CONTRIBUTE HIGHER THAN 50 PERCENT
560 actorFavGenre_ProfitConsistencyPerCluster.to_csv('allClusterActor-FavGenreProfitContribution.csv')
561 pieChartFavGenreProfitContribution = pd.DataFrame({'label':[], 'count':[]})
562 higherThan50 = actorFavGenre_ProfitConsistencyPerCluster[actorFavGenre_ProfitConsistencyPerCluster['favgenrecontribution'] >= 50]
563 lowerThan50 = actorFavGenre_ProfitConsistencyPerCluster[actorFavGenre_ProfitConsistencyPerCluster['favgenrecontribution'] < 50]
564 pieChartFavGenreProfitContribution = pieChartFavGenreProfitContribution.append({'label':'cluster_with_actor_favourite_genre_\n
565     profit_contribution_more_than_50_percent', 'count': higherThan50.shape[0]}, ignore_index= True)
566 pieChartFavGenreProfitContribution = pieChartFavGenreProfitContribution.append({'label':'cluster_with_actor_favourite_genre_\n
567     profit_contribution_less_than_50_percent', 'count': lowerThan50.shape[0]}, ignore_index= True)
568 plotModule.pieChart('Actor_Favourite_Genre_Profit_Contribution_Comparison', pieChartFavGenreProfitContribution['label'] ,
569     pieChartFavGenreProfitContribution['count'])
570
571 # HYPOTHESIS ROI : ACTOR FAV GENRE ROI CONTRIBUTE HIGHER THAN 50 PERCENT
572 actorFavGenre_RoiConsistencyPerCluster.to_csv('allClusterActor-FavGenreRoiContribution.csv')
573 pieChartFavGenreRoiContribution = pd.DataFrame({'label':[], 'count':[]})
574 higherThan50 = actorFavGenre_RoiConsistencyPerCluster[actorFavGenre_RoiConsistencyPerCluster['favgenrecontribution'] >= 50]
575 lowerThan50 = actorFavGenre_RoiConsistencyPerCluster[actorFavGenre_RoiConsistencyPerCluster['favgenrecontribution'] < 50]
576 pieChartFavGenreRoiContribution = pieChartFavGenreRoiContribution.append({'label':'cluster_with_actor_favourite_genre_\nroi_
577     contribution_more_than_50_percent', 'count': higherThan50.shape[0]}, ignore_index = True)
578 pieChartFavGenreRoiContribution = pieChartFavGenreRoiContribution.append({'label':'cluster_with_actor_favourite_genre_\nroi_
579     contribution_more_less_50_percent', 'count': lowerThan50.shape[0]}, ignore_index = True)
580 plotModule.pieChart('Actor_Favourite_Genre_Roi_Contribution_Comparison' , pieChartFavGenreRoiContribution['label'],
581     pieChartFavGenreRoiContribution['count'])
582
583 # youtube Trailer view to Other REsponse every cluster
584 youtubeViewPearson_perCluster = youtubeViewPearson_perCluster.sort_values('pearson_revenue', ascending = False)
585 youtubeViewPearson_perCluster.to_csv('pearsonAllClusterYoutubeViewandAllResponseDataset.csv')
586 plotModule.barcharth(youtubeViewPearson_perCluster['cluster_actor_label'],
587     youtubeViewPearson_perCluster['pearson_revenue'],
588     'actor_Label', 'Pearson_Score',
589     'Pearson_Score_Barchart_Every_\n_Cluster_Youtube_VIEWS_and_Revenue', 'pearsonAllClusterYoutubeViewandRevenue.
590     jpg')
591
592 instagramHashtagPearson_perCluster = instagramHashtagPearson_perCluster.sort_values('pearson_revenue', ascending = False)
593 instagramHashtagPearson_perCluster.to_csv('pearsonAllClusterInstagramHashtagCountandAllResponseDataset.csv')
594 plotModule.barcharth(instagramHashtagPearson_perCluster['cluster_actor_label'],
595     instagramHashtagPearson_perCluster['pearson_revenue'],
596     'actor_Label', 'pearson_Score',
597     'Pearson_Score_Barchart_Every_\n_Cluster_Instagram_HashTag_Count_and_revenue' ,
598     'pearsonAllClusterInstagramHashtagCountandRevenue.jpg')

```

Listing A.4: clusteringGenerator.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 18 22:12:20 2020
4
5 @author: Teuku Hashrul
6 """
7
8
9 import pandas as pd
10 import numpy as np
11 import shutil
12 import matplotlib.pyplot as plt
13 import pickle
14 from sklearn.cluster import AgglomerativeClustering, KMeans
15 import os
16 from sklearn.metrics.pairwise import cosine_similarity , cosine_distances
17 from sklearn.metrics import silhouette_score
18 import sys
19 np.set_printoptions(threshold=sys.maxsize)
20 import plotModule
21

```

```

22| dataset = pd.read_csv('hasilEksperimen_FinalForm.csv')
23|
24| # =====
25| # use one hot encoding to make column actors become the features for clustering
26| # =====
27|
28| #split by , to make actors
29| splitted_actors = (dataset.set_index(dataset.columns.drop('actors',1).tolist()))
30|     .actors.str.split(',', expand=True)
31|     .stack()
32|     .reset_index()
33|     .rename(columns={0:'actors'})
34|     .loc[:, dataset.columns]
35| )
36| #preprocessing removing front space
37| splitted_actors['actors'] = splitted_actors['actors'].str.lstrip()
38| #preprocessing lowercase
39| splitted_actors['actors'] = splitted_actors['actors'].str.lower()
40|
41| # to convert actor into one hot style
42| onehot_actors = pd.crosstab(splitted_actors['title'],splitted_actors['actors']).rename_axis(None, axis=1).add_prefix('`')
43|
44| merged_inner = pd.merge(left=dataset,right=onehot_actors, left_on='title', right_on='title')
45|
46| cluster = pd.merge(left=dataset[['title' , 'revenue']],right=onehot_actors, left_on='title', right_on='title')
47|
48| merged_inner.to_csv('IMDBMOVIE_withonehotactor.csv')
49| cluster_del = cluster
50| del cluster_del['title']
51| del cluster_del['revenue']
52|
53|
54| # fulldata : dataset with all the column
55| # features : column used from the fullDataset (sub dataframe from full Data)
56| # methods : 'kmeans' or 'agglo' or any other methods
57| def generateCluster(fullDataset,features , method , n, titlefeatures):
58|
59|     # iterate every label
60|     allLabelMean = pd.DataFrame({"n":+ str(n): []})
61|     # create dataframe consist of label and the meancluster distance to join with the real dataset
62|     labelMeanDistance = pd.DataFrame({'label':[],'intraclusterdistance':[]})
63|     if method == 'kmeans':
64|         kmeans = KMeans(n_clusters = n)
65|         kmeans.fit(features)
66|         cluster_del['label'] = kmeans.labels_
67|         fullDataset['label'] = kmeans.labels_
68|         centroid = pd.DataFrame(kmeans.cluster_centers_)
69|         #remove label for countable
70|         centroid = centroid.iloc[:, :-1]
71|         for index in range(0,n):
72|             # take the only label in idx:
73|             rowLabelNow = cluster_del[cluster_del['label'] == index]
74|             del rowLabelNow['label']
75|             #locate the centroid
76|             centroidNow = centroid.iloc[index]
77|
78|             #count all the euclidean distance between every row in row labelnow and the centroidNow and put it in the label Now
79|             dist = (rowLabelNow - np.array(centroidNow)).pow(2).sum(1).pow(0.5)
80|             meanDistNow = np.mean(dist)
81|
82|             allLabelMean = allLabelMean.append({"n":+str(n):meanDistNow}, ignore_index = True)
83|             labelMeanDistance = labelMeanDistance.append({'label':index,'intraclusterdistance':meanDistNow}, ignore_index = True)
84|     elif method == 'agglo':
85|         agglo = AgglomerativeClustering(n_clusters = n , affinity='cosine', linkage='average')
86|         agglo.fit(features)
87|
88|         cluster_del['label'] = agglo.labels_
89|         fullDataset['label'] = agglo.labels_
90|
91|         print('experiment_' + str(n))
92|         for index in range(0,n):
93|             # take the only label in idx:
94|             rowLabelNow = cluster_del[cluster_del['label'] == index]
95|             del rowLabelNow['label']
96|             #create centroid from mean of all the member of the cluster
97|
98|
99|             centroidNow = np.mean(rowLabelNow)
100|
101|             if index == 8:
102|                 print(rowLabelNow)
103|                 print(fullDataset[fullDataset['label'] == 8])
104|
105|             #count all the cosine similarity distance between every row in row labelnow and the centroidNow and put it in the
106|             #label Now
107|             #dist = (rowLabelNow - np.array(centroidNow)).pow(2).sum(1).pow(0.5)
108|
109|             dist = 0
110|             lenn = 0
111|             for idx, row in rowLabelNow.iterrows():
112|                 cosine = cosine_similarity([row],[np.array(centroidNow)])
113|                 dist = dist + cosine[0][0]
114|                 lenn = lenn + 1
115|             meanDistNow = dist / lenn
116|
117|             allLabelMean = allLabelMean.append({"n":+str(n):meanDistNow}, ignore_index = True)
118|             print('label_:_' + str(index) + '_mean_distance:_' + str(meanDistNow))
119|             labelMeanDistance = labelMeanDistance.append({'label':index,'intraclusterdistance':meanDistNow}, ignore_index = True)

```

```

120 #leftouter join labelMeanDistance and the real dataset clustered
121 dataset_clusteredmean = pd.merge(labelMeanDistance,fullDataset,how='left', on = ['label'])
122
123 folder = ''
124 if method == 'kmeans':
125     folder = 'kmeans\\kmeans_n'+str(n)+ '\\'
126 elif method == 'agglo' :
127     folder = 'agglo\\agglo_n' +str(n)+ '\\'
128
129
130
131 if os.path.exists(folder):
132     shutil.rmtree(folder)
133 os.makedirs(folder)
134
135 #save dataset with label and the mean distance
136 dataset_clusteredmean.to_csv(folder+'clustered'+titlefeatures+str(n) +'dataset.csv')
137 # save model using pickle
138 model = None
139 if method == 'kmeans':
140     model = kmeans
141 elif method == 'agglo':
142     model = agglo
143 pickle.dump(model, open(folder+'model'+ '_clustered_ '+str(n)+titlefeatures, 'wb'))
144 return allLabelMean
145
146 # =====
147 #
148 # =====
149
150 silhouettescore_ex = pd.DataFrame({'n':[], "score":[]})
151 n_experiment = [10,50,100,200,300]
152 for i in n_experiment:
153     kmeans = KMeans(n_clusters=i)
154     kmeans.fit(cluster_del)
155     scorek = silhouette_score(cluster_del , kmeans.labels_)
156     silhouettescore_ex= silhouettescore_ex.append({'n':str(i), "score" :scorek} , ignore_index = True)
157
158
159 plotModule.barchart(silhouettescore_ex['n'] , silhouettescore_ex['score'] , 'Number_of_Clusters' , 'Silhouette_Score' , 'Kmeans_Silhouette_Score_Actor_Comparison',
160                     'kmeansSilhouettescore'+ str(n_experiment[0]) + '-' +str(n_experiment[len(n_experiment)-1])+'.jpg')
161
162
163
164
165 #also need to count timer
166 import timeit
167 start = timeit.default_timer()
168 #switchable 'kmeans' or 'agglo'
169 method = 'agglo'
170 allExperimentMean = pd.DataFrame({'n':[],'meanintracluster':[]})
171 experiment =[10,100,200,300,400]
172 for n_experiment in experiment:
173     allLabelMean = generateCluster(dataset,cluster_del ,method ,n_experiment , 'actor')
174     allExperimentMean = allExperimentMean.append({'n':n_experiment, 'meanintracluster':[allLabelMean['n']+str(n_experiment)]}, ignore_index=True)
175
176 stop = timeit.default_timer()
177 estimatedTime = stop - start
178
179 fig,ax = plt.subplots()
180 plt.title(method+'_intracluster_mean_distance_comparison_\nusing_box_and_whisker')
181 plt.ylabel('how_many_cluster')
182 plt.xlabel('intracluster_mean_distribution_for_every_label')
183 ax.set_yticklabels(allExperimentMean['n'])
184 ax.boxplot(allExperimentMean['meanintracluster'] , vert=False)

```

Listing A.5: prediksi.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Sep 27 12:39:54 2019
4
5 @author: TEUKU HASHRUL
6 """
7
8
9 import pandas as pd
10 import numpy as np
11 import matplotlib.pyplot as plt
12 from sklearn.linear_model import LinearRegression
13 from sklearn.preprocessing import PolynomialFeatures
14 from sklearn.model_selection import train_test_split
15 from sklearn.metrics import mean_squared_error, r2_score
16 dataset = pd.read_csv('hasileksperimen1.csv')
17
18 year_meanRating = dataset[['year', 'metascore']].groupby('year').count()
19 plt.bar(year_meanRating.index , year_meanRating['metascore'], linewidth = 1.2,edgecolor='black',alpha=0.5)
20 plt.xlabel('Year',fontweight='bold')
21 plt.ylabel('Rating',fontweight='bold')
22 plt.title('Mean_Film_Rating_Every_Year_ , fontweight='bold')
23
24
25
26
27 #function for scatter plot
28

```

```

29| def scatterplot(xVar , yVar, xName , yName):
30|     plt.scatter(xVar , yVar)
31|     plt.ylabel(yName)
32|     plt.xlabel(xName)
33|     plt.title("Scatter_plot_perbandingan_{0}_dan_{1}").format(xName , yName), fontweight = "bold")
34|     # only choose 1 , show() to the console , savefig save to the folder
35|     plt.show()
36|     #plt.savefig("scatterp-"+xName+"-"+yName+".jpg")
37|     plt.clf()
38|
39|
40| scatterplot(dataset['votes'] , dataset['revenue'] , 'Votes' , 'Revenue')
41| scatterplot(dataset['rating'] , dataset['revenue'] , 'Rating' , 'Revenue' )
42| scatterplot(dataset['metascore'] , dataset['revenue'] , 'Metascore' , 'Revenue')
43| scatterplot(dataset['year'] , dataset['revenue'] , 'Year' , 'Revenue')
44| scatterplot(dataset['runtime'] , dataset['revenue'] , 'Runtime' , 'Revenue')
45| # analisis selera pemirat dan reviewer
46|
47| normTest = dataset[['votes' , 'metascore' , 'rating']]
48| from sklearn import preprocessing
49| mm_scaler = preprocessing.MinMaxScaler()
50| votesNorm = mm_scaler.fit_transform( normTest[['votes']])
51| metascoreNorm = mm_scaler.fit_transform(normTest[['metascore']])
52| ratingNorm = mm_scaler.fit.transform(normTest[['rating']])
53| scatterplot(votesNorm , metascoreNorm, 'Votes' , 'Metascore')
54| scatterplot(votesNorm , ratingNorm, 'Votes' , 'Rating')
55| scatterplot(metascoreNorm , ratingNorm, 'Metascore' , 'Rating')
56| dataset.to_csv('hasileksperimen2.csv')
57|
58|
59|
60| # =====
61| #
62| # =====
63|
64| # use pearson correlation to find correlation between other feature and revenue
65| from scipy.stats.stats import pearsonr
66| pcor_runtime = pearsonr(dataset['runtime'] , dataset['revenue'])
67| pcor_rating = pearsonr(dataset['rating'] , dataset['revenue'])
68| pcor_votes = pearsonr(dataset['votes'] , dataset['revenue'])
69| pcor_metascore = pearsonr(dataset['metascore'] , dataset['revenue'])
70| pcor_year = pearsonr(dataset['year'] , dataset['revenue'])
71|
72| sc_runtime = pcor_runtime[0]
73|
74| pcor_dataset = pd.DataFrame({"name":['Runtime' , 'Rating' , 'Votes' , 'Metascore' , 'Year'],
75|                             "score": [pcor_runtime[0],pcor_rating[0],pcor_votes[0],pcor_metascore[0],pcor_year[0]]})
76| # sort by score
77| pcor_dataset = pcor_dataset.sort_values('score')
78|
79| def barchart(xVar , yVar, xName , yName , title):
80|     plt.bar(xVar , yVar,linewidth = 1.2,edgecolor='black',alpha=0.5)
81|     plt.xlabel(xName,fontweight='bold')
82|     plt.ylabel(yName,fontweight='bold')
83|     plt.title(title , fontweight='bold')
84|     #set axis if needed
85|     plt.ylim(0,1)
86|
87|     #plt.savefig("barchart-"+title+".jpg")
88|
89|     #draw line
90|     plt.axhline(y=0.0, color='r' , linestyle='--')
91|     plt.show()
92|     plt.clf()
93|
94| barchart(pcor_dataset['name'] , pcor_dataset['score'] , "Features" , "Pearson_Score" , "Barchart_Pearson_Correlation_Revenue_with_
95|          Other_Features_Comparison")
96|
97| # =====
98| # Linear Regression Example
99| # =====
100|
101|
102| #barchart(datasetfeatures['features'] , datasetfeatures['score'] , 'Features' , 'Accuracy Score' , 'Barchart comparing linear
103| # Regression score')
104| x= dataset[['votes']]
105| y = dataset[['revenue']]
106|
107|
108| x_arr = np.array(x.values)
109| y_arr = np.array(y.values)
110|
111| xtrain , xtest , ytrain , ytest = train_test_split(x_arr, y_arr , test_size =0.2, random_state =False)
112|
113| linearModel = LinearRegression()
114| linearModel.fit(xtrain , ytrain)
115|
116| linearModel.intercept_
117| linearModel.coef_
118| ypred = linearModel.predict(xtest)
119|
120| #ARRAY OF SQUARED ERROR
121| se = [pow(ypred - ytest,2 )]
122| nd = np.array(se)
123| nd = nd.ravel()
124|
125| plt.title("Linear_Regression_Revenue_Prediction_Squared_Error_\nusing_votes")
```

```

126 plt.boxplot(np.array(nd), vert = False)
127 plt.xlabel("Squared_Error_(Ytrue_-,Ypred)_\n"
128     +"Q1_:" + str(np.quantile(nd,0.25))+ "\n"
129     +"Q2_:" + str(np.quantile(nd,0.5)) + "\n"
130     +"Q3_:" + str(np.quantile(nd,0.75)))
131
132 se_selected = nd[nd<20000]
133
134 # =====
135 # SCORE Linear Regression
136 # =====
137 score = linearModel.score(xtest , ytest)
138
139 rmse_linear = np.sqrt(mean_squared_error(ytest,ypred))
140 r2_linear = r2_score(ytest,ypred)
141 print("rmse_linear:" + str(rmse_linear))
142 print("r2_linear:" + str(r2_linear))
143
144 votestest = xtest.ravel()
145 revenue test = ytest.ravel()
146 revenuepred = ypred.ravel()
147 linearPredComparison = []
148 for i in range(0, len(ytest)):
149     listNow = []
150     listNow.append(votestest[i])
151     listNow.append(revenue test[i])
152     listNow.append(revenuepred[i])
153
154 linearPredComparison.append(listNow)
155
156 #intercept
157 intercept = linearModel.intercept_[0]
158 coefficient = linearModel.coef_[0][0]
159 #
160 print("linear_pred_function_using_votes->REVENUE(votes) = " + str(intercept) + " + " + str(coefficient) + "(Votes) ")
161 #extends the x axis
162 plt.figure(figsize=(8,3))
163 plt.grid(zorder=3)
164 plt.scatter(xtest , ytest, zorder=3)
165
166 plt.ylabel('revenue')
167 plt.xlabel('votes')
168
169 plt.title("Scatter_plot_perbandingan_{},dan_{}".format('votes' , 'revenue'))
170 # only choose 1 , show() to the console , savefig save to the folder
171 plt.plot(xtest, ypred, color='red', linewidth=1)
172 plt.show()
173
174 #plt.savefig("linearregressionplot")
175 plt.clf()
176
177
178 # =====
179 # Polynomial Regression
180 # =====
181 import operator
182 # transforming the data to include another axis
183 xtest = xtest
184 xtrain = xtrain
185 ytrain = ytrain
186 ytest = ytest
187
188
189
190 degree_num =2
191 polynomial_features= PolynomialFeatures(degree=degree_num)
192
193 x_poly_train = polynomial_features.fit_transform(np.array(xtrain))
194 x_poly_test = polynomial_features.fit_transform(np.array(xtest))
195
196
197 model = LinearRegression()
198 model.fit(x_poly_train, ytrain)
199 y_poly_pred = model.predict(x_poly_test)
200
201 #SQUARED_ERROR
202 se = [pow(y_poly_pred - ytest, 2)]
203
204 nd_se = np.array(se)
205 nd_se = nd_se.ravel()
206
207 se_poly_selected = nd_se[nd_se < 20000]
208
209
210 # boxplot se polynom
211 plt.title("Polynomial_Regression_Revenue_Prediction_Squared_Error_\nusing_votes")
212 plt.boxplot(nd_se, vert = False)
213 plt.xlabel("Squared_Error_(Ytrue_-,Ypred)_\n"
214     +"Q1_:" + str(np.quantile(nd_se,0.25))+ "\n"
215     +"Q2_:" + str(np.quantile(nd_se,0.5)) + "\n"
216     +"Q3_:" + str(np.quantile(nd_se,0.75)))
217
218 # comparison SE linear and polynom
219 q2_linear_se = "{:.2f}.".format( np.quantile(nd , 0.5) )
220 q2_poly_se = "{:.2f}.".format( np.quantile(nd_se, 0.5) )
221
222 fig,ax = plt.subplots()
223 plt.title('Perbandingan_Squared_Error_Prediksi_Revenue_menggunakan_Votes')
224 plt.ylabel('Regresi')

```

```

225| plt.xlabel('Distribusi_milai_SE')
226| ax.set_yticklabels(['linear\n(Q2:' + q2_linear_se+')', 'polynomial\n(Q2:' + q2_poly_se+')'])
227| ax.boxplot([se_selected, se_poly_selected], vert=False)
228|
229| # print all the coefficient : if degree is 2 there is 3 coefficient a + bx + cx^2
230| model.coef_
231| arrcoef = model.coef_
232| rmse_poly = np.sqrt(mean_squared_error(ytest,y_poly_pred))
233| r2_poly = r2_score(ytest,y_poly_pred)
234| error = model.intercept_
235|
236| #extends the x axis
237| plt.figure(figsize=(7,3))
238|
239| #draw grid
240| plt.grid(zorder=0)
241| plt.scatter(x, y, s=10, zorder=3)
242| plt.title("Scatter_plot_perbandingan_{0}_dan_{1} ".format('votes' , 'revenue'))
243| # sort the values of x before line plot
244| sort_axis = operator.itemgetter(0)
245| sorted_zip = sorted(zip(x,y_poly_pred), key=sort_axis)
246| x, y_poly_pred = zip(*sorted_zip)
247| plt.plot(x, y_poly_pred, color='m')
248| plt.show()
249|
250|
251| # =====
252| # experiment detecting outlier
253| # =====
254|
255| from sklearn.cluster import KMeans
256| kmeans_model = KMeans(n_clusters = 2 , random_state = 0)
257| kmeans_model.fit(dataset[['revenue']])
258|
259|
260|
261|
262| # append labales of to the data
263| y_pred = kmeans_model.predict(dataset[['revenue']])
264|
265| dataset['labels'] = y_pred
266|
267|
268| # ambil yang outlier
269| posoutlier = dataset[dataset['labels'] == 1]
270| notoutlier = dataset[dataset['labels'] == 0]
271|
272|
273| # get all centroid centers
274| centroid = pd.DataFrame(kmeans_model.cluster_centers_)
275| #draw grid
276| plt.grid(zorder=0)
277| plt.scatter(notoutlier['votes'] , notoutlier['revenue'] , marker ='s' , s =50 , c='lightgreen' , label ='cluster1')
278| plt.scatter(posoutlier['votes'] , posoutlier['revenue'] , marker = 'v' , s =50 , c = 'red' , label ='cluster2')
279| plt.scatter(centroid[0] , centroid[1] , c = 'yellow' , s =75 , edgecolors='black' )
280| plt.legend(scatterpoints = 1)
281|
282| plt.show()
283| plt.clf()
284|
285|
286|
287|
288| # =====
289| # create bar chart
290| # =====

```

Listing A.6: genreClusterAnalysis-Revenue.py

```

1| # -*- coding: utf-8 -*-
2| """
3| Created on Thu Mar 12 00:25:28 2020
4|
5| @author: lenovo
6| """
7|
8| import pandas as pd
9| import plotModule
10| import os
11| import shutil
12| from scipy.stats.stats import pearsonr
13| import numpy as np
14| # from file genreCluster attempt, we choose that 189 is the optimal cluster with kmeans
15|
16| dataframe = pd.read_csv('aggo\\aggo_n189\\clusteredgenre189_dataset.csv')
17|
18| # how many genre combination
19| distinctGenreCombination = dataframe[['genre']].drop_duplicates()
20|
21| # Global analysis
22| genre_countTitle = dataframe[['genre' , 'title']].groupby('genre').count()
23| genre_countTitle = genre_countTitle.sort_values('title' , ascending = False)
24|
25| top20_genre_countTitle = genre_countTitle.head(20)
26| plotModule.barcharth(genre_countTitle.index , genre_countTitle['title'] , 'Genre_combination' , 'How_Many_Films_Created','Films_
27| _created_for_every_genre_\n_comparison' , 'genrecluster_combinationtitle_barchart.jpg')
plotModule.barcharth(top20_genre_countTitle.index , top20_genre_countTitle['title'] , 'Genre_combination' , 'Top_20_How_Many_Films
    _Created','Films_created_for_every_genre_\n_comparison' , 'top20genrecluster_combinationtitle_barchart.jpg')

```

```

28
29
30 genre_sumRevenue = dataframe[['genre', 'revenue']].groupby('genre').sum()
31 genre_sumRevenue = genre_sumRevenue.sort_values('revenue', ascending = False)
32 top20_genre_sumRevenue = genre_sumRevenue.head(20)
33 plotModule.barcharth(genre_sumRevenue.index, genre_sumRevenue['revenue'], 'Genre_Combination', 'Total_Revenue', 'Total_Revenue_'
34     _every_genre_\n_combination_barchart', 'genrecluster_totalRevenue_bchart.jpg')
35 plotModule.barcharth(top20_genre_sumRevenue.index, top20_genre_sumRevenue['revenue'], 'Genre_Combination', 'Total_Revenue', 'Total'
36     _Revenue_every_genre_\n_combination_barchart', 'top20_genrecluster_totalRevenue_bchart.jpg')
37
38 # get label only
39 labelOnly = dataframe[['label']]
40
41 # distinct label for looping purpose
42 labelOnly = labelOnly.drop_duplicates()
43
44 # HYPOTHESIS GENRE.1 : suatu Kombinasi punya actor dominan -> jumlah kontribusi
45 actorContributionPerCluster = pd.DataFrame({'genrecluster':[], 'clustercontribution':[]})
46
47 # HYPOTHESIS GENRE 2 : actor dominan dari kombinasi punya kontribusi revenue tinggi
48 actorRevenueContributionPerCluster = pd.DataFrame({'genrecluster':[], 'clustercontribution':[]})
49
50 # HYPOTHESIS GENRE PROFIT : actor dominan dari kombinasi punya kontribusi profit tinggi
51 actorProfitContributionPerCluster = pd.DataFrame({'genrecluster':[], 'clustercontribution':[]})
52
53 #HYPOTHESIS GENRE ROI : actor dominan dari kombinasi memiliki kontribusi roi tinggi
54 actorRoiContributionPerCluster = pd.DataFrame({'genrecluster':[], 'clustercontribution':[]})
55
56 #Barchart every cluster pearson revenue profit roi
57 youtubeViewPearson_perCluster = pd.DataFrame({'genrecluster':[], 'pearson_revenue':[], 'pearson_profit':[], 'pearson_roi':[]})
58
59 #Barchart every cluster actor hashtag count pearson
60 instagramHashtagPearson_perCluster = pd.DataFrame({'genrecluster':[], 'pearson_revenue':[], 'pearson_profit':[], 'pearson_roi':[]})
61
62
63
64
65 folderName = 'genreClusterAnalysis'
66 if os.path.exists(folderName):
67     shutil.rmtree(folderName)
68 os.makedirs(folderName)
69 for index,label in labelOnly.iterrows():
70     #get label now
71
72     labelNow = label[0]
73     print(labelNow)
74
75     #select the data for label Now
76     dataLabelNow = dataframe[dataframe['label'] == labelNow]
77
78     #since the cluster genre already accurate ,get 1 of the data
79     genreRep = dataLabelNow.genre[dataLabelNow.first_valid_index()]
80
81     # create directory for this label for every purpose
82     thisLabelFolderName = 'genre-Label-' + str(labelNow) + '-' + genreRep
83     thisLabelDirPath = folderName+'\\'+thisLabelFolderName
84     if os.path.exists(thisLabelDirPath):
85         shutil.rmtree(thisLabelDirPath)
86     os.makedirs(thisLabelDirPath)
87
88
89     #save the data
90     name_DataLabelNow = 'genre-' + 'label-' + str(labelNow) + '_dataset.csv'
91     dataLabelNow.to_csv(thisLabelDirPath+'\\'+name_DataLabelNow)
92
93     #generate WordCloud Actor from labelNow
94     split_column_actor = plotModule.split_column_actor(dataLabelNow)
95     #trim actor cause there are 'kirsten dunst' and ' Kirsten dunst'
96     split_column_actor['actors'] = split_column_actor['actors'].str.strip()
97
98     actoronly = split_column_actor['actors']
99     actoronly = actoronly.astype(str)
100    np_actor = np.array(actoronly).astype(str)
101    text_actor= "\n".join(np_actor).lower()
102
103    wordCloudFileName = thisLabelDirPath + '\\wordcloud-genrelabel-' + str(labelNow) + '.jpg'
104    plotModule.generateWordCloud(text_actor, wordCloudFileName, 'Actor_Distribution_for_Genre_Label_cluster:' + str(labelNow))
105
106
107     #boxPlot revenue for this cluster genre revenue estimation
108     boxplot_revenueName = thisLabelDirPath + '\\boxandwhiskerRevenue-genrelabel-' + str(labelNow) + '.jpg'
109     plotModule.boxplot(dataLabelNow['revenue'],False , 'Box_and_Whisker_Revenue_Distribution_genre_cluster-' + str(labelNow) ,
110         boxplot_revenueName)
111
112     #boxplot rating for this cluster rating estimation
113     boxplot_ratingName = thisLabelDirPath + '\\boxandwhiskerRating-genrelabel-' + str(labelNow) + '.jpg'
114     plotModule.boxplot(dataLabelNow['rating'],False , 'Box_and_Whisker_Rating_Distribution_genre_cluster-' + str(labelNow) ,
115         boxplot_ratingName)
116
117     #boxplot budget for this cluster estimation
118     boxplot_budgetName = thisLabelDirPath + '\\boxandwhiskerRoi-genrelabel-' + str(labelNow) + '.jpg'
119     plotModule.boxplot(dataLabelNow['us_budget'] , False , 'Box_and_Whisker_Budget_Distribution_genre_cluster-' + str(labelNow),
120         boxplot_budgetName)
121
122     #box plot profit for this cluster estimation
123     boxplot_profitName = thisLabelDirPath + '\\boxandwhiskerProfit-genrelabel-' + str(labelNow) + '.jpg'

```

```

122 plotModule.boxplot(dataLabelNow['profit'] , False , 'Box_and_Whisker_Profit_Distribution_genre_cluster-'+ str(labelNow) ,
123   boxplot_profitName)
124
125 #box plot roi for this cluster estimation
126 boxplot_roiName = thisLabelDirPath + '\\boxandwhiskerRoi-genrelabel-' + str(labelNow)+'.jpg'
127 plotModule.boxplot(dataLabelNow['roi'] , False , 'Box_and_Whisker_Roi_Distribution_genre_cluster-' + str(labelNow) ,
128   boxplot_roiName)
129
130 #boxplot youtubeView for this cluster estimation
131 boxplot_viewCountName = thisLabelDirPath + '\\boxandwhiskerviewCount-genrelabel-'+str(labelNow)+'.jpg'
132 plotModule.boxplot(dataLabelNow['viewCount'] , False , 'Box_and_Whisker_view_Count_Youtube_Distribution_genre_cluster-' + str(
133   labelNow), boxplot_viewCountName)
134
135 #boxplot Instagram Hashtag Count for this cluster estimation
136 boxplot_InstagramHashtagCountName = thisLabelDirPath + '\\boxandwhiskerHashtagCountInstagram-genrelabel-'+str(labelNow)+'.jpg'
137 plotModule.boxplot(dataLabelNow['hashtagcount'] , False , 'Box_and_Whisker_Instagram_Count_Hashtag_Distribution_genre_cluster-
138   '+ str(labelNow), boxplot_InstagramHashtagCountName)
139
140 # HYPOTHESIS GENRE 1 . : Every genre combination have dominant actor
141 # count every actor play how many title
142 actor_title = split_column_actor[['actors' , 'title']]
143 countAllActorInThisCluster = actor_title.groupby('actors').count()
144 # howManyFilmActorPlaysInCluster / howManyTitlesInTheCluster * 100
145 countAllActorInThisCluster['contribution'] = countAllActorInThisCluster['title'] / dataLabelNow.shape[0] * 100
146 actorContributionDataFrameName = thisLabelDirPath + '\\cluster-'+str(labelNow)+'-genreTitleContribution.csv'
147 #saveThisClusterActorContribution
148 countAllActorInThisCluster.to_csv(actorContributionDataFrameName)
149 #sort to get most contributing play in this cluster
150 countAllActorInThisCluster = countAllActorInThisCluster.sort_values('contribution', ascending=False)
151 mostPlayedActor = countAllActorInThisCluster.index[0]+str(labelNow)
152 mostPlayedContribution = countAllActorInThisCluster.contribution[0]
153
154 #add to The Global DataFrame
155 actorContributionPerCluster = actorContributionPerCluster.append({"genrecluster":mostPlayedActor, "clustercontribution":
156   mostPlayedContribution} , ignore_index=True)
157
158 # HYPOTHESIS GENRE 2 : the dominant actor also contribute to the biggest in revenue
159 actor_revenue = split_column_actor[['actors' , 'revenue']]
160 sumAllRevenuePerActorInThisCluster = actor_revenue.groupby('actors').sum()
161 # howManyFilmActorPlaysInCluster / howManyTitlesInTheCluster * 100
162 sumAllRevenuePerActorInThisCluster['contribution'] = sumAllRevenuePerActorInThisCluster['revenue'] / dataLabelNow['revenue'].sum() * 100
163 sumAllRevenuePerActorInThisCluster = sumAllRevenuePerActorInThisCluster.sort_values('contribution' , ascending = False)
164 sumActorRevenueDataFrameName = thisLabelDirPath + '\\cluster-'+str(labelNow)+'-actorContributionRevenue.csv'
165 sumAllRevenuePerActorInThisCluster.to_csv(sumActorRevenueDataFrameName)
166
167 thisGenreCluster_favActor = genreRep + '-' + mostPlayedActor
168 thisGenreCluster_favRevenueContribution = sumAllRevenuePerActorInThisCluster.contribution[countAllActorInThisCluster.index[0]]
169 actorRevenueContributionPerCluster = actorRevenueContributionPerCluster.append({'genrecluster':thisGenreCluster_favActor,
170   'clustercontribution':thisGenreCluster_favRevenueContribution}, ignore_index = True)
171
172 # HYPOTHESIS PROFIT IN CLUSTER GENRE : the dominant actor also contribute to the biggest
173 actor_profit = split_column_actor[['actors' , 'profit']]
174 sumAllProfitPerActorInThisCluster = actor_profit.groupby('actors').sum()
175 #howMuchProfitActorPlaysInThisCluster / HowMuchProfitInThisCluster * 100
176 sumAllProfitPerActorInThisCluster['contribution'] = sumAllProfitPerActorInThisCluster['profit'] / dataLabelNow['profit'].sum() * 100
177 sumAllProfitPerActorInThisCluster = sumAllProfitPerActorInThisCluster.sort_values('contribution' , ascending= False)
178 sumActorProfitDataFrameName = thisLabelDirPath + '\\cluster-'+ str(labelNow)+'-actorContributionProfit.csv'
179 sumAllProfitPerActorInThisCluster.to_csv(sumActorProfitDataFrameName)
180
181 thisGenreCluster_favProfitContribution = sumAllProfitPerActorInThisCluster.contribution[countAllActorInThisCluster.index[0]]
182 actorProfitContributionPerCluster = actorProfitContributionPerCluster.append({'genrecluster':thisGenreCluster_favActor,
183   'clustercontribution':thisGenreCluster_favProfitContribution}, ignore_index = True)
184
185 #HYPOTHESIS ROI IN CLUSTER GENRE : the dominant genre also contribute to the biggest
186 actor_roi = split_column_actor[['actors' , 'roi']]
187 sumAllRoiPerActorInThisCluster = actor_roi.groupby('actors').sum()
188 #howmuchroiactorPlayesInThisCluster / HowMuchRoiInThisCluster * 100
189 sumAllRoiPerActorInThisCluster['contribution'] = sumAllRoiPerActorInThisCluster['roi'] / dataLabelNow['roi'].sum() * 100
190 sumAllRoiPerActorInThisCluster = sumAllRoiPerActorInThisCluster.sort_values('contribution' , ascending = False)
191 sumActorRoiDataFrameName = thisLabelDirPath + '\\cluster-'+ str(labelNow) + '-actorContributionRoi.csv'
192 sumAllRoiPerActorInThisCluster.to_csv(sumActorRoiDataFrameName)
193
194 thisGenreCluster_favRoiContribution = sumAllRoiPerActorInThisCluster.contribution[countAllActorInThisCluster.index[0]]
195 actorRoiContributionPerCluster = actorRoiContributionPerCluster.append({'genrecluster':thisGenreCluster_favActor,
196   'clustercontribution':thisGenreCluster_favRoiContribution} , ignore_index = True)
197
198 # BARCHART PEARSON YOUTUBEVIEW and InstagramHashtagCount
199 if dataLabelNow.shape[0] > 1:
200   pearson_youtubecount_roi = pearsonr(dataLabelNow['roi'] , dataLabelNow['viewCount'])[0]
201   pearson_youtubecount_revenue = pearsonr(dataLabelNow['revenue'] , dataLabelNow['viewCount'])[0]
202   pearson_youtubecount_profit = pearsonr(dataLabelNow['profit'] , dataLabelNow['viewCount'])[0]
203
204   pearson_instagramcounthashtag_roi = pearsonr(dataLabelNow['roi'] , dataLabelNow['hashtagcount'])[0]
205   pearson_instagramcounthashtag_revenue = pearsonr(dataLabelNow['revenue'] , dataLabelNow['hashtagcount'])[0]
206   pearson_instagramcounthashtag_profit = pearsonr(dataLabelNow['profit'] , dataLabelNow['hashtagcount'])[0]
207
208   youtubeViewPearson_perCluster = youtubeViewPearson_perCluster.append({'genrecluster':genreRep+'-'+str(labelNow),
209     'pearson_revenue':pearson_youtubecount_revenue,
210     'pearson_profit':pearson_youtubecount_profit,
211     'pearson_roi':pearson_youtubecount_roi},

```

```

209 ignore_index =True)
210 instagramHashtagPearson_perCluster =instagramHashtagPearson_perCluster.append({'genrecluster':genreRep+'-'+str(labelNow),
211                                         'pearson_revenue':pearson_instagramcounthashtag_revenue
212                                         , 'pearson_profit':pearson_instagramcounthashtag_profit, 'pearson_roi':
213                                         pearson_instagramcounthashtag_roi}, ignore_index = True)
214
215 # HYPOTHESIS GENRE 1 : every genre combination
216 # saved all The most Played actor representatives
217 actorContributionPerCluster = actorContributionPerCluster.sort_values('clustercontribution', ascending = False)
218 actorContributionPerCluster.to_csv('allClusterGenre-mostPlayedActor.csv')
219 #visualize
220 plotModule.barcharth(actorContributionPerCluster['genrecluster'] , actorContributionPerCluster['clustercontribution'], 'genre_
221 cluster_\n_actor_representatives' , 'contribution_(Percent)' , 'Bchart_Every_Genre_Cluster_Most_Played_Actor',
222 bchartMostPlayedActor.jpg')
223 biggerThan50 = actorContributionPerCluster[actorContributionPerCluster['clustercontribution'] >= 50]
224 lowerThan50 = actorContributionPerCluster[actorContributionPerCluster['clustercontribution'] < 50]
225 pieChartActorTitleContribution = pd.DataFrame({'label':[], 'count':[]})
226 pieChartActorTitleContribution = pieChartActorTitleContribution.append({'label':'clusterGenreWithActorThat_\n_
227 PlayedMoreThan50Percent', 'count':biggerThan50.shape[0]}, ignore_index=True)
228 pieChartActorTitleContribution = pieChartActorTitleContribution.append({'label':'clusterGenreWithActorThat_\n_
229 PlayedLessThan50Percent', 'count':lowerThan50.shape[0]}, ignore_index=True)
230 # plot the Distribution for comparison
231 plotModule.pieChart('Actor_Representative_Cluster_Genre_Comparison',pieChartActorTitleContribution['label'],
232 pieChartActorTitleContribution['count'])
233
234 #HYPOTHESIS GENRE 2: actor representatives also contribute to high revenue
235 actorRevenueContributionPerCluster = actorRevenueContributionPerCluster.sort_values('clustercontribution', ascending = False)
236 actorRevenueContributionPerCluster.to_csv('allClusterGenre-favActorRevenueContribution.csv')
237 plotModule.barcharth(actorRevenueContributionPerCluster['genrecluster'] ,
238                     actorRevenueContributionPerCluster['clustercontribution'],
239                     'genre_cluster_fav_actor_representatives_\n_revenue_Contribution' ,
240                     'contribution_(Percent)' ,
241                     'Bchart_Every_Genre_Cluster_Most_Played_Actor',
242                     'bchartMostPlayedActor_revenueContribution.jpg')
243 revenueConBiggerThan50 = actorRevenueContributionPerCluster[actorRevenueContributionPerCluster['clustercontribution'] >=50]
244 revenueConLowerThan50 = actorRevenueContributionPerCluster[actorRevenueContributionPerCluster['clustercontribution'] < 50]
245
246 pieChartActor_SumRevenueContribution = pd.DataFrame({'label':[], 'count':[]})
247 pieChartActor_SumRevenueContribution = pieChartActor_SumRevenueContribution.append({'label':'favoriteActorPerGenreCluster_\n_
248 Revenue_contribute_more_than_50_percent', 'count':revenueConBiggerThan50.shape[0]}, ignore_index = True)
249 pieChartActor_SumRevenueContribution = pieChartActor_SumRevenueContribution.append({'label':'favoriteActorPerGenreCluster_\n_
250 Revenue_contribute_less_than_50_percent', 'count':revenueConLowerThan50.shape[0]}, ignore_index = True)
251 plotModule.pieChart('Revenue_Actor_Representative_Cluster_Genre_Comparison',pieChartActor_SumRevenueContribution['label'],
252 pieChartActor_SumRevenueContribution['count'])
253
254 # HYPOTHESIS GENRE PROFIT : actor representatives also contribute to highest profit
255 actorProfitContributionPerCluster = actorProfitContributionPerCluster.sort_values('clustercontribution', ascending = False)
256 actorProfitContributionPerCluster.to_csv('allClusterGenre-favActorProfitContribution.csv')
257 plotModule.barcharth(actorProfitContributionPerCluster['genrecluster'],
258                     actorProfitContributionPerCluster['clustercontribution'],
259                     'genre_cluster_fav_actor_representatives_\n_profit_Contribution',
260                     'contribution(Percent)' ,
261                     'Bchart_Every_Genre_Cluster_Actor_Profit_Contribution',
262                     'bchartMostPlayedActor_profitContribution.jpg')
263 profitConBiggerThan50 = actorProfitContributionPerCluster[actorProfitContributionPerCluster['clustercontribution']>=50]
264 profitConLowerThan50 = actorProfitContributionPerCluster[actorProfitContributionPerCluster['clustercontribution'] < 50]
265
266 pieChartActor_sumProfitContribution = pd.DataFrame({'label':[], 'count':[]})
267 pieChartActor_sumProfitContribution = pieChartActor_sumProfitContribution.append({'label':'favouriteActorPerGenreCluster_\n_Profit
268 _contribute_more_than_50_percent', 'count':profitConBiggerThan50.shape[0]}, ignore_index = True)
269 pieChartActor_sumProfitContribution = pieChartActor_sumProfitContribution.append({'label':'favouriteActorPerGenreCluster_\n_Profit
270 _contribute_less_than_50_percent', 'count':profitConLowerThan50.shape[0]}, ignore_index = True)
271 plotModule.pieChart('Profit_Actor_Representative_Cluster_Genre_Comparison',pieChartActor_sumProfitContribution['label'],
272 pieChartActor_sumProfitContribution['count'])
273
274 #HYPOTHESIS GENRE ROI : actor Representatives also contribute to highest ROI
275 actorRoiContributionPerCluster = actorRoiContributionPerCluster.sort_values('clustercontribution', ascending = False)
276 actorRoiContributionPerCluster.to_csv('allClusterGenre-favActorRoiContribution.csv')
277 plotModule.barcharth(actorRoiContributionPerCluster['genrecluster'],
278                     actorRoiContributionPerCluster['clustercontribution'],
279                     'genre_cluster_fav_actor_representatives_\n_roi_contribution',
280                     'contribution_(Percent)' ,
281                     'Bchart_Every_Genre_Cluster_Actor_ROI_Contribution',
282                     'bchartMostPlayedActor_roiContribution.jpg')
283
284 roiConBiggerThan50 = actorRoiContributionPerCluster[actorRoiContributionPerCluster['clustercontribution']>=50]
285 roiConLowerThan50 = actorRoiContributionPerCluster[actorRoiContributionPerCluster['clustercontribution']< 50]
286
287 pieChartActor_sumRoiContribution = pd.DataFrame({'label':[], 'count':[]})
288 pieChartActor_sumRoiContribution = pieChartActor_sumRoiContribution.append({'label':'favouriteActorPerGenreCluster_\n_ROI_
289 _contribute_more_than_50_percent', 'count':roiConBiggerThan50.shape[0]}, ignore_index = True)
290 pieChartActor_sumRoiContribution = pieChartActor_sumRoiContribution.append({'label':'favouriteActorPerGenreCluster_\n_ROI_
291 _contribute_less_than_50_percent', 'count':roiConLowerThan50.shape[0]}, ignore_index = True)
292 plotModule.pieChart('Roi_Actor_Representatives_Cluster_Genre_Comparison', pieChartActor_sumRoiContribution['label'],
293 pieChartActor_sumRoiContribution['count'])

```

```

291
292 # youtube Trailer view to Other REsponse every cluster
293 youtubeViewPearson_perCluster = youtubeViewPearson_perCluster.sort_values('pearson_revenue', ascending = False)
294 youtubeViewPearson_perCluster.to_csv('pearsonAllClusterYoutubeViewandAllResponseDataset.csv')
295 plotModule.barcharth(youtubeViewPearson_perCluster['genrecluster'],
296                      youtubeViewPearson_perCluster['pearson_revenue'],
297                      'genre_Label', 'Pearson_score',
298                      'Pearson_Score_Bchart_Every_\n_Cluster_Youtube_VIEWS_and_Revenue', 'pearsonAllClusterYoutubeViewandRevenue.
299                      jpg')
300
301 instagramHashtagPearson_perCluster = instagramHashtagPearson_perCluster.sort_values('pearson_revenue', ascending = False)
302 instagramHashtagPearson_perCluster.to_csv('pearsonAllClusterInstagramHashtagCountandAllResponseDataset.csv')
303 plotModule.barcharth(instagramHashtagPearson_perCluster['genrecluster'],
304                      instagramHashtagPearson_perCluster['pearson_revenue'],
305                      'genre_Label', 'pearson_Score',
306                      'Pearson_Score_Bchart_Every_\n_Cluster_Instagram_Hashtag_Count_and_revenue', 'pearsonAllClusterInstagramHashtagCountandRevenue.jpg')

```

Listing A.7: genreClusterAttempt.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 2 10:11:13 2020
4
5 @author: lenovo
6 """
7
8 import pandas as pd
9 import shutil
10 import pandas as pd
11 import matplotlib.pyplot as plt
12 import numpy as np
13 from sklearn.metrics import silhouette_score
14 import pickle
15 from sklearn.cluster import AgglomerativeClustering, KMeans
16 import os
17 from sklearn.metrics.pairwise import cosine_similarity, cosine_distances
18 import sys
19 np.set_printoptions(threshold=sys.maxsize)
20 import plotModule
21
22
23 dataset = pd.read_csv('hasilEksperimen_FinalForm.csv')
24 # fulldata : dataset with all the column
25 # features : column used from the fullDataset (sub dataframe from full Data)
26 # methods : 'kmeans' or 'agglo' or any other methods
27 def generateCluster(fullDataset,features , method , n, titlefeatures):
28     # iterate every label
29     allLabelMean = pd.DataFrame({"n:"+ str(n): []})
30     # create dataframe consist of label and the meancluster distance to join with the real dataset
31     labelMeanDistance = pd.DataFrame({'label':[],'intraclusterdistance':[]})
32     if method == 'kmeans':
33         kmeans = KMeans(n_clusters = n)
34         kmeans.fit(features)
35         cluster_del['label'] = kmeans.labels_
36         fullDataset['label'] = kmeans.labels_
37         centroid = pd.DataFrame(kmeans.cluster_centers_)
38         #remove label for countable
39         centroid = centroid.iloc[:, :-1]
40         for index in range(0,n):
41             # take the only label in idx:
42             rowLabelNow = cluster_del[cluster_del['label'] == index]
43             del rowLabelNow['label']
44             #locate the centroid
45             centroidNow = centroid.iloc[index]
46
47             #count all the euclidean distance between every row in row labelnow and the centroidNow and put it in the label Now
48             dist = (rowLabelNow - np.array(centroidNow)).pow(2).sum(1).pow(0.5)
49             meanDistNow = np.mean(dist)
50
51             allLabelMean = allLabelMean.append({"n:"+str(n):meanDistNow}, ignore_index = True)
52             labelMeanDistance = labelMeanDistance.append({'label':index,'intraclusterdistance':meanDistNow}, ignore_index = True)
53     elif method == 'agglo':
54         agglo = AgglomerativeClustering(n_clusters = n , affinity='cosine', linkage='average')
55         agglo.fit(features)
56
57         cluster_del['label'] = agglo.labels_
58         fullDataset['label'] = agglo.labels_
59
60         print('experiment_' + str(n))
61         for index in range(0,n):
62             # take the only label in idx:
63             rowLabelNow = cluster_del[cluster_del['label'] == index]
64             del rowLabelNow['label']
65             #create centroid from mean of all the member of the cluster
66
67             centroidNow = np.mean(rowLabelNow)
68
69             if index == 8:
70                 print(rowLabelNow)
71                 print(fullDataset[fullDataset['label'] == 8])
72
73             #count all the cosine similarity distance between every row in row labelnow and the centroidNow and put it in the
74             #label Now
75             #dist = (rowLabelNow - np.array(centroidNow)).pow(2).sum(1).pow(0.5)
76

```

```

77     dist = 0
78     lenn = 0
79     for idx, row in rowLabelNow.iterrows():
80         cosine = cosine_similarity([row],[np.array(centroidNow)])
81         dist = dist + cosine[0][0]
82         lenn = lenn + 1
83     meanDistNow = dist / lenn
84
85
86     allLabelMean = allLabelMean.append({"n":str(n):meanDistNow}, ignore_index = True)
87     print('Label_.' + str(index) + '_mean_distance:' + str(meanDistNow))
88     labelMeanDistance = labelMeanDistance.append({'label':index,'intraclusterdistance':meanDistNow}, ignore_index = True)
89
90 #leftouter join labelMeanDistance and the real dataset clustered
91 dataset_clusteredmean = pd.merge(labelMeanDistance,fullDataset,how='left', on = ['label'])
92
93 folder = ''
94 if method == 'kmeans':
95     folder = 'kmeans\kmeans_n'+str(n)+ '\\'
96 elif method == 'agglo' :
97     folder = 'agglo\agglo_n' +str(n)+ '\\'
98
99 if os.path.exists(folder):
100     shutil.rmtree(folder)
101 os.makedirs(folder)
102 #save dataset with label and the mean distance
103 dataset_clusteredmean.to_csv(folder+'clustered'+titlefeatures+str(n)+'_dataset.csv')
104 # save model using pickle
105 model = None
106 if method == 'kmeans':
107     model = kmeans
108 elif method == 'agglo':
109     model = agglo
110 pickle.dump(model, open(folder+'model'+ '_clustered_'+str(n)+titlefeatures, 'wb'))
111 return allLabelMean
112
113
114 #split by , to make actors
115 splitted_genre = (dataset.set_index(dataset.columns.drop('genre',1).tolist())
116                     .genre.str.split(',', expand=True)
117                     .stack()
118                     .reset_index()
119                     .rename(columns={0:'genre'})
120                     .loc[:, dataset.columns]
121                 )
122
123 #preprocessing removing front space
124 splitted_genre['genre'] = splitted_genre['genre'].str.lstrip()
125 #preprocessing lowercase
126 splitted_genre['genre'] = splitted_genre['genre'].str.lower()
127
128 # make row value into column
129 onehot_genre = pd.crosstab(splitted_genre['title'],splitted_genre['genre']).rename_axis(None, axis=1).add_prefix('')
130
131 # if want to merge with the dataset
132 #merged_inner = pd.merge(left=dataset,right=onehot_actors, left_on='title', right_on='title')
133 #merged_inner.to_csv()
134 cluster = pd.merge(left=dataset[['title', 'genre']],right=onehot_genre, left_on='title', right_on='title')
135 cluster_del = cluster
136 del cluster_del['title']
137 del cluster_del['genre']
138
139 # =====
140 # experiment genre using kmeans and shilhouette
141 # =====
142 # make dataframe consist of all the experiment score
143 silhouettescore_ex = pd.DataFrame({"n":[] , "score":[]})
144 n_experiment = [10,50,100,200,300]
145 for i in n_experiment:
146     kmeans = KMeans(n_clusters=i)
147     kmeans.fit(cluster_del)
148     scorek = silhouette_score(cluster_del , kmeans.labels_)
149     silhouettescore_ex= silhouettescore_ex.append({"n":str(i), "score" :scorek} , ignore_index = True)
150
151
152 plotModule.barchart(silhouettescore_ex['n'] , silhouettescore_ex['score'] , 'Number_of_Clusters' , 'Silhouette_Score' , 'Kmeans_Silhouette_Score_Genre_Comparison',
153                         'kmeansSilhouettescore'+ str(n_experiment[0]) + '-' +str(n_experiment[len(n_experiment)-1])+'.jpg')
154
155
156
157
158
159
160 #also need to count timer
161 import timeit
162 start = timeit.default_timer()
163 #switchable 'kmeans' or 'agglo'
164 method = 'agglo'
165 allexperimentMean = pd.DataFrame({'n':[],'meanintracluster':[]})
166 experiment =[189,180,190,200]
167 for n_experiment in experiment:
168     allLabelMean = generateCluster(dataset,cluster_del ,method ,n_experiment, 'genre')
169     allexperimentMean.append({'n':n_experiment,'meanintracluster':[allLabelMean['n']+str(n_experiment)]}),
170             ignore_index=True)
171 stop = timeit.default_timer()
172 estimatedTime = stop - start
173

```

```

174| fig,ax = plt.subplots()
175| plt.title(method+'\nintraclass_mean_distance_comparison\nusing_box_and_whisker_cluster_genre_attempt')
176| plt.ylabel('how_many_cluster')
177| plt.xlabel('intraclass_mean_distribution_for_every_label')
178| ax.set_yticklabels(allexperimentMean['n'])
179| ax.boxplot(allexperimentMean['meanintraclass'] , vert=False)

```

Listing A.8: heatmapGenerator.py

```

1| # -*- coding: utf-8 -*-
2| """
3| Created on Mon Mar 30 13:49:36 2020
4|
5| @author: lenovo
6| """
7|
8| import pandas as pd
9| import seaborn as sns
10|
11|
12| dataset = pd.read_csv('hasileksperimen-WithInstagram2.csv')
13|
14| corr = dataset[['votes', 'runtime', 'rating', 'metascore',
15|                 'profit', 'roi', 'us_budget', 'revenue',
16|                 'viewCount', 'hashtagcount','actorhashtagcount']].corr()
17| ax = sns.heatmap(
18|     corr,
19|     vmin=-1, vmax=1, center=0,
20|     cmap=sns.diverging_palette(20, 220, n=200),
21|     square=True
22| )
23| ax.set_xticklabels(
24|     ax.get_xticklabels(),
25|     rotation=45,
26|     horizontalalignment='right'
27| );

```

Listing A.9: mergeddataset-instagramwith-actor.py

```

1|
2|
3| # -*- coding: utf-8 -*-
4| """
5| Created on Tue Mar 31 20:40:08 2020
6|
7| @author: lenovo
8| """
9|
10| import pandas as pd
11| import re
12| # =====
13| # dataset = pd.read_csv('hasileksperiment-withInstagram.csv')
14| # listactor = pd.read_csv('secondattemptactororig.csv')
15| #
16| #
17| # dataset['hashtagigactor'] = dataset['actors']
18| # dataset['hashtagigactor'] = dataset['hashtagigactor']
19| #
20| #
21| #
22| # for i, row in dataset.iterrows():
23| #     first_actor = row.actors
24| #     first_actor = first_actor.split(',') [0]
25| #     first_actor = first_actor.lower()
26| #     first_actor = first_actor.replace(' ', '')
27| #     first_actor = re.sub(r'[^w\s]', '', first_actor)
28| #     first_actor = '#'+first_actor
29| #     dataset.at[i, 'hashtagigactor'] = first_actor
30| # new_dataset = pd.merge(dataset, listactor , how ='left', on ='hashtagigactor')
31| #
32| #
33| # new_dataset.to_csv('hasileksperimen-WithInstagram2.csv')
34| # =====
35|
36|
37| dataset = pd.read_csv('hasileksperimen-WithInstagram2.csv')
38|
39| secondhashtag = pd.read_csv('instagramhashtagactor_secondscrappeddata.csv')
40| secondhashtag.rename(columns = {'Title': 'hashtagigactor2', 'Field1': 'actorhashtagcount2' }, inplace= True)
41| thirdhashtag = pd.read_csv('instagramhashtagactor_thirdscrappeddata.csv')
42| thirdhashtag.rename(columns = {'Title': 'hashtagigactor3', 'Field1': 'actorhashtagcount3'}, inplace = True)
43| fourthhashtag = pd.read_csv('instagramhashtagactor_fourscrappeddata.csv')
44| fourthhashtag.rename(columns = {'Title': 'hashtagigactor4', 'Field1': 'actorhashtagcount4'} ,inplace = True)
45|
46| # second hashtag
47| dataset['hashtagigactor2'] = dataset['actors']
48| for i, row in dataset.iterrows():
49|     second_actor = row.actors
50|     second_actor = second_actor.split(',') [1]
51|     second_actor = second_actor.lower()
52|     second_actor = second_actor.replace(' ', '')
53|     second_actor = re.sub(r'[^w\s]', '', second_actor)
54|     second_actor = '#'+second_actor
55|     dataset.at[i, 'hashtagigactor2'] = second_actor
56| dataset= pd.merge(dataset, secondhashtag , how ='left', on ='hashtagigactor2')
57| #

```

```

58 # third
59 dataset['hashtagigactor3'] = dataset['actors']
60 for i, row in dataset.iterrows():
61     third_actor = row.actors
62     third_actor = third_actor.split(',') [2]
63     third_actor = third_actor.lower()
64     third_actor = third_actor.replace('_', ' ')
65     third_actor = re.sub(r'[^w\s]', '', third_actor)
66     third_actor = '#'+third_actor
67     dataset.at[i,'hashtagigactor3'] = third_actor
68 dataset= pd.merge(dataset, thirdhashtag , how ='left', on ='hashtagigactor3')
69
70 # fourth
71 dataset['hashtagigactor4'] = dataset['actors']
72 for i, row in dataset.iterrows():
73     if(len(row.actors.split(',')) > 3):
74         fourth_actor = row.actors
75         fourth_actor = fourth_actor.split(',') [3]
76         fourth_actor = fourth_actor.lower()
77         fourth_actor = fourth_actor.replace('_', ' ')
78         fourth_actor = re.sub(r'[^w\s]', '', fourth_actor)
79         fourth_actor = '#'+fourth_actor
80     dataset.at[i,'hashtagigactor4'] = fourth_actor
81 dataset= pd.merge(dataset, fourthhashtag , how ='left', on ='hashtagigactor4')
82
83
84
85
86
87 dataset.to_csv('hasilEksperimen_FinalForm.csv')

```

Listing A.10: plotModule.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 13 15:02:03 2020
4
5 @author: lenovo
6
7
8 import matplotlib.pyplot as plt
9 from wordcloud import WordCloud
10 import shutil
11 import math
12 import pandas as pd
13 import numpy as np
14 import matplotlib.pyplot as plt
15 import pickle
16 from sklearn.cluster import AgglomerativeClustering, KMeans
17 import os
18 from sklearn.metrics.pairwise import cosine_similarity , cosine_distances
19 import sys
20 np.set_printoptions(threshold=sys.maxsize)
21
22 def scatterplot(xVar , yVar, xName , yName , dirFileName):
23     plt.figure(figsize=(10,13 ))
24     plt.scatter(xVar , yVar)
25     plt.ylabel(yName)
26     plt.xlabel(xName)
27     plt.title("Scatter_plot_perbandingan_{0}_dan_{1}").format(xName , yName), fontweight = "bold")
28     # only choose 1 , show() to the console , savefig save to the folder
29
30     plt.savefig(dirFileName)
31     plt.clf()
32
33
34 # method to create histogram on every numerical feature
35 def histogram(xData , xName , yName , num_bins , title , intervalX , dirToSave):
36
37     import numpy as np
38     plt.figure(figsize=(10,8))
39     plt.grid(zorder = 0)
40
41     # set interval for x
42     if intervalX is not None:
43         plt.xticks(np.arange(min(xData) , max(xData) , intervalX))
44
45
46     n , bins , patches = plt.hist(xData , num_bins , facecolor = 'green' , alpha =0.5 , linewidth = 1.2,
47                                     edgecolor='black' , zorder=3)
48     plt.xlabel(xName)
49     plt.ylabel(yName)
50     # setting range histogram x axis plt.xticks(np.arange(lowest,highest , interval))
51     #plt.xticks(np.arange(-10,850,100))
52     plt.title(title ,fontweight = "bold")
53     plt.savefig(dirToSave)
54     #plt.show()
55     plt.clf()
56
57
58 def barchart(xVar , yVar , xName , yName , title , dirName):
59     plt.figure(figsize=(10,13))
60     plt.bar(xVar , yVar,edgecolor='black')
61     plt.xlabel(xName,fontweight='bold')
62     plt.ylabel(yName,fontweight='bold')
63     plt.title(title , fontweight='bold')
64     #set axis if needed
65     #plt.ylim(0,1)

```



```

165     plt.legend(patches, labels, loc='left_center', bbox_to_anchor=(-0.1, 1.),
166                 fontsize=8)
167 # =====
168 #
169 #     plt.savefig('piechart.png', bbox_inches='tight')
170 #     plt.figure(figsize=(10,10))
171 #     plt.title(title , fontweight = "bold")
172 #     plt.pie(values , labels = labels , autopct ='%.1f%%' , shadow=True , startangle = 90)
173 #     plt.legend()
174 #
175 # =====
176     plt.show()
177     plt.clf()
178
179 def barcharth(yLabel , xValue , yName , xName , title, directoryfilepath):
180     # plt.figure(figsize=(15,30))
181     plt.figure(figsize=(17,10))
182     #
183
184     plt.title(title,fontweight='bold')
185     plt.barh(yLabel , xValue ,edgecolor='black')
186     plt.xlabel(xName)
187     plt.ylabel(yName)
188     #plt.show()
189     plt.savefig(directoryfilepath, dpi=150)
190     plt.clf()
191
192
193
194 # split genre multiple row by comma delimiter
195 def split_column_actor(inputdataset):
196     reshaped = (inputdataset.set_index(inputdataset.columns.drop('actors',1).tolist())
197                 .actors.str.split(',') , expand=True)
198     .stack()
199     .reset_index()
200     .rename(columns={0:'actors'})
201     .loc[:, inputdataset.columns]
202     )
203
204     return reshaped
205
206 # split genre multiple row by comma delimiter
207 def split_column_genre(inputdataset):
208     reshaped = (inputdataset.set_index(inputdataset.columns.drop('genre',1).tolist())
209                 .genre.str.split(',') , expand=True)
210     .stack()
211     .reset_index()
212     .rename(columns={0:'genre'})
213     .loc[:, inputdataset.columns]
214
215     return reshaped
216 #split_genre = split_column_genre(dataset_)
217 #genreonly = split_genre['genre']
218 #genreonly = genreonly.astype(str)
219 #np_genre = np.array(genreonly).astype(str)
220 #text = " ".join(np_genre).lower()
221
222 # function create wordcloud , input single text
223 def generateWordCloud(text , pathToSave, title):
224     wordcloud = WordCloud(repeat=False,collocations=False , background_color='white').generate(text)
225     #display the generate image
226     plt.title(title)
227     plt.imshow(wordcloud , interpolation='bilinear')
228     plt.axis('off')
229     plt.savefig(pathToSave)
230     #plt.show()
231     plt.clf()
232
233
234 #call wordcloud
235 #generateWordCloud(text , 'genre')
236
237 # =====
238 #
239 # =====
240 def multiBoxPlot(title , isVertical, yData , xLabels,xData):
241
242     fig,ax = plt.subplots()
243     plt.xlim(-1,1)
244     plt.title(title)
245     plt.ylabel(yLabels)
246     plt.xlabel(xLabels)
247     ax.set_yticklabels(yData)
248
249     ax.boxplot(xData , vert=isVertical)
250     plt.show()
251     plt.clf()
252
253 def saveMultiBoxPlot(title , isVertical, yData ,yLabels , xLabels,xData, dirToSave ):
254
255     fig,ax = plt.subplots()
256     plt.xlim(-1,1)
257     plt.title(title)
258     plt.ylabel(yLabels)
259     plt.xlabel(xLabels)
260     ax.set_yticklabels(yData)
261
262     ax.boxplot(xData , vert=isVertical)
263     plt.savefig(dirToSave)

```

264| plt.clf()

Listing A.11: regressionPerClusterModelling.py

```

1# -*- coding: utf-8 -*-
2"""
3Created on Sat Mar 14 07:40:58 2020
4
5@author: Teuku Hashrul
6"""
7import pickle
8from scipy.stats.stats import pearsonr
9from sklearn.linear_model import LinearRegression
10from sklearn.preprocessing import PolynomialFeatures
11from sklearn.tree import DecisionTreeRegressor
12import os
13import shutil
14import pandas as pd
15import numpy as np
16import plotModule
17#experiment to train all the cluster
18# since 160 is better cluster, we assigned to the classification
19
20clustered_dataframe = pd.read_csv('agglo\\agglo_n180\\clusteredactors180_dataset.csv')
21distinct_label = clustered_dataframe[['label']].drop_duplicates()
22
23folderName = 'trained_modelPerCluster'
24if os.path.exists(folderName):
25    shutil.rmtree(folderName)
26os.makedirs(folderName)
27for labelNow in distinct_label.label:
28
29    dataLabelNow = clustered_dataframe[clustered_dataframe['label'] == labelNow]
30    # create directory for this label for every purpose
31    thisLabelFolderName = 'actor-Clustered-' + str(labelNow)
32    thisLabelDirPath = folderName+'\\'+thisLabelFolderName
33    if os.path.exists(thisLabelDirPath):
34        shutil.rmtree(thisLabelDirPath)
35    os.makedirs(thisLabelDirPath)
36
37    #save the data
38    feature_list = ['runtime', 'rating' , 'votes' , 'metascore', 'us_budget']
39
40
41    name_DataLabelNow = 'actor-'+labelNow+'_dataset.csv'
42    dataLabelNow.to_csv(thisLabelDirPath+'\\'+name_DataLabelNow)
43
44    response_list = ['revenue', 'roi', 'profit']
45    for response in response_list:
46        thisResponseLabelDirPath = thisLabelDirPath + '\\\\' + response
47        if os.path.exists(thisResponseLabelDirPath):
48            shutil.rmtree(thisResponseLabelDirPath)
49        os.makedirs(thisResponseLabelDirPath)
50
51        this_response      = dataLabelNow[[response]]
52        y_arr = np.array(this_response.values)
53
54        #create dataframe to store best pearson score
55        pearson_feature_rank = pd.DataFrame({'feature': [] , 'score': []})
56
57        for feature in feature_list:
58
59            this_feature      = dataLabelNow[[feature]]
60            x_arr = np.array(this_feature.values)
61
62            # create model
63            thisRegressionModel = LinearRegression()
64            thisRegressionModel.fit(x_arr , y_arr)
65
66            # save model
67            modelName          = 'actor-label-'+str(labelNow)+'-'+feature+'-LinearRegression.sav'
68            modelNameDirPath = thisResponseLabelDirPath + '\\\\' + modelName
69            pickle.dump(thisRegressionModel, open(modelNameDirPath, 'wb'))
70
71            polynomial_features= PolynomialFeatures(degree=2)
72            x_poly = polynomial_features.fit_transform(x_arr)
73
74            thisPolynomModel = LinearRegression()
75            thisPolynomModel.fit(x_poly, y_arr)
76
77            polynomModelFileName = 'actor-label-'+str(labelNow)+'-'+feature+'-PolynomRegression.sav'
78            polynomModelDirPath = thisResponseLabelDirPath + '\\\\' + polynomModelFileName
79            pickle.dump(thisPolynomModel, open(polynomModelDirPath, 'wb'))
80
81
82
83    # Scatter plot every feature with the revenue
84    if(dataLabelNow.shape[0] > 1):
85        scatterfeature_responseDirPath = thisResponseLabelDirPath +'\\actor-label-'+str(labelNow)+ '-' +feature+ '-' +
86            response+'scatterPlot.jpg'
87        plotModule.scatterplot(x_arr , y_arr , feature , response,scatterfeature_responseDirPath)
88
89    # count pearson score
90    pearsonfeatureNow = pearsonr(dataLabelNow[feature] , dataLabelNow[response])[0]
91    pearson_feature_rank = pearson_feature_rank.append({'feature':feature , 'score':pearsonfeatureNow} , ignore_index
92            = True)

```

```

93     # create multi feature model
94     multiFeature = dataLabelNow[feature_list]
95     multi_arr = np.array(multiFeature.values)
96     decTree_Regressor = DecisionTreeRegressor(random_state = 0,max_depth =3)
97     decTree_Regressor.fit(multi_arr , y_arr)
98     decTreeRegFileName = 'actor-label'+str(labelNow)+ '-multifeature-DecisionTreeRegressor.sav'
99     decTreeRegDirPath = thisResponseLabelDirPath + '\\\\' + decTreeRegFileName
100    pickle.dump(decTree_Regressor , open(decTreeRegDirPath, 'wb'))
101
102
103
104
105    pearsonFeatureScoreFileName = 'actor-label'+ str(labelNow)+ 'pearsonScoreComparison.csv'
106    pearson_feature_rank.to_csv(thisResponseLabelDirPath+'\\\\'+pearsonFeatureScoreFileName)

```

Listing A.12: regressionPerClusterPredictAttempt.py

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Mar 14 08:38:09 2020
4  @author: Teuku Hashrul
5  """
6
7
8
9  import pandas as pd
10 import numpy as np
11 from sklearn.preprocessing import PolynomialFeatures
12 from sklearn.neighbors import KNeighborsClassifier
13 import pickle
14 from sklearn.metrics import mean_squared_error, r2_score
15 import plotModule
16 import os
17 import shutil
18
19 test_Data = pd.read_csv('imdb_finalTestDataSet.csv')
20 train_Data = pd.read_csv('imdb_finalTrainingDataSet.csv')
21
22 clustered_data = pd.read_csv('agglo\\agglo_n180\\clusteredactors180_dataset.csv')
23 clustered_data_labelOnly = clustered_data[['title', 'label']]
24
25 merged_trainData = pd.merge(train_Data,clustered_data_labelOnly,how='left' , on='title')
26
27 deleting_train = ['Unnamed:_0.1.1','Unnamed:_0', 'Unnamed:_0.1', 'title', 'genre', 'description',
28   'director', 'actors', 'year', 'runtime', 'rating', 'votes',
29   'revenue', 'metascore', 'budget', 'us_budget', 'label', 'profit', 'roi']
30 deleting_test = ['Unnamed:_0.1.1','Unnamed:_0', 'Unnamed:_0.1', 'title', 'genre', 'description',
31   'director', 'actors', 'year', 'runtime', 'rating', 'votes',
32   'revenue', 'metascore', 'budget', 'us_budget', 'profit', 'roi']
33
34 # =====
35 # CLASSIFICATION ATTEMPT TO FIND GOOD CLUSTER FOR EVERY TEST DATA
36 # =====
37
38 train_actorPredictor = merged_trainData.drop(deleting_train , axis =1)
39 train_response      = merged_trainData[['label']]
40
41 test_actorPredictor = test_Data.drop(deleting_test , axis=1)
42 KNN_model_iris = KNeighborsClassifier( n_neighbors = 5, metric = 'euclidean')
43
44 #Train the model using the training sets
45 KNN_model_iris.fit(train_actorPredictor , train_response)
46
47 #Predict the response for test dataset
48 y_pred = KNN_model_iris.predict(test_actorPredictor)
49
50 test_Data['label'] = y_pred
51
52
53 response_list=['revenue' , 'profit', 'roi']
54 # result -> 3 response ->
55 resultFolder = 'result'
56 if os.path.exists(resultFolder):
57   shutil.rmtree(resultFolder)
58 os.makedirs(resultFolder)
59 for response in response_list:
60   thisResponseFolderName = resultFolder + '\\\\' + response
61   if os.path.exists(thisResponseFolderName):
62     shutil.rmtree(thisResponseFolderName)
63   os.makedirs(thisResponseFolderName)
64
65 # =====
66 # REGRESSION ATTEMPT ON EVERY TEST DATA
67 # =====
68 predictionResult_VotesDataFrame = pd.DataFrame({"title":[] , "votes":[] , response:[], response+"_predicted_linear":[] ,
69   "intercept":[], "coef":[], "label":[], response+"_predicted_polytom":[]})
70
71 predictionResult_MetascoreDataFrame = pd.DataFrame({"title":[] , "metascore":[] , response:[], response+"_predicted_linear":[] ,
72   "intercept":[], "coef":[], "label":[], response+"_predicted_polytom":[]})
73
74 predictionResult_RatingDataFrame = pd.DataFrame({"title":[] , "rating":[] , response:[], response+"_predicted_linear":[] ,
75   "intercept":[], "coef":[], "label":[], response+"_predicted_polytom":[]})
76
77 predictionResult_RuntimeDataFrame = pd.DataFrame({"title":[] , "runtime":[] , response:[], response+"_predicted_linear":[] ,
78   "intercept":[], "coef":[], "label":[], response+"_predicted_polytom":[]})
79
80 predictionResult_BestDataFrame = pd.DataFrame({"title":[] , "feature":[], "value":[] , "Revenue":[] , response+"_
81   _predicted_linear":[] ,
82   "intercept":[], "coef":[], "label":[], response+"_predicted_polytom":[]})

```

```

81 predictionResult_DecTreeReg = pd.DataFrame({"title":[] , "revenue":[] , response+"_predicted":[] , "label":[]})
82
83 test_AllIndex = test_Data.index.values
84 for indexNow in test_AllIndex:
85
86     print("predicting:_Data" + str(indexNow)+ 'for_response:_' + response)
87     test_DataNow = test_Data.index[indexNow]
88
89     movieTitleNow = test_Data.title[indexNow]
90     # real revenue
91     movieResponseNow = test_Data[response][indexNow]
92     movieLabelNow = test_Data.label[indexNow]
93     #predictor features
94     movieBudgetNow = test_Data.us_budget[indexNow]
95     movieVotesNow = test_Data.votes[indexNow]
96     movieMetascoreNow = test_Data.metascore[indexNow]
97     movieRatingNow = test_Data.rating[indexNow]
98     movieRuntimeNow = test_Data.runtime[indexNow]
99
100    #FOR LINEAR REGRESSION
101    #  $Y = a + b (feature)$ 
102    #  $a = \text{intercept}$ 
103    #  $b = \text{coef}$ 
104
105    # =====
106    #VOTES ATTEMPT
107    # =====
108    # take the model from the loaded model
109    dirFileVotesName = "trained_modelPerCluster\\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+str(
110        movieLabelNow)+"-votes' + "-LinearRegression.sav"
111    loaded_linearRegressionVotesModel = pickle.load(open(dirFileVotesName, 'rb'))
112    intercept_ = loaded_linearRegressionVotesModel.intercept_[0]
113    coef_ = loaded_linearRegressionVotesModel.coef_[0,0]
114    #predict attempt
115    moviePredictedResponseNow = loaded_linearRegressionVotesModel.predict([[movieVotesNow]])[0,0]
116
117    ## read polynomia features
118    dirPolynomVotesfileName = "trained_modelPerCluster\\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"++
119        str(movieLabelNow)+"-votes' + "-PolynomRegression.sav"
120    loaded_polynomRegressionModel = pickle.load(open(dirPolynomVotesfileName, 'rb'))
121    polynomial_features= PolynomialFeatures(degree=2)
122    x_poly = polynomial_features.fit_transform([[movieVotesNow]])
123    y_poly_pred = loaded_polynomRegressionModel.predict(x_poly)[0,0]
124
125    predictionResult_VotesDataFrame = predictionResult_VotesDataFrame.append({"title":movieTitleNow , "votes":movieVotesNow,
126        response:movieResponseNow, response+"_predicted_linear":moviePredictedResponseNow
127        , "intercept":intercept_ , "coef":coef_ , "label":movieLabelNow, response+"_predicted_polynom":y_poly_pred} , ignore_index =
128        True)
129
130    # =====
131    # METASCORE ATTEMPT
132    # =====
133    # take the model from the loaded model
134    dirFileMetascoreFileName = "trained_modelPerCluster\\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"++
135        str(movieLabelNow)+"-metascore' + "-LinearRegression.sav"
136    loaded_linearRegressionMetascoreModel = pickle.load(open(dirFileMetascoreFileName, 'rb'))
137    intercept_ = loaded_linearRegressionMetascoreModel.intercept_[0]
138    coef_ = loaded_linearRegressionMetascoreModel.coef_[0,0]
139    #predict attempt
140    moviePredictedResponseNow = loaded_linearRegressionMetascoreModel.predict([[movieMetascoreNow]])[0,0]
141
142    ## read polynomia features
143    dirPolynomMetascorefileName = "trained_modelPerCluster\\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"++
144        str(movieLabelNow)+"-metascore' + "-PolynomRegression.sav"
145    loaded_polynomRegressionModel = pickle.load(open(dirPolynomMetascorefileName, 'rb'))
146    polynomial_features= PolynomialFeatures(degree=2)
147    x_poly = polynomial_features.fit_transform([[movieMetascoreNow]])
148    y_poly_pred = loaded_polynomRegressionModel.predict(x_poly)[0,0]
149
150    predictionResult_MetascoreDataFrame = predictionResult_MetascoreDataFrame.append({"title":movieTitleNow , "metascore":movieMetascoreNow,
151        response:movieResponseNow, response+"_predicted_linear":moviePredictedResponseNow
152        , "intercept":intercept_ , "coef":coef_ , "label":movieLabelNow, response+"_predicted_polynom":y_poly_pred} , ignore_index =
153        True)
154
155    # =====
156    # RATING ATTEMPT
157    # =====
158    # take the model from the loaded model
159    dirFileRatingFileName = "trained_modelPerCluster\\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+str(
160        movieLabelNow)+"-rating' + "-LinearRegression.sav"
161    loaded_linearRegressionRatingModel = pickle.load(open(dirFileRatingFileName, 'rb'))
162    intercept_ = loaded_linearRegressionRatingModel.intercept_[0]
163    coef_ = loaded_linearRegressionRatingModel.coef_[0,0]
164    #predict attempt
165    moviePredictedResponseNow = loaded_linearRegressionRatingModel.predict([[movieRatingNow]])[0,0]
166
167    ## read polynomia features
168    dirPolynomRatingfileName = "trained_modelPerCluster\\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"++
169        str(movieLabelNow)+"-rating' + "-PolynomRegression.sav"
170    loaded_polynomRegressionModel = pickle.load(open(dirPolynomRatingfileName, 'rb'))
171    polynomial_features= PolynomialFeatures(degree=2)
172    x_poly = polynomial_features.fit_transform([[movieRatingNow]])
173    y_poly_pred = loaded_polynomRegressionModel.predict(x_poly)[0,0]

```

```

170 predictionResult_RatingDataFrame = predictionResult_RatingDataFrame.append({"title":movieTitleNow , "rating": movieRatingNow, response:movieResponseNow, response+"_predicted_linear":moviePredictedResponseNow , "intercept":intercept_ , "coef":coef_ , "label":movieLabelNow, response+"_predicted_poly":y_poly_pred} , ignore_index = True)
171 # =====
172 # RUNTIME ATTEMPT
173 # =====
174
175 # take the model from the loaded model
176 dirFileRuntimeFileName = "trained_modelPerCluster\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+ str(movieLabelNow)+"-runtime "+ "-LinearRegression.sav"
177 loaded_linearRegressionRuntimeModel = pickle.load(open(dirFileRuntimeFileName, 'rb'))
178 intercept_ = loaded_linearRegressionRuntimeModel.intercept_[0]
179 coef_ = loaded_linearRegressionRuntimeModel.coef_[0,0]
180 #predict attempt
181 moviePredictedResponseNow = loaded_linearRegressionMetascoreModel.predict([[movieRuntimeNow]])[0,0]
182
183 ## read polynomial features
184 dirPolynomRuntimeFileName = "trained_modelPerCluster\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+ str(movieLabelNow)+"-runtime "+ "-PolynomRegression.sav"
185 loaded_polynomRegressionModel = pickle.load(open(dirPolynomRuntimeFileName, 'rb'))
186 polynomial_features= PolynomialFeatures(degree=2)
187 x_poly = polynomial_features.fit_transform([[movieRuntimeNow]])
188 y_poly_pred = loaded_polynomRegressionModel.predict(x_poly)[0,0]
189
190
191 predictionResult_RuntimeDataFrame = predictionResult_RuntimeDataFrame.append({"title":movieTitleNow , "runtime": movieRuntimeNow, response:movieResponseNow, response+"_predicted_linear":moviePredictedResponseNow , "intercept":intercept_ , "coef":coef_ , "label":movieLabelNow, response+"_predicted_poly":y_poly_pred} , ignore_index = True)
192
193 # =====
194 # DECISION TREE MULTI REGRESSOR
195 # =====
196
197 # take the model from the loaded model
198 dirFileMultiDecTreeFileName = "trained_modelPerCluster\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+ str(movieLabelNow)+"-multifeature" + "-DecisionTreeRegressor.sav"
199 loaded_DcisionTreeRegressionMultiModel = pickle.load(open(dirFileMultiDecTreeFileName, 'rb'))
200 feature_list = ['runtime', 'rating' , 'votes' , 'metascore' , 'us_budget']
201
202 multiFeature = [movieRuntimeNow , movieRatingNow , movieVotesNow , movieMetascoreNow , movieBudgetNow]
203 #predict attempt
204 moviePredictedResponseNow = loaded_DcisionTreeRegressionMultiModel.predict([multiFeature])
205 predictionResult_DecTreeReg = predictionResult_DecTreeReg.append({"title":movieTitleNow , response:movieResponseNow , response+"_predicted":moviePredictedResponseNow , "label":movieLabelNow} , ignore_index = True)
206
207 # =====
208 # BEST FEATURE COMBINED
209 # =====
210
211 # untuk tiap test data
212 # baca label kelompoknya
213 # baca dataframe best feature nya
214 # ambil urutan satu nama featurenya
215 # terus ambil model yang featurenya sama juga
216 # yaudah bikin linear sama polynom nya
217 pearsonFileDir = 'trained_modelPerCluster\actor-Clustered-' + str(movieLabelNow)+"\\\"+response +'\\"actor-label-' +str(movieLabelNow) +'pearsonScoreComparison.csv'
218 pearsonDataFrame = pd.read_csv(pearsonFileDir)
219 pearsonDataFrame = pearsonDataFrame.sort_values('score' , ascending = False)
220 thisLabelBestFeature = pearsonDataFrame.iloc[0].feature
221
222
223
224 bestLinearModelFileName = "trained_modelPerCluster\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+ str(movieLabelNow)+"-"+ thisLabelBestFeature +"-LinearRegression.sav"
225 bestModel_Linear = pickle.load(open(bestLinearModelFileName, 'rb'))
226 best_x = test_Data[thisLabelBestFeature][indexNow]
227 bestlinear_predicted = bestModel_Linear.predict([[best_x]])[0,0]
228 intercept_ = bestModel_Linear.intercept_[0]
229 coef_ = bestModel_Linear.coef_[0,0]
230
231
232 bestPolynomModelFileName = "trained_modelPerCluster\actor-Clustered-"+str(movieLabelNow)+"\\\"+response +"\\"actor-label-"+ str(movieLabelNow)+"-"+ thisLabelBestFeature +"-PolynomRegression.sav"
233 bestModel_Polynom = pickle.load(open(bestPolynomModelFileName, 'rb'))
234 x_poly = polynomial_features.fit_transform([[best_x]])
235 bestPolynom_predicted = bestModel_Polynom.predict(x_poly)[0,0]
236
237
238 predictionResult_BestDataFrame = predictionResult_BestDataFrame.append({"title":movieTitleNow , "feature": thisLabelBestFeature , "value":best_x , response:test_Data[response][indexNow] , response+"_predicted_linear": bestlinear_predicted , "intercept":intercept_ , "coef":coef_ , "label":movieLabelNow, response+"_predicted_poly":bestPolynom_predicted} , ignore_index = True)
239
240
241 # save dataframe
242
243 predictionResult_VotesDataFrame.to_csv(thisResponseFolderName+'\\\'+'predictionResult_VotesOnlyClusteredActor.csv')
244 predictionResult_MetascoreDataFrame.to_csv(thisResponseFolderName+'\\\'+'predictionResult_MetascoreOnlyClusteredActor.csv')
245 predictionResult_RatingDataFrame.to_csv(thisResponseFolderName+'\\\'+'predictionResult_RatingOnlyClusteredActor.csv')
246 predictionResult_RuntimeDataFrame.to_csv(thisResponseFolderName+'\\\'+'predictionResult_RuntimeOnlyClusteredActor.csv')
247 predictionResult_DecTreeReg.to_csv(thisResponseFolderName+'\\\'+'predictionResult_DecTreeRegressorClusteredActor.csv')
248 predictionResult_BestDataFrame.to_csv(thisResponseFolderName+'\\\'+'predictionResult_BestFeatureClusteredActor.csv')
249
250
251
252
253
254

```

```

255
256
257 # Plot 1 box and whisker 1 feature
258 # -> votes
259 # -> metascore
260 # -> rating
261 # -> runtime
262 # -> best
263 # -> DecisionTree
264 # ->
265 arrPredictionResult = [predictionResult_VotesDataFrame,
266                         predictionResult_RuntimeDataFrame,
267                         predictionResult_RatingDataFrame,
268                         predictionResult_MetascoreDataFrame,
269                         predictionResult_BestDataFrame]
270
271 featureDirName = thisResponseFolderName+'\\'+'features'
272 if os.path.exists(featureDirName):
273     shutil.rmtree(featureDirName)
274 os.makedirs(featureDirName)
275 for predictionResult in arrPredictionResult:
276     # count r2 for linear
277     allPredictedLabel = predictionResult[['label']]
278     allPredictedLabel = allPredictedLabel.drop_duplicates('label')
279
280     thisPredictionResult_PredictorName = predictionResult.keys()[1]
281     thisPredictedResultR2_DataFrameLinear = pd.DataFrame({'label':[], 'r2':[]})
282     thisPredictedResultR2_DataFramePolynom = pd.DataFrame({'label':[], 'r2':[]})
283     for labelNow in allPredictedLabel:
284         #count r2 linear for this label in this feature prediction result
285         thisLabelPredictedData = predictionResult[predictionResult['label'] == labelNow]
286         r2_linear = r2_score(thisLabelPredictedData[response] , thisLabelPredictedData[response+'_predicted_linear'])
287         thisPredictedResultR2_DataFrameLinear = thisPredictedResultR2_DataFrameLinear.append({'label':labelNow , "r2":r2_linear} , ignore_index=True)
288
289         r2_polynom = r2_score(thisLabelPredictedData[response] , thisLabelPredictedData[response+'_predicted_polynom'])
290         thisPredictedResultR2_DataFramePolynom = thisPredictedResultR2_DataFramePolynom.append({'label':labelNow , "r2":r2_linear} , ignore_index=True)
291
292     # since r2 cannot be count if there are less than 2 observation for every cluster , it will return nan so dropped it
293     thisPredictedResultR2_DataFrameLinear = thisPredictedResultR2_DataFrameLinear.dropna()
294     thisPredictedResultR2_DataFramePolynom = thisPredictedResultR2_DataFramePolynom.dropna()
295
296
297
298
299
300 #create multi box and whisker : 1 box and whisker for 1 feature linear and polynom
301 boxWhiskerThisPredictionFeature = pd.DataFrame({'model':[], 'arrayR2':[]})
302 boxWhiskerThisPredictionFeature = boxWhiskerThisPredictionFeature.append({'model':'linear', 'arrayR2':
303     thisPredictedResultR2_DataFrameLinear.r2} , ignore_index = True)
304 boxWhiskerThisPredictionFeature = boxWhiskerThisPredictionFeature.append({'model':'polynom', 'arrayR2':
305     thisPredictedResultR2_DataFramePolynom.r2} , ignore_index = True)
306
307
308 #create feature dir Name
309 thisFeatureDirName = featureDirName + '\\'+ thisPredictionResult_PredictorName
310 if os.path.exists(thisFeatureDirName):
311     shutil.rmtree(thisFeatureDirName)
312 os.makedirs(thisFeatureDirName)
313
314 thisFeatureResponseMultiBoxPlotDirName = thisFeatureDirName + '\\'+ thisPredictionResult_PredictorName+'_multiBoxPlot_'+
315     response+'_r2Score.jpg'
316 plotModule.saveMultiBoxPlot(response+'__prediction_with_'+ thisPredictionResult_PredictorName + '_r2_' + response +'_'
317     'score_box_and_whisker_\n_for_every_cluster_using_different_model',
318     False,
319     boxWhiskerThisPredictionFeature.model,
320     'Regressor',
321     'r2_Distribution',
322     boxWhiskerThisPredictionFeature.arrayR2,
323     thisFeatureResponseMultiBoxPlotDirName)
324
325 thisFeatureLinearDataFrameName = thisPredictionResult_PredictorName + '_LinearR2DataFrame.csv'
326 thisPredictedResultR2_DataFrameLinear.to_csv(thisFeatureDirName + '\\'+ thisFeatureLinearDataFrameName)
327
328 thisFeaturePolynomDataFrameName = thisPredictionResult_PredictorName + '_PolynomR2DataFrame.csv'
329 thisPredictedResultR2_DataFramePolynom.to_csv(thisFeatureDirName + '\\'+ thisFeaturePolynomDataFrameName)
330
331 print('-----')
332
333 # =====
334 # DECISION TREE REGRESSOR
335 # =====
336 # CREATE FOR DECISION TREE REGRESSOR
337 thisPredictedResultR2_DecTreeDataFrame = pd.DataFrame({'label':[], 'r2':[]})
338
339 allPredictedLabel = predictionResult_DecTreeReg[['label']]
340 allPredictedLabel = allPredictedLabel.drop_duplicates('label').label
341
342 for labelNow in allPredictedLabel:
343     #count r2 linear for this label in this feature prediction result
344     thisLabelPredictedData = predictionResult_DecTreeReg[predictionResult_DecTreeReg['label'] == labelNow]
345     r2_labelNow = r2_score(thisLabelPredictedData[response] , thisLabelPredictedData[response+'_predicted'])
```

```

348     thisPredictedResultR2_DecTreeDataFrame = thisPredictedResultR2_DecTreeDataFrame.append({'label': labelNow , 'r2':
349         r2_labelNow}, ignore_index = True)
350
351 # since r2 cannot be count if there are less than 2 observation for every cluster , it will return nan so dropped it
352 thisPredictedResultR2_DecTreeDataFrame = thisPredictedResultR2_DecTreeDataFrame.dropna()
353
354
355 #create multi box and whisker : 1 box and whisker for 1 feature linear and polynom
356 regTreeR2DirPath = featureDirName+'\\decTree'
357 if os.path.exists(regTreeR2DirPath):
358     shutil.rmtree(regTreeR2DirPath)
359 os.makedirs(regTreeR2DirPath)
360
361 regTreeR2DirPathR2FileName = regTreeR2DirPath + '\\'+ 'DecisionTreeRegressorR2Distribution.csv'
362 thisPredictedResultR2_DecTreeDataFrame.to_csv(regTreeR2DirPathR2FileName)
363
364 regTreeR2DirPathBoxplot = regTreeR2DirPath + '\\'+ 'boxPlotR2DecisionTreeRegressor.jpg'
365 thisPredictedResultR2_DecTreeDataFrame
366 plotModule.boxplot(thisPredictedResultR2_DecTreeDataFrame.r2,
367     False,
368     'Decision_Tree_Regressor_Multi_Feature\\n_R2_Score_Distribution',
369     regTreeR2DirPathBoxplot)
370

```

Listing A.13: scrappingIMDB-API.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 05:22:31 2020
4
5 @author: lenovo
6 """
7
8 # =====
9 # this experiment is using to scrap data from IMDB
10 # to complete data like budget
11 # https://imdbpy.readthedocs.io/en/latest/
12 # =====
13 import pandas as pd
14 #if new machine using the imdb library dont forget to pip install imdbpy
15 import imdb
16
17 #set up imdb py object
18 ia = imdb.IMDb()
19
20 # read dataset
21 dataset = pd.read_csv('hasileksperimen2.csv')
22
23 # create container for
24 titleAndGross = pd.DataFrame({'title':[], 'budget':[]})
25 for title in dataset.title:
26     print(title)
27     # search keyword movie
28     movies = ia.search_movie(title)
29     #get one movie
30     movieNow = ia.get_movie(movies[0].movieID)
31     #know all the possible keys , because one movie is a dictionary with key and value
32     #guardian.infoSet2keys
33     #guardian.keys()
34     budget = 0
35
36     if movieNow.get('box_office') is not None:
37
38         #get box office data liek budget and gross from object movie
39         boxOfficeData = movieNow['box_office']
40         #box office consist of 3 item gross and budget using key
41
42         if boxOfficeData.get('Budget') is not None:
43             budget = boxOfficeData['Budget']
44
45
46         print('title: '+ title + '_budget: '+ str(budget))
47         titleAndGross = titleAndGross.append({'title':title , 'budget':budget} , ignore_index = True)
48
49
50 movies = ia.search_movie('Into_The_Woods')
51     #get one movie
52 movieNow = ia.get_movie(movies[0].movieID)
53     #know all the possible keys , because one movie is a dictionary with key and value
54     #guardian.infoSet2keys
55     #guardian.keys()
56 movieNow['title']

```

Listing A.14: scrapYoutube.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 27 21:03:05 2020
4
5 @author: lenovo
6 """
7 # =====
8 # Source Tutorial
9 # activate API KEYS FIRST IN THE GOOGLE MANAGER
10 # https://www.dataquest.io/blog/python-api-tutorial/

```

```

11| # https://developers.google.com/youtube/v3/docs/search/list?api_params=%7B%22part%22%3A%22snippet%22%2C%22q%22%3A%22the%20dark%20knight%20trailer%22%7D&api_key=true
12| # =====
13| import requests
14| import json
15| import pandas as pd
16|
17| apikey = "AIzaSyBAHdAIu1-_l10NkgME_E8xbFDTce1RTw"
18| test_id = "3J6o7hcmBbE"
19| response = requests.get('https://www.googleapis.com/youtube/v3/videos?part=statistics&id=' + test_id + '&key=' + apikey)
20|
21|
22| # method to print in json object , so it will easier to read
23| def jprint(obj):
24|     # create a formatted string of the Python JSON object
25|     text = json.dumps(obj, sort_keys=True, indent=4)
26|     print(text)
27|
28| # =====
29| # {'viewCount': '144043132',
30| # 'likeCount': '1887384',
31| # 'dislikeCount': '92479',
32| # 'favoriteCount': '0',
33| # 'commentCount': '90586'}
34| # =====
35| response.json()['items'][0]['statistics']
36|
37|
38| #
39|
40| query = 'the_dark_knight_trailer'
41| search_response = requests.get('https://www.googleapis.com/youtube/v3/search?part=snippet&q=' + query + '&key=' + apikey)
42|
43| search_response.json()
44|
45|
46| jprint(search_response.json())
47|
48| search_response.json()['items'][0]['id']['videoId']
49| title = search_response.json()['items'][0]['snippet']['title']
50|
51| video_id = search_response.json()['items'][0]['id']['videoId']
52| # get all the data statsitics
53|
54| statistic_response = requests.get('https://www.googleapis.com/youtube/v3/videos?part=statistics&id=' + video_id + '&key=' + apikey)
55|
56| statistic_response.json()
57|
58| # read data frame
59| # seach get
60| # get statistics from id
61| # append to dataframe
62|
63| dataset = pd.read_csv('hasileksperiment-withROI.csv')
64| resultDataFrame = pd.DataFrame({'title':[],'youtubeTitle':[],'viewCount_Youtube':[],'likeCount_Youtube':[],'dislikeCount_Youtube':[],'commentCount_Youtube':[]})
65|
66|
67| for title in dataset.title:
68|
69|     # search video
70|     query = title + '_trailer'
71|     print('searching: ' + query)
72|     #search get id
73|
74|     search_response = requests.get('https://www.googleapis.com/youtube/v3/search?part=snippet&q=' + query + '&maxResults=1&key=' + apikey)
75|     youtubeTitle = search_response.json()['items'][0]['snippet']['title']
76|     video_id = search_response.json()['items'][0]['id']['videoId']
77|     # get all the data statsitics
78|
79|     statistic_response = requests.get('https://www.googleapis.com/youtube/v3/videos?part=statistics&id=' + video_id + '&key=' + apikey)
80|
81|
82|     viewCount = statistic_response.json()['items'][0]['statistics']['viewCount']
83|     likeCount = 0
84|     dislikeCount = 0
85|     commentCount = 0
86|     if 'likeCount' in statistic_response.json()['items'][0]['statistics']:
87|         likeCount = statistic_response.json()['items'][0]['statistics']['likeCount']
88|     if 'dislikeCount' in statistic_response.json()['items'][0]['statistics']:
89|         dislikeCount = statistic_response.json()['items'][0]['statistics']['dislikeCount']
90|     if 'commentCount' in statistic_response.json()['items'][0]['statistics']:
91|         commentCount = statistic_response.json()['items'][0]['statistics']['commentCount']
92|
93|     resultDataFrame = resultDataFrame.append({'title':title,'youtubeTitle':youtubeTitle,'viewCount_Youtube':viewCount,'likeCount_Youtube':likeCount,'dislikeCount_Youtube':dislikeCount,'commentCount_Youtube':commentCount}, ignore_index = True)
94|
95|
96| jprint(search_response.json())
97| search_response.json()
98|
99|

```

Listing A.15: urlGenerator-youtube.py

```

1| # -*- coding: utf-8 -*-
2| """
3| Created on Sat Mar 28 19:43:07 2020

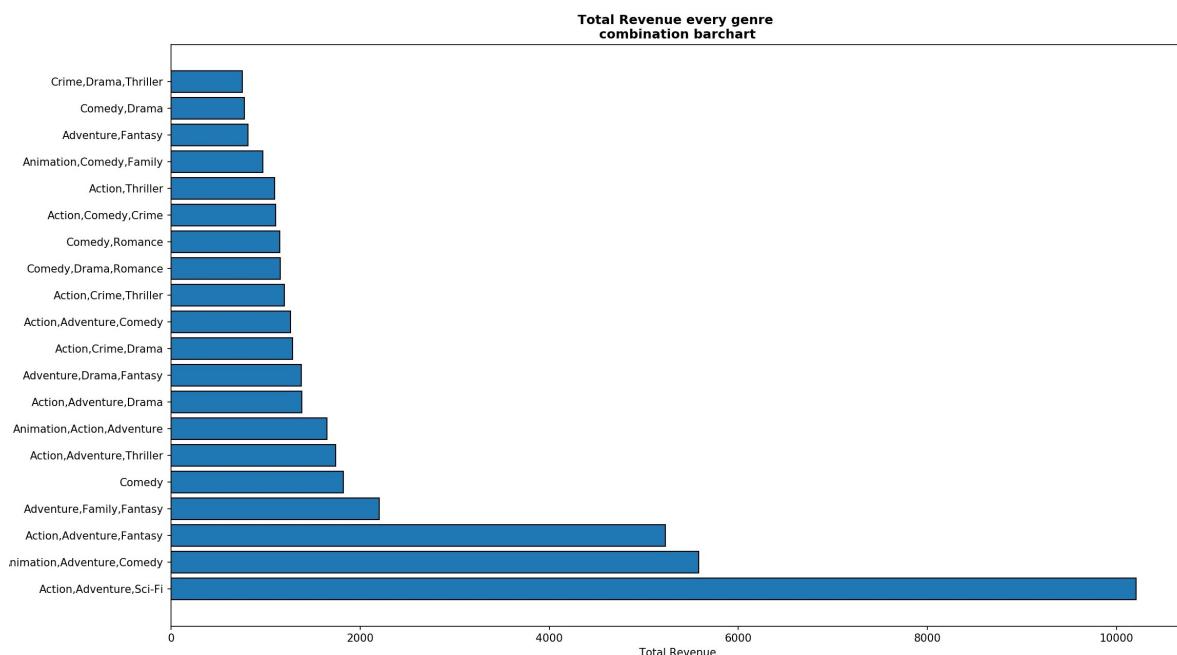
```

```
4 | @author: lenovo
5 | """
6 |
7 |
8 | import pandas as pd
9 | import seaborn as sns
10| dataset = pd.read_csv('hasileksperiment-withROI.csv')
11|
12|
13|
14| text = ''
15| for title in dataset.title:
16|     youtube_url = 'www.youtube.com/results?search_query=' + title + '_trailer'
17|     text = text + youtube_url + '\n'
18|
19|
20|
21|
22| final_Data = pd.read_excel('hasileksperiment-withYoutube.xlsx')
23|
24| corr = final_Data[['revenue', 'profit', 'roi','votes','metascore', 'runtime',
25|                    'rating', 'us_budget', 'viewCount']].corr()
26| ax = sns.heatmap(
27|     corr,
28|     vmin=-1, vmax=1, center=0,
29|     cmap=sns.diverging_palette(20, 220, n=200),
30|     square=True
31| )
32| ax.set_xticklabels(
33|     ax.get_xticklabels(),
34|     rotation=45,
35|     horizontalalignment='right'
36| );
```

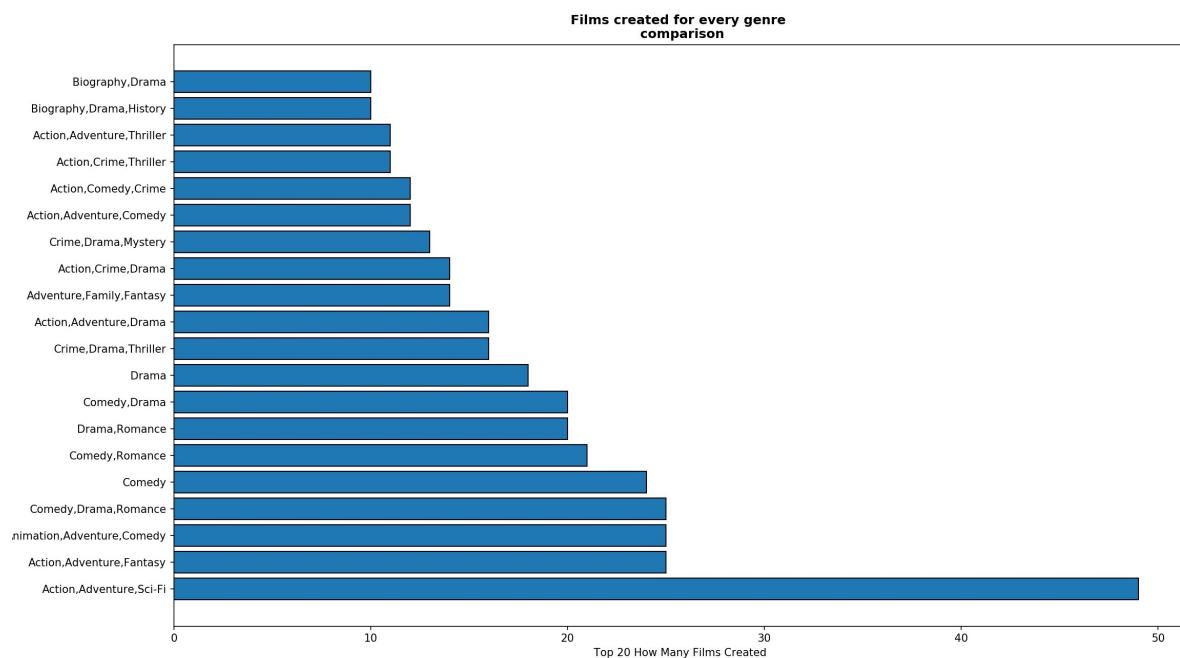


## LAMPIRAN B

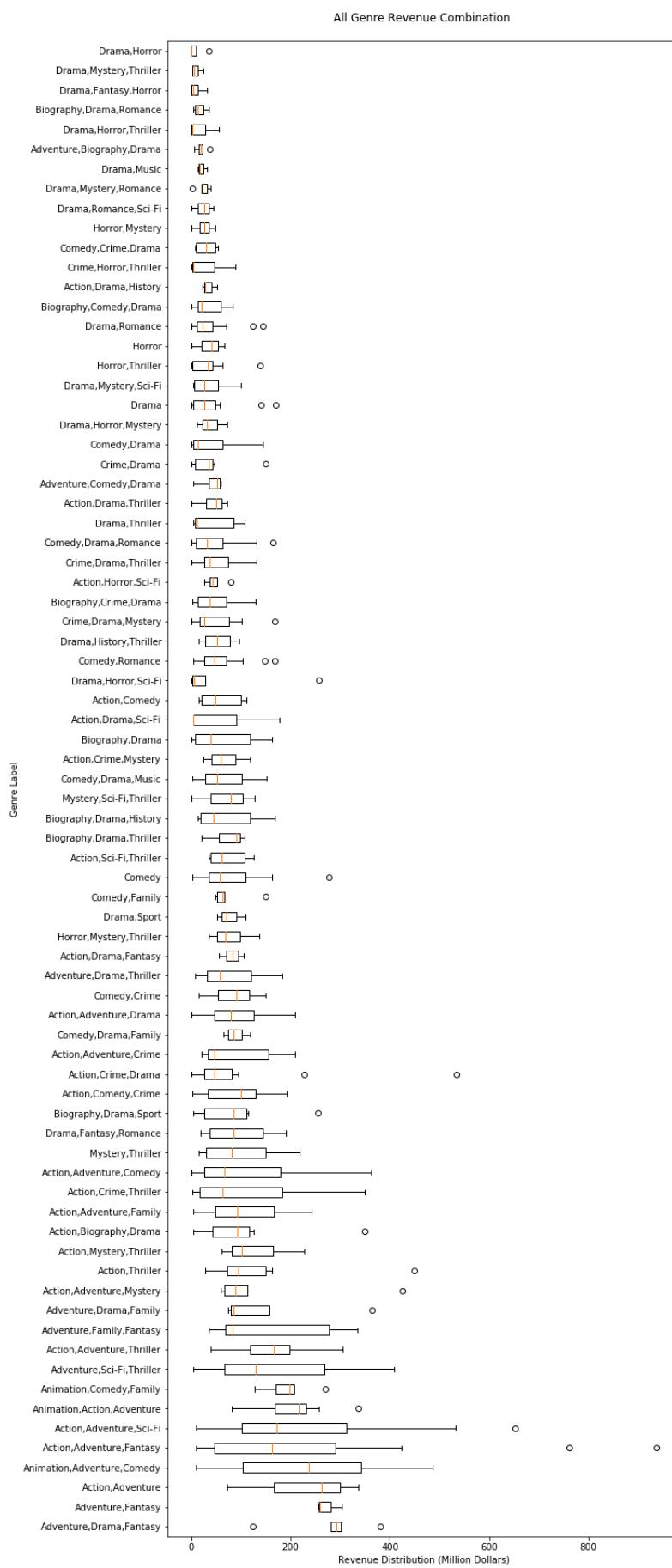
### HASIL EKSPERIMEN



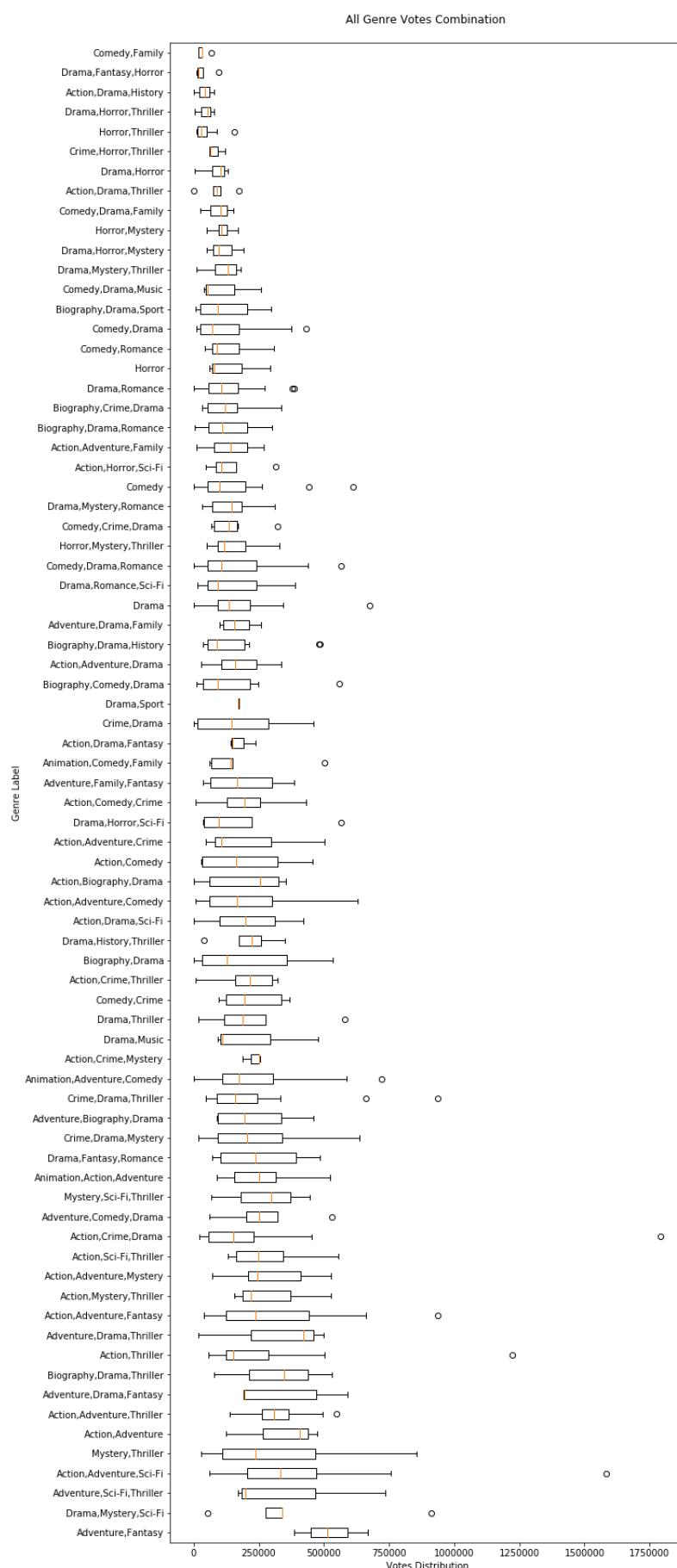
Gambar B.1: 20 Kombinasi genre dengan akumulasi *revenue* tertinggi



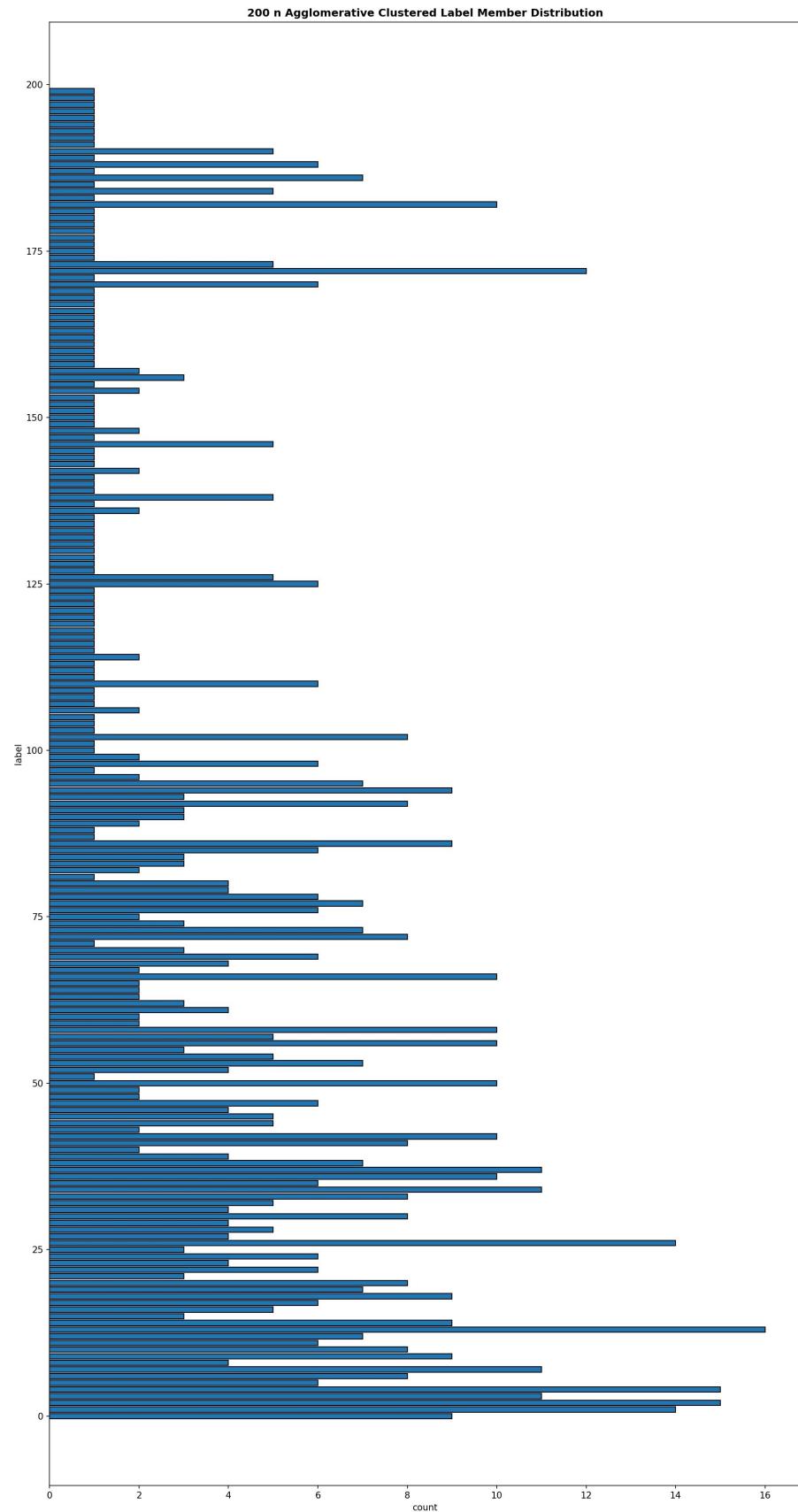
Gambar B.2: 20 Kombinasi genre yang paling banyak dibuat



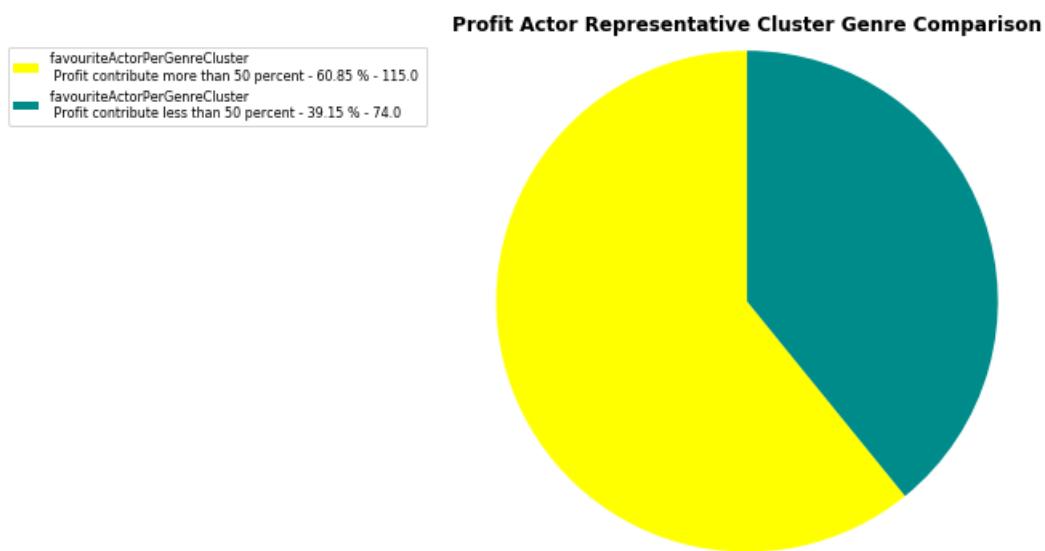
Gambar B.3: Distribusi revenue semua kombinasi genre



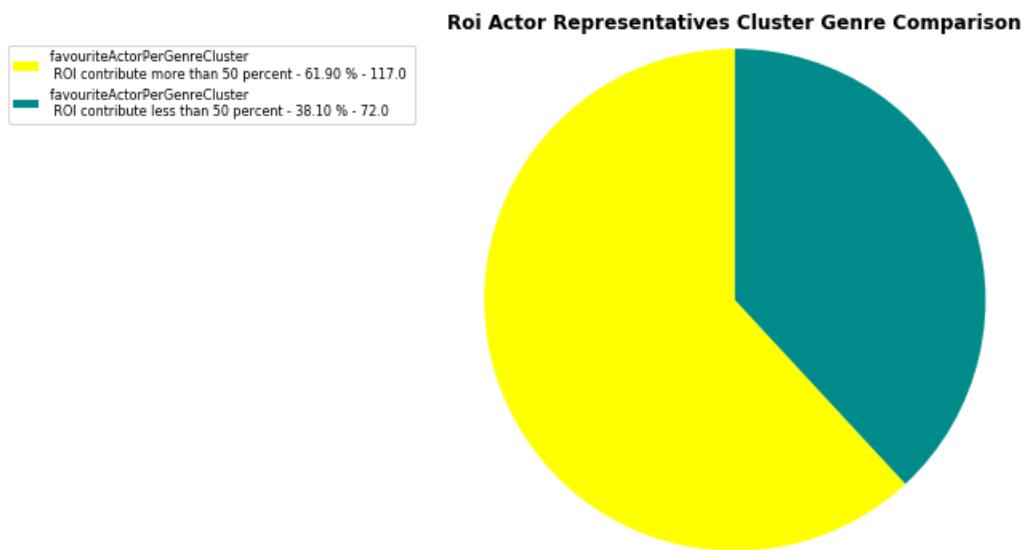
Gambar B.4: Distribusi votes semua kombinasi votes



Gambar B.5: Distribusi Jumlah anggota hasil cluster aktor



Gambar B.6: Jumlah perbandingan kelompok aktor yang genre favoritnya berkontribusi tertinggi pada *profit*



Gambar B.7: Jumlah perbandingan kelompok aktor yang genre favoritnya berkontribusi tertinggi pada *ROI*