

## **BAB IV**

### **HASIL PENELITIAN DAN PEMBAHASAN**

#### **4.1. Penelitian**

##### 1. Data

##### a. Masyarakat Desa Margo Mulyo

Desa Margo Mulyo terletak di Kecamatan Tumijajar, Kabupaten Tulang Bawang Barat, Provinsi Lampung yang merupakan salah satu Desa transmigrasi pada tahun 1983, profil Desa Margo Mulyo adalah sebagai berikut

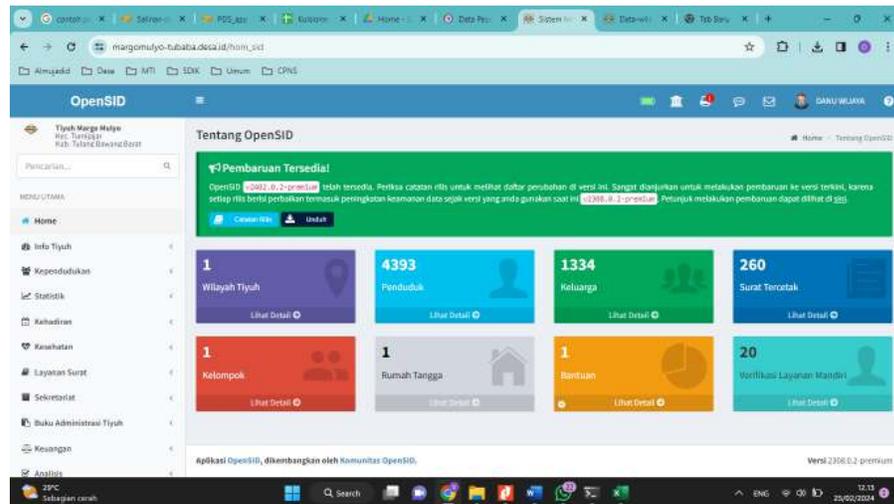
Nama Desa	: Margo Mulyo
Kepala Desa Saat ini	: Fajar Ria Kurniawan, S.T., M.M
Luas Wilayah	: 1.006 Ha
Jumlah Penduduk	: 4.393 Jiwa
	- Laki-laki : 2.234
	- Perempuan : 2.159
Jumlah RT	: 25
Jumlah RW	: 7
Status Desa Menurut IDM	: Mandiri

Sebagai salah satu Desa yang telah mendapat predikat mandiri berdasarkan Indeks Desa Membangun (IDM) yang dikeluarkan oleh Kementerian Desa, Pembangunan Daerah Tertinggal, dan Transmigrasi, Desa Margo Mulyo berupaya mewujudkan Masyarakat yang mampu untuk turut serta dalam proses digitalisasi yang ada di Desa. Untuk itu, Kepala Desa Margo Mulyo mulai menerapkan pelayanan digital sejak tahun 2021 melalui Sistem Informasi Desa dengan platform *OpenSID*.

b. Sistem Informasi Desa *OpenSID*

*OpenSID* merupakan *Platform Opensource* yang dibuat oleh pegiat desa yang ada di Indonesia, dengan perkumpulan tersebut para anggota yang memiliki keahlian dalam *programming* bekerjasama dan membentuk tim professional untuk mengembangkan *OpenSID* sampai seperti sekarang ini.

Sistem yang terintegrasi merupakan salah satu keunggulan dari *OpenSID* ini, dimana setiap data yang ada di Desa dapat juga diawasi dan menjadi laporan untuk Kecamatan dan tingkatan di atasnya sehingga jika Desa mampu menerapkan dan menggunakan *OpenSID* dengan maksimal maka seharusnya desa tersebut sangat terbantu terutama dalam pelaporan, pembuatan surat, dan tentu saja dalam upaya mewujudkan transparansi terhadap masyarakat secara umum. Berikut adalah tampilan *OpenSID* Desa Margo Mulyo



**Gambar 4.1** Sistem Informasi Desa pada *OpenSID*

Beberapa fitur dalam *OpenSID* yang memudahkan perangkat desa yaitu dalam pelaporan keuangan, pelaporan jumlah penduduk, pelaporan kegiatan pembangunan, layanan surat, dan lain sebagainya. Dan aplikasi *mobile* juga telah tersedia di *Play Store* yang dapat diunduh di gawai dan telah diunduh sebanyak 5.000 + pengguna.

c. Opini Masyarakat

Pada penelitian ini, penulis menebar kuisisioner saat proses pendataan berkala pada penduduk guna mendapatkan opini masyarakat terkait digitalisasi yang ada di Desa, dan diperoleh data sebanyak 1.514 opini. Berikut adalah gambar *google form* dan hasil dari kuisisioner

Gambar 4.2 Tampilan Google Form

A1	A	B	C	D	E
1	Timestampo	Score	Bagaimana pendapat anda dengan informasi yang diberikan melalui website dan media sosial?	Bagaimana pendapat anda dengan pelayanan pembuatan surat dengan sistem online?	
2	14/09/2023 10:43:06		Bagus tapi kurang update si, terakhir bulan 5 arkteknya		Cepat dan efisien
3	14/09/2023 10:44:40		Sangat baik, karena terbuka buat masyarakat		Sangat baik, merasa di perhatikan
4	14/09/2023 10:45:10		Bagus		Bagus, karena bisa mempermudah pola isugus, karena masyarakat bisa mandiri
5	14/09/2023 10:47:13		Bagus, karena masyarakat bisa tau informasi yg di nantikan		Puas dengan sistem online, membuat su
6	14/09/2023 10:47:00		Bagus karena dengan adanya informasi melalui website kita jadi mengerti berita dan informasi sekitar desa kita		Puas dengan sistem online, membuat su
7	14/09/2023 10:47:13		Sangat setuju, jadi lebih terbuka		Setuju, jadi lebih cepat prosesnya
8	14/09/2023 10:48:02		Bagus, karena informasi lebih cepat sampai ke masyarakat		Lebih efisien
9	14/09/2023 10:48:31		Tidak mengerti		Tidak paham online
10	14/09/2023 10:48:53		Menarik		Lebih cepat
11	14/09/2023 10:50:21		Bagus, karena dapat melihat informasi secara cepat dimana pun dan kapan pun		Sudah bagus, dengan adanya sistem or
12	14/09/2023 10:50:32		Bagus, dan tepat		Bagus dan cepat
13	14/09/2023 10:50:52		Saya belum pernah mengulunya		Saya belum pernah mencoba membuat
14	14/09/2023 10:51:07		Sangat mempermudah		Lebih cepat dan bisa mempersingkat w
15	14/09/2023 10:51:44		Tepat sasaran		Bagus, karena bisa mempercepat waktu
16	14/09/2023 10:52:32		Mempermudah mendapatkan informasi		Sangat bagus karena lebih mempermudah
17	14/09/2023 10:52:45		Bagus karena bisa menyebar luaskan informasi		Ribet, semanya serba online
18	14/09/2023 10:52:59		Belum begitu mengerti. Hanya sedikit karena keterbatasan pengetahuan tentang media sosial		Puas karena sangat efisien masalah w

Gambar 4.3 Tampilan Hasil Google Form

## 4.2. Analisis Data

### 1. Kecerdasan Buatan

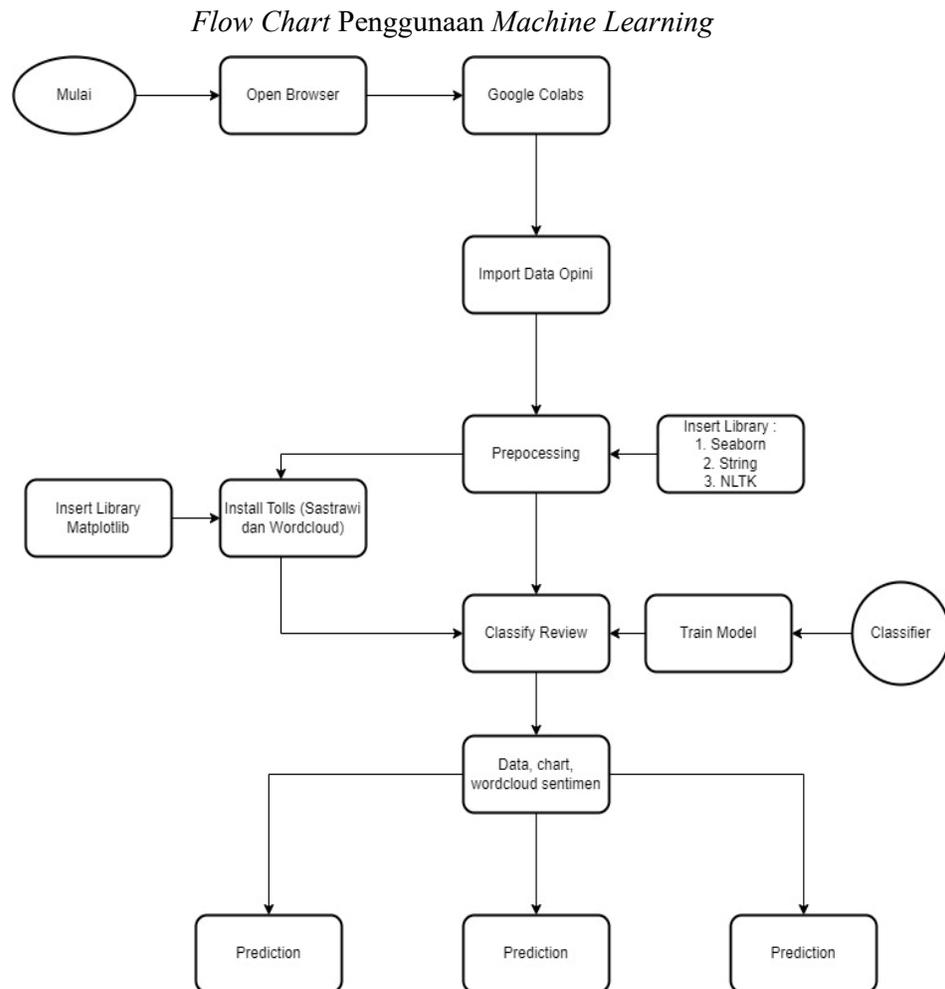
Kecerdasan buatan atau *Artificial Intelligence (AI)* adalah teknologi yang memungkinkan mesin atau komputer untuk melakukan tugas yang biasanya membutuhkan kecerdasan manusia, seperti belajar, merencanakan memecahkan masalah dan mengambil sebuah keputusan. Dalam pengembangan kecerdasan buatan, mesin atau komputer dilatih menggunakan algoritma dan data untuk meningkatkan kemampuannya dalam melakukan tugas-tugas.

### 2. *Machine Learning* dan *Natural Language Processing (NLP)*

*Machine Learning* adalah cabang dari ilmu kecerdasan buatan yang memungkinkan komputer atau mesin untuk belajar dari data dan pengalaman untuk meningkatkan kemampuannya dalam melakukan tugas tertentu tanpa perlu pemrograman. dalam hal ini, komputer dilatih menggunakan algoritma dan data untuk meningkatkan kemampuannya mengenali pola dan membuat prediksi.

*Natural language Processing (NLP)* adalah cabang kecerdasan buatan yang memungkinkan komputer atau mesin untuk memahami, menganalisis dan menghasilkan Bahasa alami manusia. NLP menggunakan algoritma dan data untuk mengajarkan komputer atau mesin memahami Bahasa manusia, termasuk tata Bahasa yang digunakan, sintaksis, dan semantik sehingga dapat memproses dan menginterpretasikan teks atau ucapan dengan cara yang lebih manusiawi dan alami.

Kedua metode diatas adalah salah satu metode yang digunakan dalam penelitian ini, dengan menggunakan AI diharapkan data yang diolah dapat memberikan prediksi atau hasil yang lebih sesuai dan tepat.



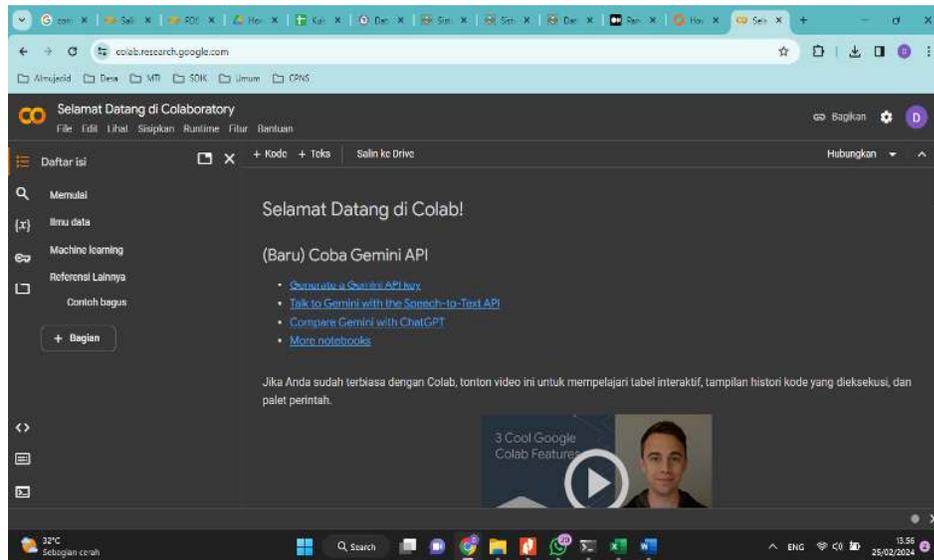
*Sumber : Olah data Penulis*

**Gambar 4.4 Flow Chart Penggunaan Machine Learning**

### 3. *Google Colabs*

*Google colabs* atau yang biasa disebut dengan *colab* merupakan layanan dari *Google* yang berbentuk *cloud computing* sehingga memungkinkan kita untuk menulis atau mengeksekusi Bahasa pemrograman *python* pada aplikasi *browser*. Layanan ini menyediakan lingkungan pengembangan terpadu (IDE) yang dapat diakses dari mana saja dengan menggunakan akses internet untuk menghubungkannya, layanan sumber daya komputasi yang diberikan juga sangat baik dalam pemrosesan grafis (GPU) dan unit pemrosesan tensor (TPU) untuk mempercepat pelatihan model *Machine Learning*.

Karena kemudahannya, siapa saja dapat mengakses kapan saja tanpa memerlukan spesifikasi komputer yang tinggi, maka penulis menggunakan *google colabs* dalam penelitian kali ini, berikut adalah tampilan awal *google colabs*

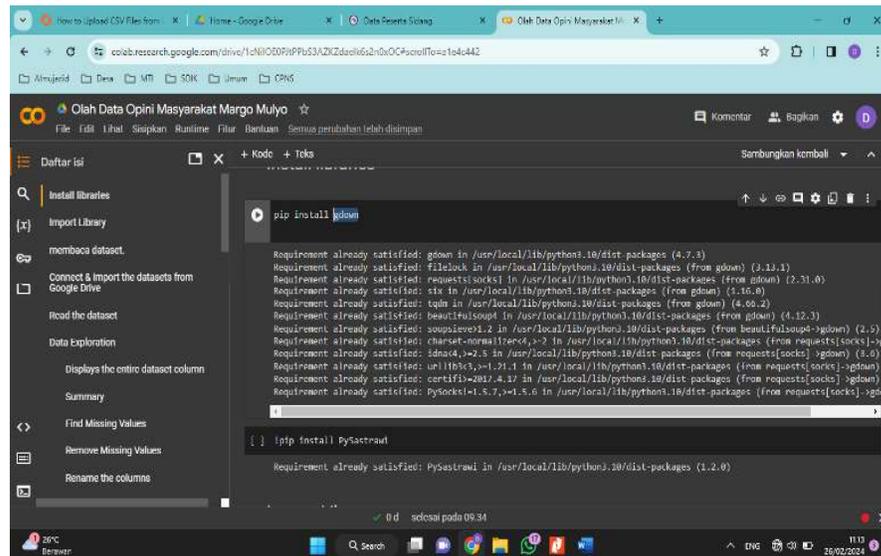


**Gambar 4.5** Laman google colabs

Dalam proses analisis data menggunakan *google colabs* ada beberapa tahapan yang harus dilakukan agar program dapat berjalan dengan baik, tahapan tersebut diantaranya

a. *Install Libraries*

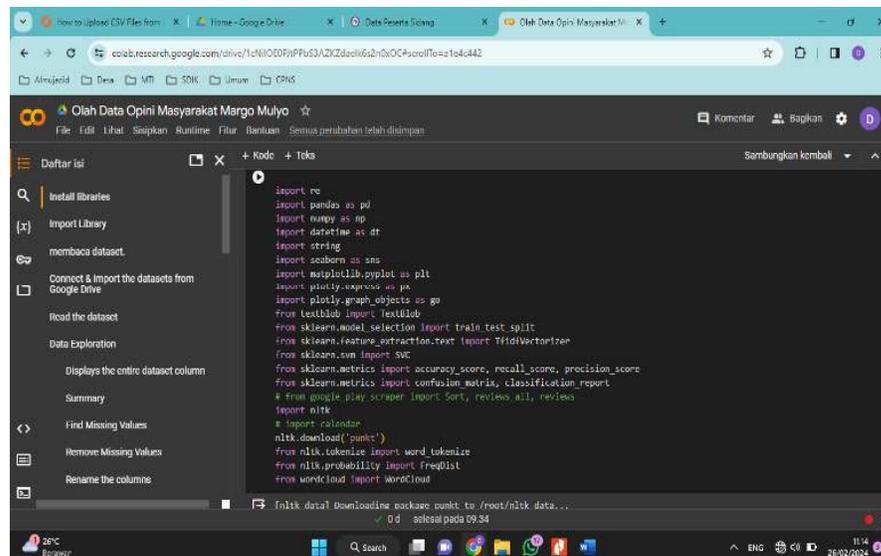
Ada beberapa *library* yang perlu diinstall untuk analisis yang akan penulis lakukan, diantaranya *gdown* dan Sastrawi. *Gdown* adalah kependekan dari *Google Drive Public File Downloader* adalah *library* yang memungkinkan *colabs* untuk mengambil data/dokumen dari *google drive* dan *library* sastrawi digunakan *colabs* untuk mengenali Bahasa Indonesia.



Gambar 4.6 Install Library

### b. Import Library

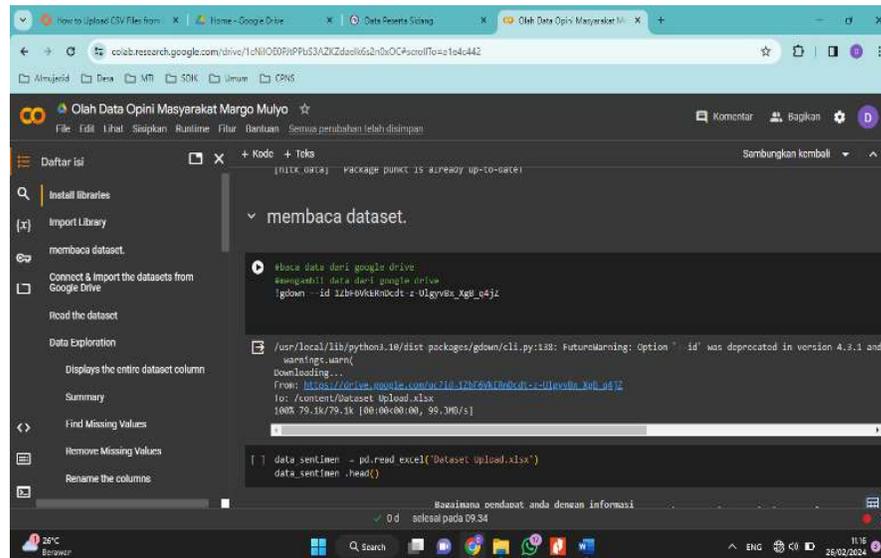
Setelah berhasil *terinstall*, Langkah selanjutnya adalah *import data library* seperti pada gambar berikut



Gambar 4.7 Import Library

c. Import Data

Untuk dapat memasukkan data hasil dari opini masyarakat ke dalam *google colabs* kita harus memasukkan data tersebut dengan perintah sebagai berikut



```

!pip install gdown
gdown --id 1zBw0vskm0cdt-z-Vlgv0x_xgr_04jz

from google.colab import drive
drive.mount('/content/drive')

!gdown --id 1zBw0vskm0cdt-z-Vlgv0x_xgr_04jz

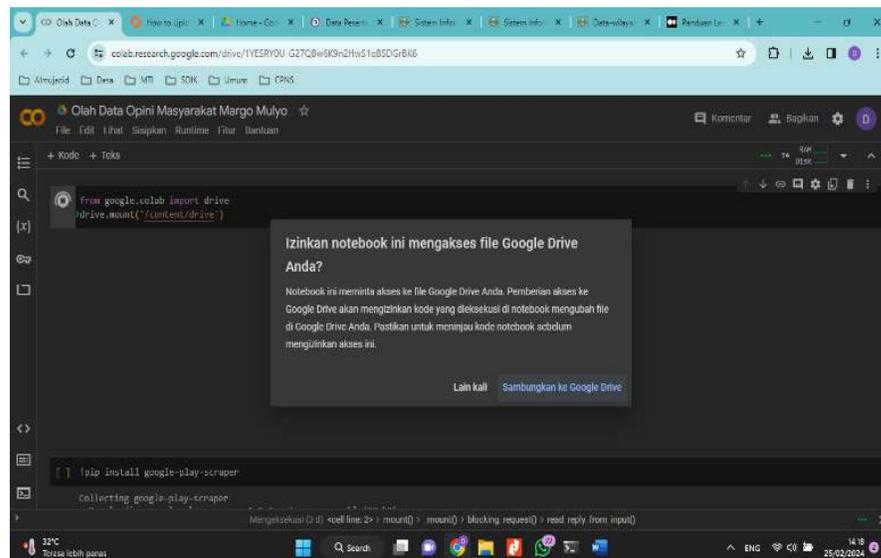
!cat /content/Dataset_upload.xlsx
100% 79.1k/79.1k [00:00<00>, 95.3MB/s]

!data_sentimen - pd.read_excel('Dataset_upload.xlsx')
data_sentimen.head()

```

**Gambar 4.8** Printah Import data

Kemudian hubungkan ke akun google drive seperti gambar berikut



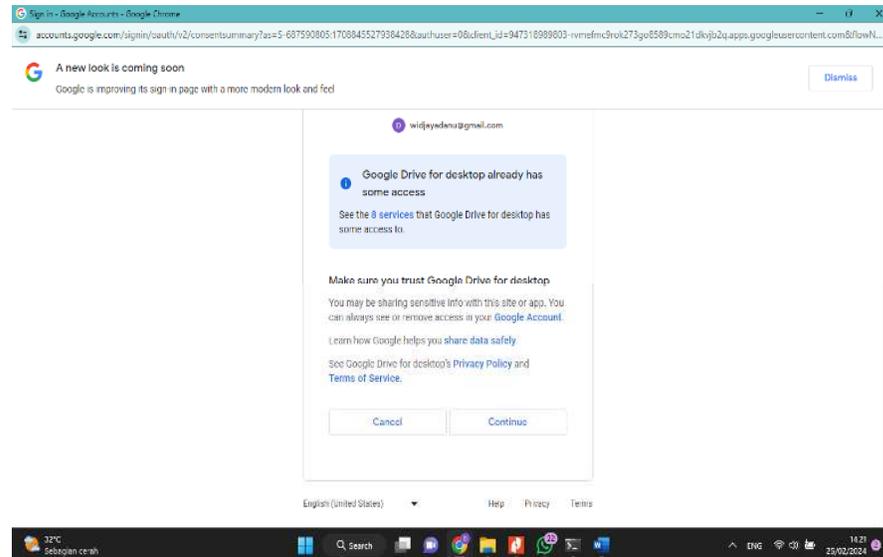
```

!pip install google-play-scraper
collecting google-play-scraper

```

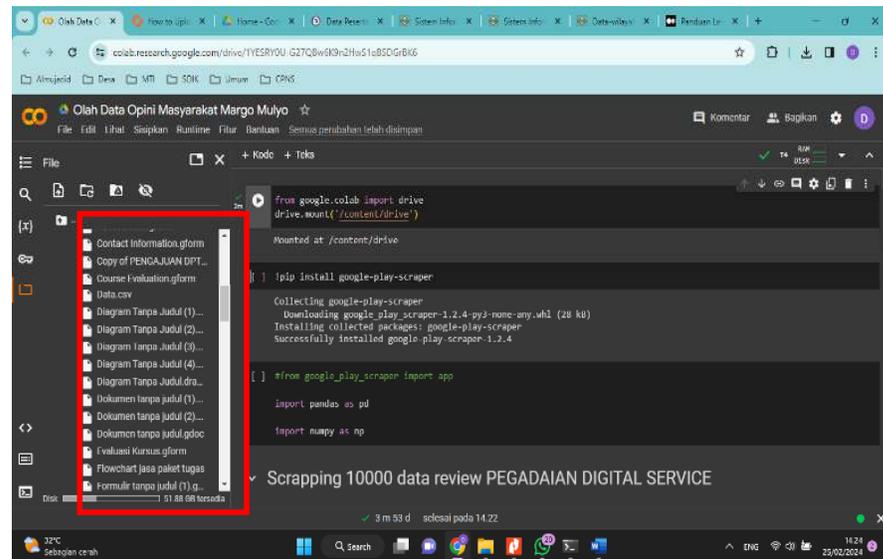
**Gambar 4.9** Colabs Meminta Ijin Akses Ke Google Drive

Kemudian *login* akun, dan berikan akses *google colabs*



**Gambar 4.10** Login Akun Google

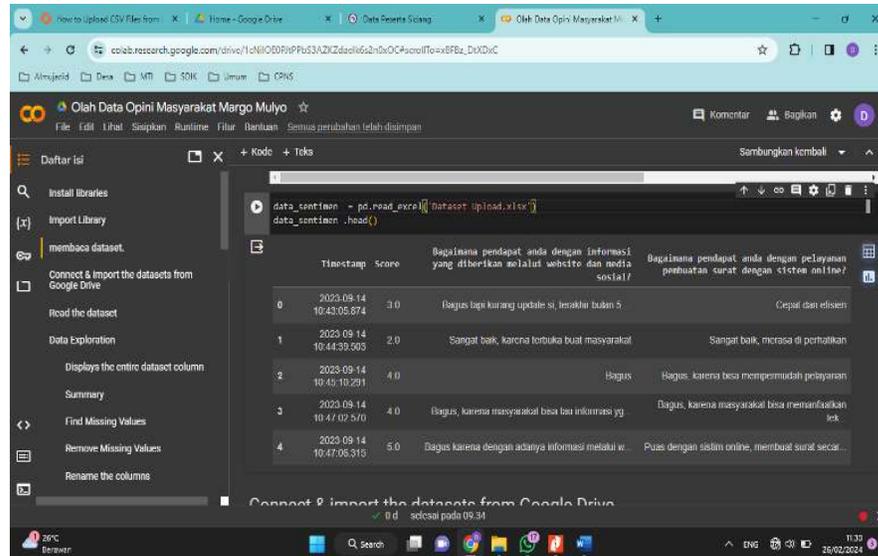
Jika berhasil maka akan tampil drive kita seperti gambar dibawah ini



**Gambar 4.11** Data Berhasil Terbaca

Setelah berhasil menghubungkan *colabs* dengan *google drive*, masukkan perintah berikut “*data\_sentimen = pd.read\_excel('Dataset Upload.xlsx')*” untuk perintah pemanggilan dataset sentimen dengan nama ‘Dataset

Upload.xlsx’ dan perintah “`data_sentimen.head()`” untuk menampilkan lima data teratas dalam file ‘Dataset Upload.xlsx’ seperti gambar berikut



Gambar 4.12 Membaca dan Menampilkan Dataset

d. Menampilkan Ringkasan Dataset

Memasukkan perintah “`info = data_sentimen.info()` `print('Ringkasan informasi:', info)`” untuk menampilkan ringkasan dataset yang berisi jumlah baris dan tipe data seperti gambar berikut

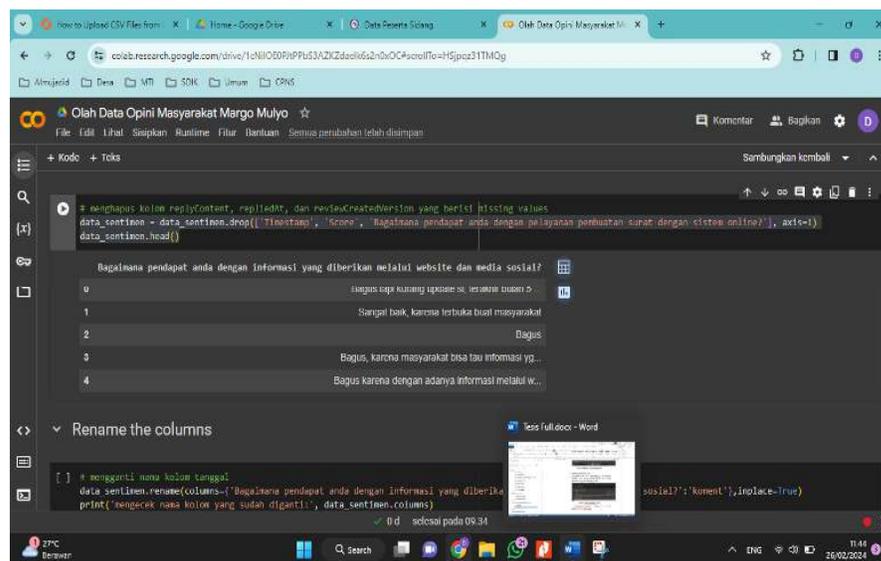


Gambar 4.12 Menampilkan Ringkasan Dataset

Berdasarkan hasil ringkasan diatas, diketahui data opini yang di isi sebanyak 1.513 yang dapat dianalisis.

e. Menghapus Kolom yang Berisi *Missing Value*

Didalam dataset biasanya terdapat data yang tidak diperlukan, bisa berisi data kosong atau kolom yang tidak diperlukan dalam proses analisis data sentimen, untuk itu perlu dilakukan penghapusan kolom yang berisi data kosong/data yang tidak diperlukan dengan memasukkan perintah berikut “`data_sentimen = data_sentimen.drop(['Timestamp', 'Score', 'Bagaimana pendapat anda dengan pelayanan pembuatan surat dengan sistem online?'], axis=1) data_sentimen.head()`” kemudian akan tampil dataset yang sudah terhapus kolomnya sesuai dengan yang kita inginkan seperti gambar berikut

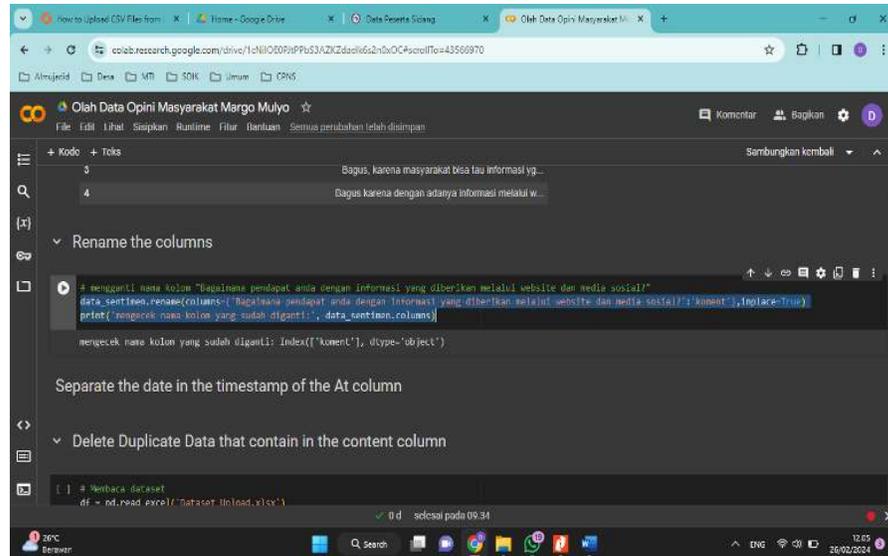


**Gambar 4.13 Menampilkan Hasil Remove Missing Dataset**

Pada perintah diatas, penulis hanya mengambil satu kolom yang akan dijadikan sebagai data analisis sentimen yaitu kolom yang berisi data opini “Bagaimana pendapat anda dengan informasi yang diberikan melalui website dan media sosial?”.

Untuk lebih rapih, penulis mengganti nama kolom agar lebih ringkas menjadi kolom koment dengan perintah sebagai berikut “`data_sentimen.rename(columns={'Bagaimana pendapat anda dengan informasi yang diberikan melalui website dan media`

`sosial?': 'koment'}, inplace=True) print('mengecek nama kolom yang sudah diganti:', data_sentimen.columns)'' seperti gambar berikut`



```

# Mengganti nama kolom "DateInena pendapat anda dengan informasi yang diberikan melalui website dan media sosial?"
data_sentimen.rename(columns={'DateInena pendapat anda dengan informasi yang diberikan melalui website dan media sosial?': 'koment'}, inplace=True)
print('mengecek nama kolom yang sudah diganti:', data_sentimen.columns)

mengecek nama kolom yang sudah diganti: Index(['koment'], dtype='object')

Separate the date in the timestamp of the At column

Delete Duplicate Data that contain in the content column

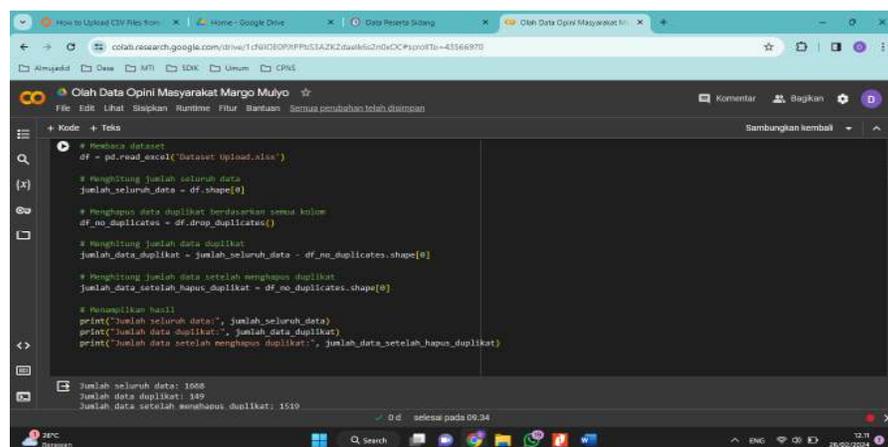
# Membaru dataset
df = pd.read_excel('dataset_upload.xlsx')

```

**Gambar 4.14 Menampilkan Hasil Mengganti Nama Kolom**

#### f. Menghapus Data Duplikat

Dalam dataset pasti terdapat data yang memiliki kesamaan atau data *duplicate* sehingga perlu dilakukan penghapusan data tersebut dengan memasukkan perintah seperti pada gambar berikut ini



```

# Membaru dataset
df = pd.read_excel('dataset_upload.xlsx')

# Menghitung jumlah seluruh data
jumlah_seluruh_data = df.shape[0]

# Menghapus data duplikat berdasarkan semua kolom
df_no_duplicates = df.drop_duplicates()

# Menghitung jumlah data duplikat
jumlah_data_duplikat = jumlah_seluruh_data - df_no_duplicates.shape[0]

# Menghitung jumlah data setelah menghapus duplikat
jumlah_data_setelah_hapus_duplikat = df_no_duplicates.shape[0]

# Menampilkan hasil
print('jumlah seluruh data:', jumlah_seluruh_data)
print('jumlah data duplikat:', jumlah_data_duplikat)
print('jumlah data setelah menghapus duplikat:', jumlah_data_setelah_hapus_duplikat)

jumlah seluruh data: 1508
jumlah data duplikat: 149
jumlah data setelah menghapus duplikat: 1510

```

**Gambar 4.15 Menghapus Data Ganda**

Berdasarkan perintah diatas maka diperoleh hasil sebagai berikut

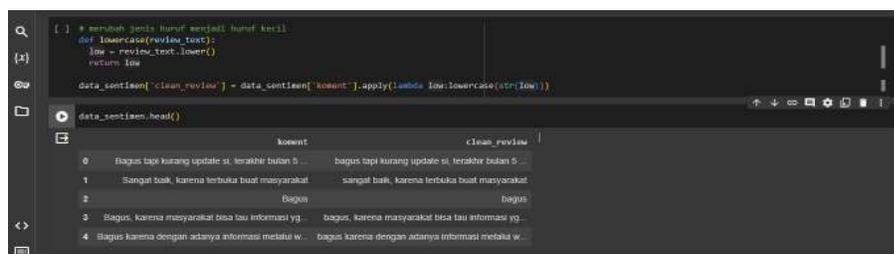
Jumlah seluruh data	: 1668
Jumlah data duplikat	: 149
Jumlah data setelah menghapus duplikat	: 1519

### 4.3. Data Preparation

Ulasan sudah bersih dari duplikasi dan siap untuk melanjutkan proses, yaitu persiapan data untuk membersihkan ulasan dari emoji, tanda baca, ketidakserasian jenis huruf, pengulangan kata, normalisasi, dan lain sebagainya.

#### 1. Case Folding

*Case Folding* adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf a sampai z yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter*. Untuk melakukannya perlu dimasukkan perintah sebagai berikut “`def lowercase(review_text): low = review_text.lower() return low` data\_sentimen ['clean\_review'] = data\_sentimen ['komentar'].apply(`lambda low:lowercase(str(low))`)” setelah itu untuk melihat hasilnya masukkan perintah “`data_sentimen.head()`” dapat dilihat pada gambar berikut



```
[ ] # membuat jadi huruf menjadi huruf kecil
def lowercase(review_text):
    low = review_text.lower()
    return low

data_sentimen['clean_review'] = data_sentimen['komentar'].apply(lambda low:lowercase(str(low)))

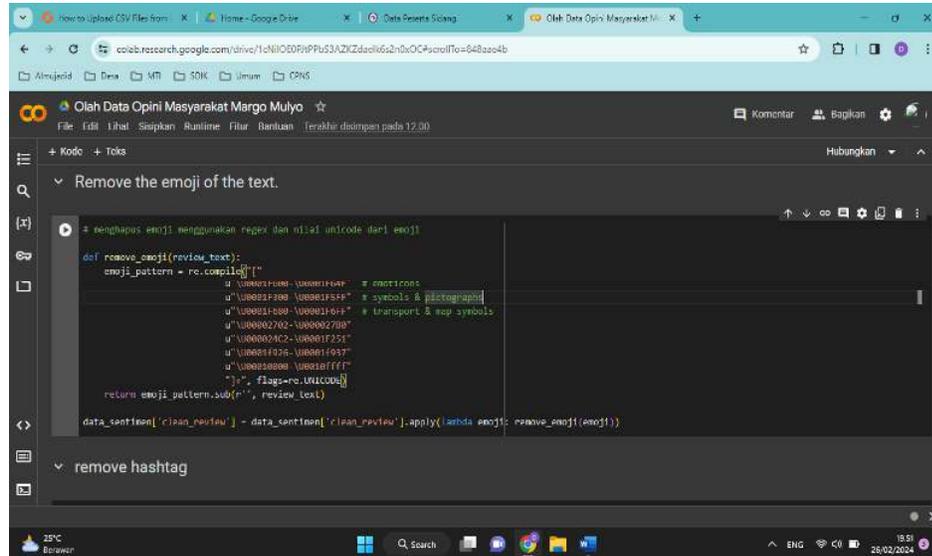
data_sentimen.head()
```

	komentar	clean_review
0	Bagus tapi kurang update si terakhir bulan 5 ...	bagus tapi kurang update si terakhir bulan 5 ...
1	Sangat baik, karena terbuka buat masyarakat	sangat baik, karena terbuka buat masyarakat
2	Bagus	bagus
3	Bagus, karena masyarakat bisa tau informasi yg ...	bagus, karena masyarakat bisa tau informasi yg ...
4	Bagus karena dengan adanya informasi melalui w ...	bagus karena dengan adanya informasi melalui w ...

Gambar 4.16 Proses *Case Folding*

## 2. *Cleaning Data*

Proses selanjutnya adalah membersihkan data yaitu dengan menghapus emoji menggunakan regex dan nilai unicode dari emoji dengan perintah seperti pada gambar berikut



```

# menghapus emoji menggunakan regex dan nilai unicode dari emoji

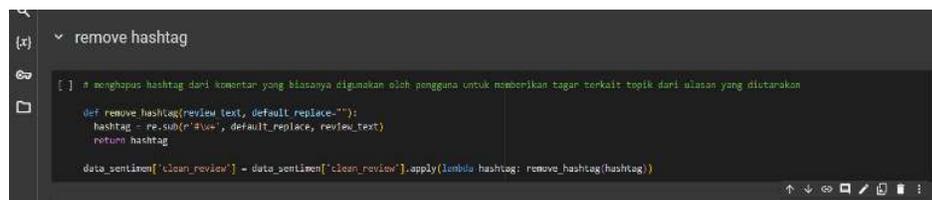
def remove_emoji(review_text):
    emoji_pattern = re.compile(
        u"[\U0001F600-\U0001F64F] # emoticons
        u"[\U0001F300-\U0001F3FF] # symbols & pictographs
        u"[\U00012000-\U000120FF] # transport & map symbols
        u"[\U00002700-\U000027BF]
        u"[\U000024C2-\U000024E9]
        u"[\U0001F920-\U0001F94F]
        u"[\U0001F6E0-\U0001F6FF]
        "]" # flags=re.UNICODE
    )
    return emoji_pattern.sub("", review_text)

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda emoji: remove_emoji(emoji))
  
```

Gambar 4.17 Proses *Cleaning Data*

## 3. *Remove Hashtag*

Menghapus hashtag dari opini yang biasanya digunakan oleh pengguna untuk memberikan tagar terkait topik dari ulasan yang diutarakan dengan perintah sebagai berikut



```

# menghapus hashtag dari komentar yang biasanya digunakan oleh pengguna untuk memberikan tagar terkait topik dari ulasan yang diutarakan

def remove_hashtag(review_text, default_replace=""):
    hashtag = re.sub("#[a-z]*", default_replace, review_text)
    return hashtag

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda hashtag: remove_hashtag(hashtag))
  
```

Gambar 4.18 Proses *Cleaning Data (Hashtag)*

#### 4. Menghapus Angka, Tanda Baca, dan *Superscript*

Dalam sebuah data, biasanya terdapat kalimat yang menggunakan angka, tanda baca, ataupun *superscript* yang tidak dibutuhkan dalam proses pengolahan data. Untuk itu perlu dilakukan pembersihan dengan menghapusnya seperti gambar berikut



```
def remove_number(review_text, default_replace=" "):
    # Hapus angka
    num = re.sub(r'\d+', default_replace, review_text)

    # Hapus URL
    url_removed = re.sub(r'http[s]?|www[s]?|https[s]?|', num, flags=re.MULTILINE)

    return url_removed

# Terapkan fungsi pada kolom 'clean_review'
data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda text: remove_number(text))

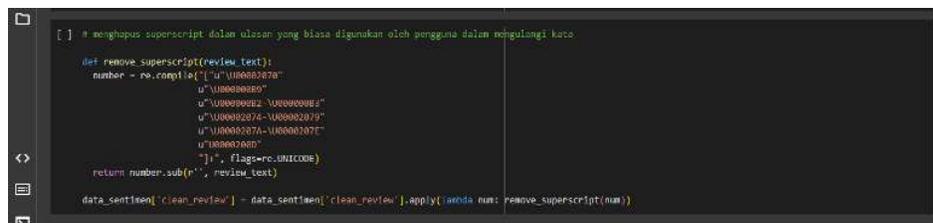
[] # menghapus tanda baca

def remove_punctuation(review_text, default_text=" "):
    list_punct = string.punctuation
    delete_punct = str.maketrans(list_punct, '' * len(list_punct))
    new_review = ' '.join(review_text.translate(delete_punct).split())

    return new_review

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda punct: remove_punctuation(punct))
```

**Gambar 4.19** Proses *Cleaning Data* (Angka dan Tanda Baca)



```
[] # menghapus superscript dalam ulasan yang biasa digunakan oleh pengguna dalam mengulangi kata

def remove_superscript(review_text):
    number = re.compile(["u\u2070-\u2079",
                        "u\u207a-\u207b",
                        "u\u207c-\u207d",
                        "u\u207e-\u207f",
                        "u\u2080-\u208f",
                        "u\u2090-\u209f",
                        "u\u209a-\u209b",
                        "u\u209c-\u209d",
                        "u\u209e-\u209f"], flags=re.UNICODE)
    return number.sub("", review_text)

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda num: remove_superscript(num))
```

**Gambar 4.20** Proses *Cleaning Data* (*Superscript*)

#### 5. Remove Words Repetition

Melakukan pembatasan jumlah huruf menjadi dua bentuk yaitu untuk mengembalikan kata ke bentuk awal dan menghindari terjadinya kata ganda yang memiliki arti sama tetapi berbeda penulisan dengan perintah seperti pada gambar

The screenshot shows a Google Colab notebook with the following code:

```

# melakukan pembatasan jumlah huruf menjadi dua untuk mengeliminasi kata ke bentuk asal
# dan menghindari terjadinya kata ganda yang memiliki arti sama tetapi berbeda penulisan
def word_repetition(review_text):
    review = re.sub(r'(\w+)\s+', r'\1', review_text)
    return review

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda word: word_repetition(word))

# melakukan pembatasan kata berulang menjadi satu kali, seperti suka suka > suka
def repetition(review_text):
    repeat = re.sub(r'(\w+)(\s+\w+)+', r'\1', review_text, flags=re.IGNORECASE)
    return repeat

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda word: repetition(word))

```

Gambar 4.21 Proses *Remove Words Repetition*

#### 6. *Remove White Space*

Meskipun telah dibersihkan, ulasan masih bisa mengandung spasi ganda yang dapat berasal dari pengguna ketika mengetik ataupun ketika melakukan pembersihan ulasan. Maka dari itu, menghapus spasi ganda diperlukan dengan memberikan perintah berikut “*def remove\_extra\_spaces(review\_text):*

```

review = re.sub(r'^s+', ' ', review_text)
return review
data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda extra_spaces: remove_extra_spaces(extra_spaces))”

```

#### 7. Membersihkan Kata-kata yang tidak Perlu

Komentar tidak hanya berisi kata yang bisa ditafsirkan atau dimengerti oleh manusia. Ada kalanya pengguna memberikan informasi berupa ketawa, atau kata abstrak yang tidak bisa/sulit diterjemahkan. Dimana kata tersebut bisa membingungkan orang yang menganalisisnya karena tidak tahu arti dari kata tersebut. Maka dari itu, Penulis melakukan pembersihan secara mendalam

dengan menghapus seluruh kata abstrak yang nantinya tidak memberikan informasi apapun kepada model dan hanya menghambat kinerja model.

Kalimat-kalimat tersebut diantaranya 'ujjy', 'dehdje', 'jwdnmb', 'wxvlezuvws', 'ircel', 'swjjw', 'dmv', 'dxrgz','qjtajuq','zzhwntdft','xywrfw', 'fztnmdrrduy', 'xsezuz', 'kbyv', 'iyyveyzqkdgm', 'arlrmdjy', 'sfarif', 'sfomar', 'nzxx', 'mtsc0','wkwk', 'wkkwk', 'wkwkwk', 'hihi', 'hihihi', 'hihihi', 'hehehe', 'hehehehe', 'hehe', huhu', 'huhuu', 'ancok', 'guak', 'cokcok', 'hmmm', 'annya', 'huftt', 'href'.



```

bannedword = ['ujjy', 'dehdje', 'jwdnmb', 'wxvlezuvws', 'ircel', 'swjjw', 'dmv', 'dxrgz', 'qjtajuq', 'zzhwntdft', 'xywrfw', 'fztnmdrrduy', 'xsezuz', 'kbyv', 'iyyveyzqkdgm', 'arlrmdjy', 'sfarif', 'sfomar', 'nzxx', 'mtsc0', 'wkwk', 'wkkwk', 'wkwkwk', 'hihi', 'hihihi', 'hihihi', 'hehehe', 'hehehehe', 'hehe', 'huhu', 'huhuu', 'ancok', 'guak', 'cokcok', 'hmm', 'annya', 'huftt', 'href']

re_banned_words = re.compile("(" + "|".join(bannedword) + ")\\w*", re.I)

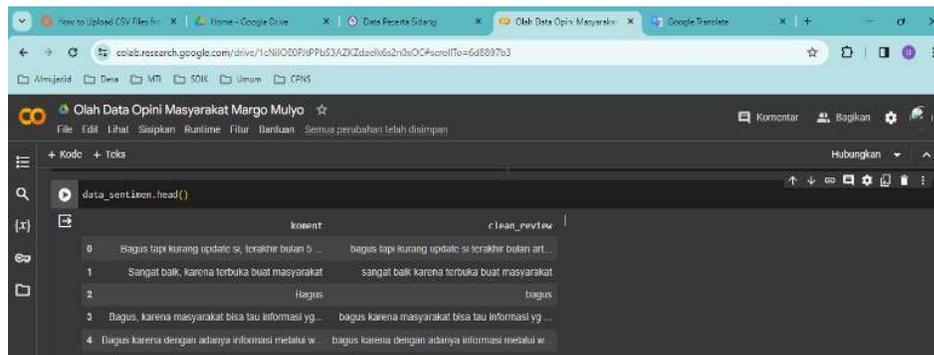
def RemoveBannedWords(toPretxt):
    global re_banned_words
    return re_banned_words.sub("", toPretxt)

data_sentimen['clean_review'] = data_sentimen['clean_review'].apply(lambda banned: RemoveBannedWords(banned))

```

**Gambar 4.22 Proses Hapus Data tidak digunakan**

Setelah melakukan pembersihan data, untuk melihat hasilnya masukkan perintah berikut “`data_sentimen.head()`” maka akan tampil lima data teratas yang telah dilakukan pembersihan

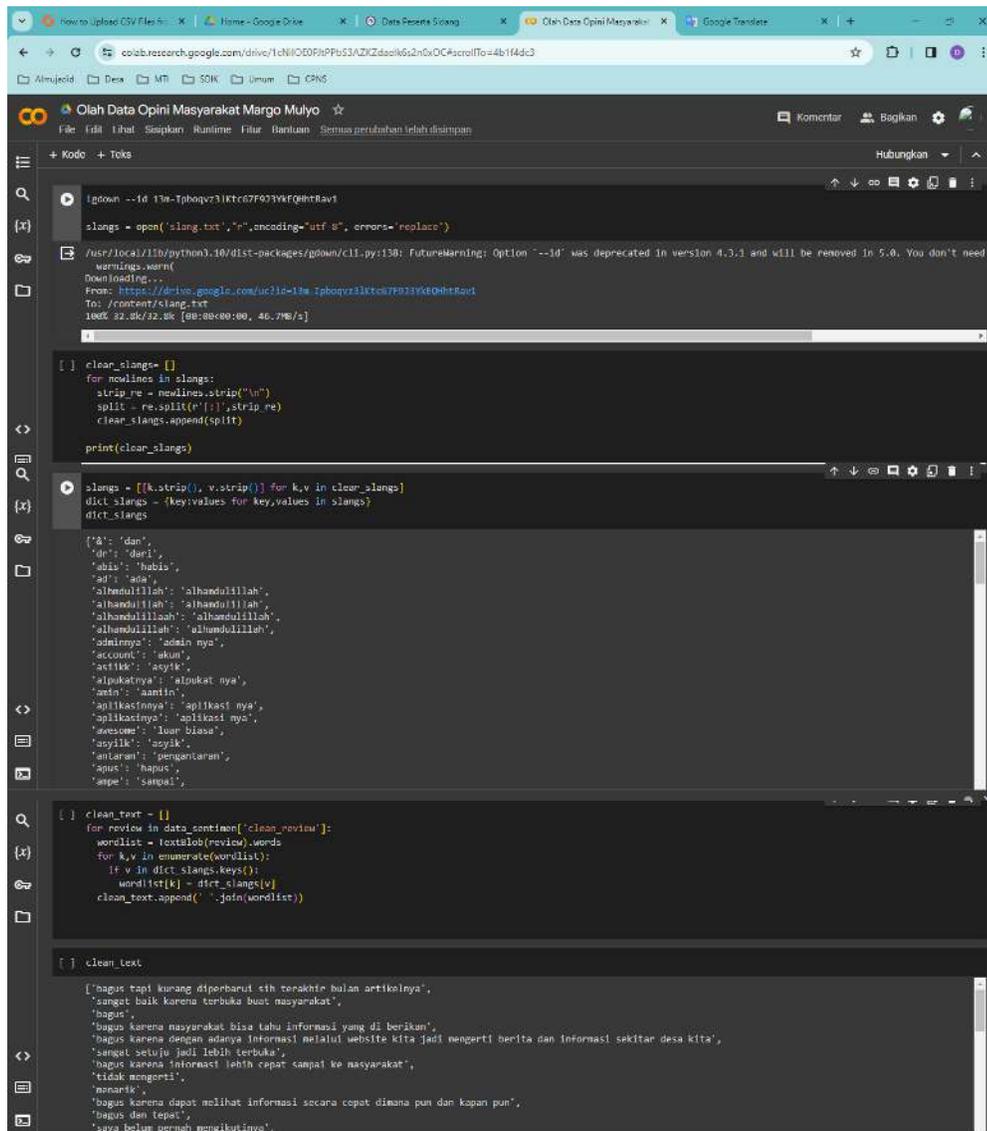


	komentar	clean_review
0	Bagus tapi kurang update si, terakhir bulan 9 ...	bagus tapi kurang update si terakhir bulan art...
1	Sangat baik, karena terbuka buat masyarakat	sangat baik karena terbuka buat masyarakat.
2	Bagus	bagus
3	Bagus, karena masyarakat bisa tau informasi yg ...	bagus karena masyarakat bisa tau informasi yg ...
4	Bagus karena dengan adanya informasi melata w	bagus karena dengan adanya informasi melata w

**Gambar 4.23 Menampilkan Data Hasil Pembersihan**

#### 4.4. Normalisasi Kata Sesuai KBBI

Dalam sebuah opini biasanya terdapat kata yang tidak menggunakan kaidah penulisan yang baik dan benar sesuai dengan Kamus Besar Bahasa Indonesia (KBBI), menormalisasikan kata singkat dan tidak baku menjadi kata baku sesuai dengan KBBI seperti perintah pada gambar berikut



```

!wget --id 13w-1pboqvz31ktr07P03YKfQHt8Av1
slangs = open('slangs.txt','r',encoding='utf-8', errors='replace')

!wget --id 13w-1pboqvz31ktr07P03YKfQHt8Av1
!cat /content/slans.txt
From: https://drive.google.com/uc?id=13w-1pboqvz31ktr07P03YKfQHt8Av1
To: /content/slans.txt
100% 22.8k/22.8k [00:00<00:00, 46.7MB/s]

[] clean_slangs = []
for newlines in slangs:
    strip_re = newlines.strip("\n")
    split = re.split(r'[!]',strip_re)
    clean_slangs.append(split)

print(clean_slangs)

slangs = [[k.strip(), v.strip()] for k,v in clean_slangs]
dict_slangs = {key:values for key,values in slangs}
dict_slangs

{'!': 'dan',
 'de': 'dari',
 'abis': 'habis',
 'ad': 'ada',
 'alhamdulillah': 'alhamdulillah',
 'alhamdulillah': 'alhamdulillah',
 'alhamdulillah': 'alhamdulillah',
 'alhamdulillah': 'alhamdulillah',
 'alhamdulillah': 'alhamdulillah',
 'admirnya': 'admir nya',
 'account': 'akun',
 'asyik': 'asyik',
 'apukatnya': 'apukat nya',
 'amin': 'aminin',
 'aplikasinya': 'aplikasi nya',
 'aplikasinya': 'aplikasi nya',
 'awesome': 'luar biasa',
 'asyik': 'asyik',
 'antaran': 'penganteran',
 'agus': 'hapus',
 'sepe': 'sempai',

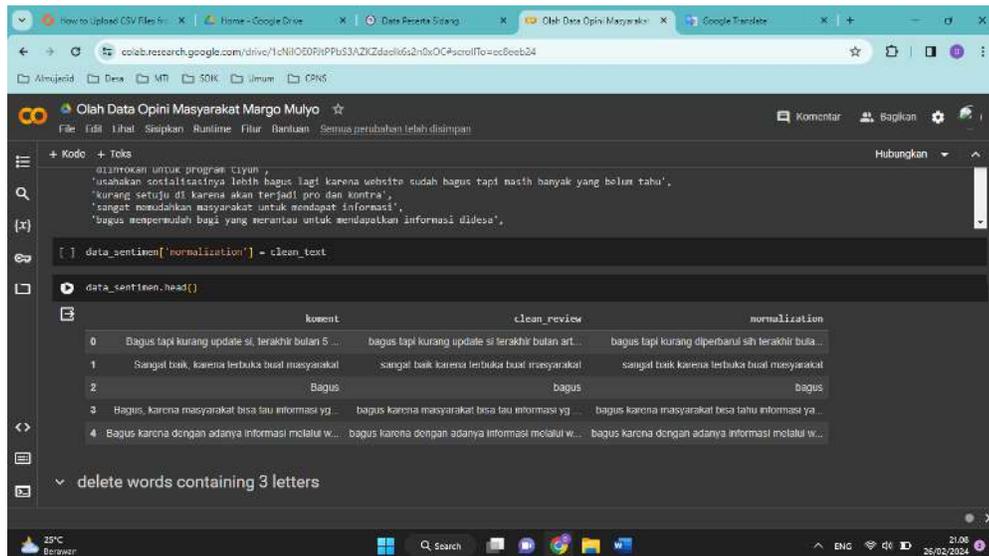
[] clean_text = []
for review in data_sentimen['clean_review']:
    wordlist = textblob(review).words
    for k,v in enumerate(wordlist):
        if v in dict_slangs.keys():
            wordlist[k] = dict_slangs[v]
    clean_text.append(' '.join(wordlist))

[] clean_text

['bagus tapi kurang diperbandi sih terakhir bulan artikalnya',
 'sangat baik karena terbuka buat masyarakat',
 'bagus',
 'bagus karena masyarakat bisa tahu informasi yang di berikan',
 'bagus karena dengan adanya informasi melalui website kita jadi mengerti berita dan informasi sekitar desa kita',
 'sangat setuju jadi lebih terbuka',
 'bagus karena informasi lebih cepat sampai ke masyarakat',
 'tidak mengerti',
 'menarik',
 'bagus karena dapat melihat informasi secara cepat dimana pun dan kapan pun',
 'bagus dan tepat',
 'saya belum pernah mengikutinya',

```

Gambar 4.24 Menormalkan Kata Sesuai KBBI

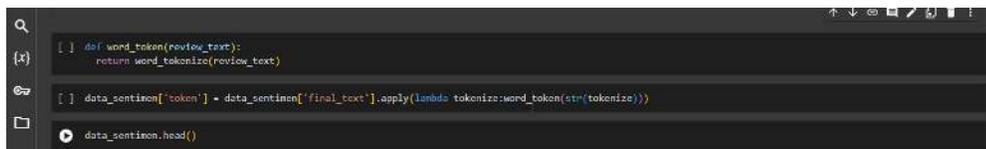


**Gambar 4.25 Menampilkan 5 Data setelah Kata Sesuai KBBI**

Selain itu, setelah melakukan analisis terhadap ulasan yang telah bersih, menghapus kata abstrak tidak cukup untuk membuat ulasan bersih karena masih ditemukan beberapa kata yang tidak sesuai dan membingungkan. Oleh karena itu, penulis melakukan penghapusan kata yang terdiri dari tiga huruf, seperti oh, iya, ini, itu, dll, dan tidak memberikan informasi penting bagi model saat melakukan prediksi dengan memasukkan perintah berikut “*def remove\_small\_words(text): text = re.sub(r'\b\w{1,3}\b', "", text) return text*” dan “*data\_sentimen ['final\_text'] = data\_sentimen ['normalization'].apply (lambdaremove: remove\_small\_words (str(remove)))*”.

## 4.5. Word Tokenizing

*Word Tokenizing* adalah sebuah proses yang menjadikan kalimat dipotong menjadi sebuah kata yang kemudian akan digunakan sebagai “Token”. Perintah yang digunakan adalah sebagai berikut



```
def word_token(review_text):
    return word_tokenize(review_text)

data_sentimen['token'] = data_sentimen['final_text'].apply(lambda tokeniz:word_token(str(tokeniz)))

data_sentimen.head()
```

Gambar 4.26 Perintah *Word Tokenizing*

Hasil dari *Tokenizing* seperti tabel berikut

	<b>koment</b>	<b>clean_review</b>	<b>normalization</b>	<b>final_text</b>	<b>token</b>
0	Bagus tapi kurang update si, terakhir bulan 5 ...	bagus tapi kurang update si terakhir bulan art...	bagus tapi kurang diperbarui sih terakhir bula...	bagus tapi kurang diperbarui terakhir bulan a...	[bagus, tapi, kurang, diperbarui, terakhir, bu...
1	Sangat baik, karena terbuka buat masyarakat	sangat baik karena terbuka buat masyarakat	sangat baik karena terbuka buat masyarakat	sangat baik karena terbuka buat masyarakat	[sangat, baik, karena, terbuka, buat, masyarakat]
2	Bagus	bagus	bagus	bagus	[bagus]
3	Bagus, karena masyarakat bisa tau informasi yg...	bagus karena masyarakat bisa tau informasi yg ...	bagus karena masyarakat bisa tahu informasi ya...	bagus karena masyarakat bisa tahu informasi ya...	[bagus, karena, masyarakat, bisa, tahu, inform...
4	Bagus karena dengan adanya informasi melalui w...	[bagus, karena, dengan, adanya, informasi, mel...			

Tabel 4.1 Word Tokenizing



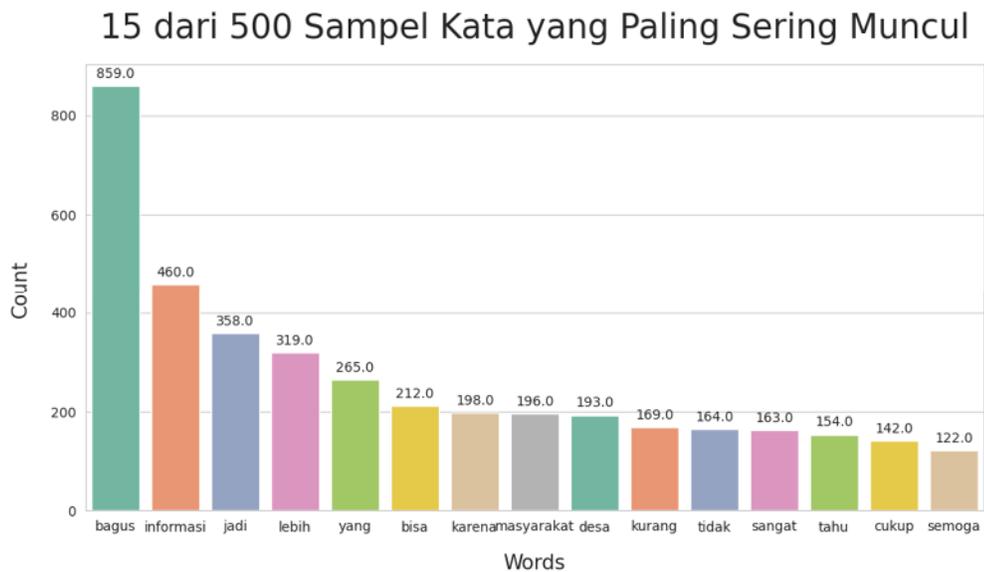
**Gambar 4.28 Menampilkan 500 kata dan jumlah kemunculannya**

Agar lebih rapih dan kata-kata diurutkan secara *descending* dan hanya diurutkan sebanyak 15 kata ternayak yang dibicarakan dengan perintah “`most_freq_df.sort_values (by='count', ascending = False) top15 = most_freq_df.iloc [:15]`” kemudian untuk melihat hasil visualisasi kita dapat menuliskan perintah berikut

```
# membuat visualisasi 15 sampel kata yang paling sering dibicarakan oleh pengguna
plt.figure(figsize=(12,6))
sns.set_style('whitegrid')

ax = sns.barplot(x='words', y='count', data=top15, palette = 'Set2')
for annotate in ax.patches:
    ax.annotate(format(annotate.get_height(), '.1f'),
                (annotate.get_x() + annotate.get_width()/2.,
                 annotate.get_height()), ha = 'center', va='center', xytext=(0,9),
                textcoords = 'offset points')
plt.title("15 dari 500 Sampel Kata yang Paling Sering Muncul", fontsize=25, pad=20)
plt.xlabel('Words', fontsize=15, labelpad=15)
plt.ylabel('Count', fontsize=15, labelpad=15)
plt.show()
```

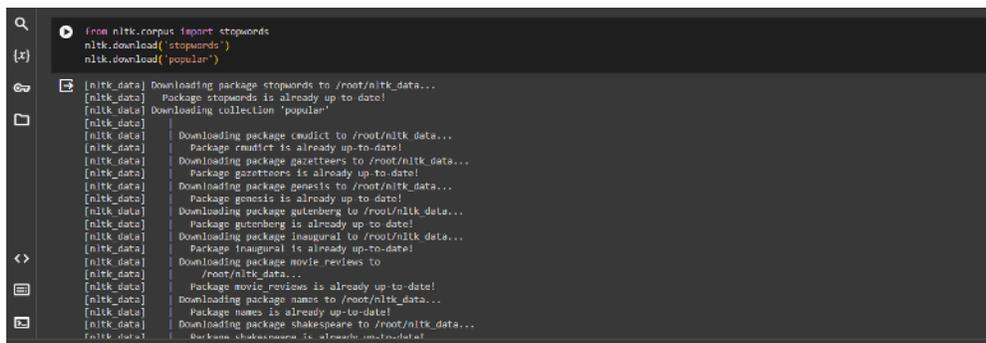
**Gambar 4.29 Perintah Visualisasi 15 Kata Paling Sering Muncul**



**Gambar 4.30 Hasil Visualisasi 15 Kata Paling Sering Muncul**

## 4.7. Remove Stopword

*Remove Stopword* adalah proses menghilangkan kata-kata penghubung kalimat, sehingga kata-kata tersebut tidak dibutuhkan. Misalkan, kata “dan”, “juga”, “walau”, dan lain sebagainya.



```

from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('popular')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading collection 'popular'
[nltk_data]
[nltk_data] | Downloading package cmudict to /root/nltk_data...
[nltk_data] | Package cmudict is already up-to-date!
[nltk_data] | Downloading package gazetteers to /root/nltk_data...
[nltk_data] | Package gazetteers is already up-to-date!
[nltk_data] | Downloading package genesis to /root/nltk_data...
[nltk_data] | Package genesis is already up-to-date!
[nltk_data] | Downloading package gutenberg to /root/nltk_data...
[nltk_data] | Package gutenberg is already up-to-date!
[nltk_data] | Downloading package inaugural to /root/nltk_data...
[nltk_data] | Package inaugural is already up-to-date!
[nltk_data] | Downloading package movie_reviews to
[nltk_data] | /root/nltk_data...
[nltk_data] | Package movie_reviews is already up-to-date!
[nltk_data] | Downloading package names to /root/nltk_data...
[nltk_data] | Package names is already up-to-date!
[nltk_data] | Downloading package shakespeare to /root/nltk_data...
[nltk_data] | Package shakespeare is already up-to-date!
  
```

Gambar 4.31 Proses Install dan Unduh Stopword



```

[] # menyiapkan kata stopwords bahasa Indonesia
Indonesian_stop = stopwords.words('Indonesian')

[] # menghitung jumlah kata stopwords bahasa Indonesia
print('length of Indonesian stopwords:', len(Indonesian_stop))

length of Indonesian stopwords: 758

[] # memasukkan stopwords bahasa Indonesia ke dalam DataFrame
stopwords_indo = pd.DataFrame(Indonesian_stop, columns=['stopwords_Indonesia'])

[] # menyimpan stopwords bahasa Indonesia ke dalam format xlsx
stopwords_indo.to_excel('stopwords_Indonesia.xlsx', index=False)
  
```

Gambar 4.32 Proses Install, memasukkan, dan menyimpan dalam dataset



```

function that used to remove stopwords

[] def remove_stopwords(review_text, indo_stopwords):
    tokenize = []
    for token in review_text:
        if token not in indo_stopwords:
            tokenize.append(token)
    return tokenize

data_sentimen['stop_review'] = data_sentimen['token'].apply(lambda stop: remove_stopwords(stop, Indonesian_stop))

data_sentimen.head()
  
```

Gambar 4.33 Fungsi Yang Digunakan untuk Menghapus Stopwords

Sebelum	Sesudah <i>Remove Stopwords</i>
[bagus, tapi, kurang, diperbarui, terakhir,]	[bagus, diperbarui, artikelnya]
[sangat, baik, karena, terbuka, buat, masyarakat]	[terbuka, masyarakat]
[bagus]	[bagus]
[bagus, karena, masyarakat, bisa, tahu, inform...]	[bagus, masyarakat, informasi]
[bagus, karena, dengan, adanya, informasi,]	[bagus, informasi, website, mengerti, berita, ..]

Tabel 4.2 Hasil Setelah dilakukan *Remove Stopword*

#### 4.8. *Stemming Normalization*

Tahap selanjutnya adalah *stemming* yaitu proses pengumpulan data yang dilakukan secara terus menerus untuk menindaklanjuti sebuah informasi, tetapi perlu dilakukan normalisasi data ke dalam Bahasa Indonesia yang baik dan benar yaitu dengan mengambil kata baku pada setiap kalimat dan menghilangkan kata imbuhan.

```

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def back_to_root(review_text):
    stop_token = ".join(review_text)
    stem = StemmerFactory()
    create_stem = stem.create_stemmer()
    result_stem = create_stem.stem(stop_token)
    return result_stem

data_sentimen['stem_review'] = data_sentimen['stop_review'].apply(lambda stem:back_to_root(stem))

data_sentimen.head()

```

Gambar 4.34 Stemming Normalization

Sebelum	Sesudah <i>Stemming Normalization</i>
[bagus, diperbarui, artikelnya]	bagus baru artikel
[terbuka, masyarakat]	buka masyarakat
[bagus]	bagus
[bagus, masyarakat, informasi]	bagus masyarakat informasi
[bagus, informasi, website, mengerti, berita, ..	bagus informasi website erti berita informasi

Tabel 4.3 Hasil Setelah dilakukan *Stemming Normalization*

## 4.9. Pelabelan

Data opini masih belum mempunyai sentimen sehingga sulit untuk mencari tahu apakah pengguna memberikan ulasan positif atau negatif. Proses pemberian sentimen tidak dilakukan secara manual dengan melihat ulasan secara satu per satu karena membutuhkan waktu yang lama dan memerlukan seorang ahli di bidang bahasa yang dapat menafsirkan ulasan kemudian mengelompokkan ke sentimen positif dan negatif. Maka dari itu, penulis melakukan proses pemberian label sentimen dengan menerapkan metode yang berbasis *lexicon* atau biasa dikenal

dengan *lexicon-based method*. Kamus yang digunakan adalah *InSet Lexicon* yang terdiri dari kamus positif dan negatif.



```

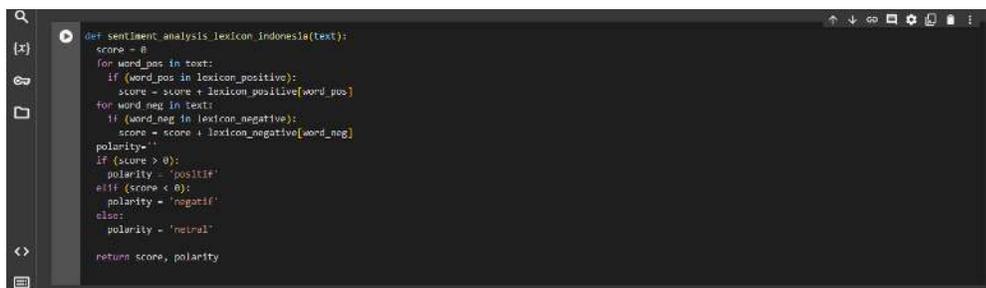
Read the positive and negative lexicon
lexicon_positive = dict()
import csv
with open('lexicon_positive.ver1.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        lexicon_positive[row[0]] = int(row[1])

lexicon_negative = dict()
import csv
with open('lexicon_negative.ver1.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        lexicon_negative[row[0]] = int(row[1])

```

**Gambar 4.35** Perintah *lexicon-based method*

Ulasan akan diberi score terlebih dahulu dengan menyesuaikan dengan bobot kata yang terdapat di kamus. +5 untuk kata positif dan -5 untuk negatif. Setelah itu, bobot ulasan dijumlahkan kemudian dikelompokkan menjadi sentimen positif, negatif, dan netral. netral ini diartikan sebagai hasil kalkulasi ulasan bernilai 0.



```

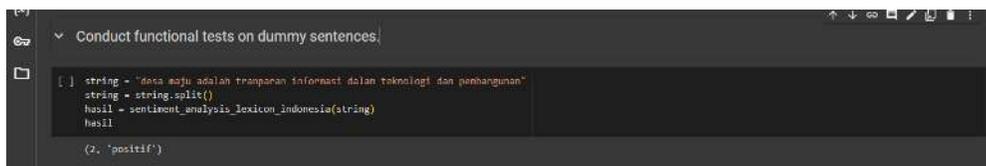
def sentiment_analysis_lexicon_indonesia(text):
    score = 0
    for word_pos in text:
        if (word_pos in lexicon_positive):
            score = score + lexicon_positive[word_pos]
    for word_neg in text:
        if (word_neg in lexicon_negative):
            score = score + lexicon_negative[word_neg]
    polarity=""
    if (score > 0):
        polarity = 'positif'
    elif (score < 0):
        polarity = 'negatif'
    else:
        polarity = 'netral'
    return score, polarity

```

**Gambar 4.35** Pembobotan sentimen

Dengan perintah diatas, alimat opini telah dilakukan pembobotan sesuai dengan kata-kata yang terdapat didalamnya. Oleh karena itu, waktu yang dibutuhkan untuk mengetahui sentimen dalam data yang banyak akan lebih cepat dikerjakan oleh *tools* diatas. Untuk lebih meyakinkan, penulis mencoba

menambahkan kalimat ujicoba dengan perintah sebagai berikut `"string = "desa maju adalah tranparan informasi dalam teknologi dan pembangunan" string = string.split () hasil = sentiment_analysis_lexicon_indonesia (string) hasil"` yaitu penulis ingin mengetahui apakah kalimat “desa maju adalah tranparan informasi dalam teknologi dan pembangunan” dibaca sebagai kalimat positif atau negatif. Dan hasil menunjukkan kalimat tersebut bernilai 2, yang artinya ‘Positif’. Seperti gambar berikut



```

Conduct functional tests on dummy sentences.

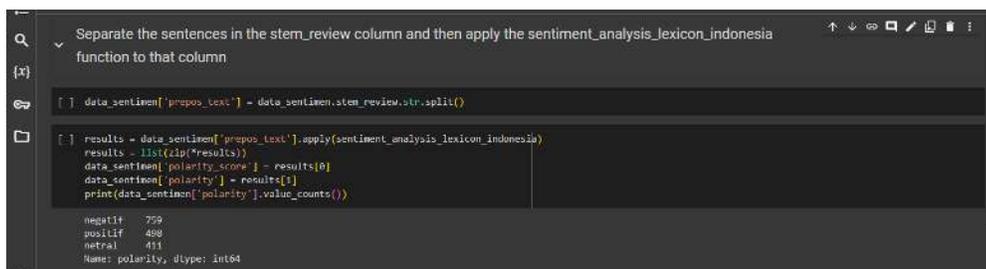
[ ] string = "desa maju adalah tranparan informasi dalam teknologi dan pembangunan"
string = string.split()
hasil = sentiment_analysis_lexicon_indonesia(string)
hasil

(2, 'positif')

```

**Gambar 4.36 Conduct Functional Tests**

Pisahkan kalimat pada kolom `stem_review` lalu terapkan fungsi `sentimen_analisis_lexicon_indonesia` pada kolom yang baru yang bernama `polarity` sehingga diperoleh hasil negatif 759, positif 498, dan netral 411.



```

Separate the sentences in the stem_review column and then apply the sentiment_analysis_lexicon_indonesia
function to that column

[ ] data_sentimen["prepos_text"] = data_sentimen.stem_review.str.split()

[ ] results = data_sentimen["prepos_text"].apply(sentiment_analysis_lexicon_indonesia)
results = list(zip(*results))
data_sentimen["polarity_score"] = results[0]
data_sentimen["polarity"] = results[1]
print(data_sentimen["polarity"].value_counts())

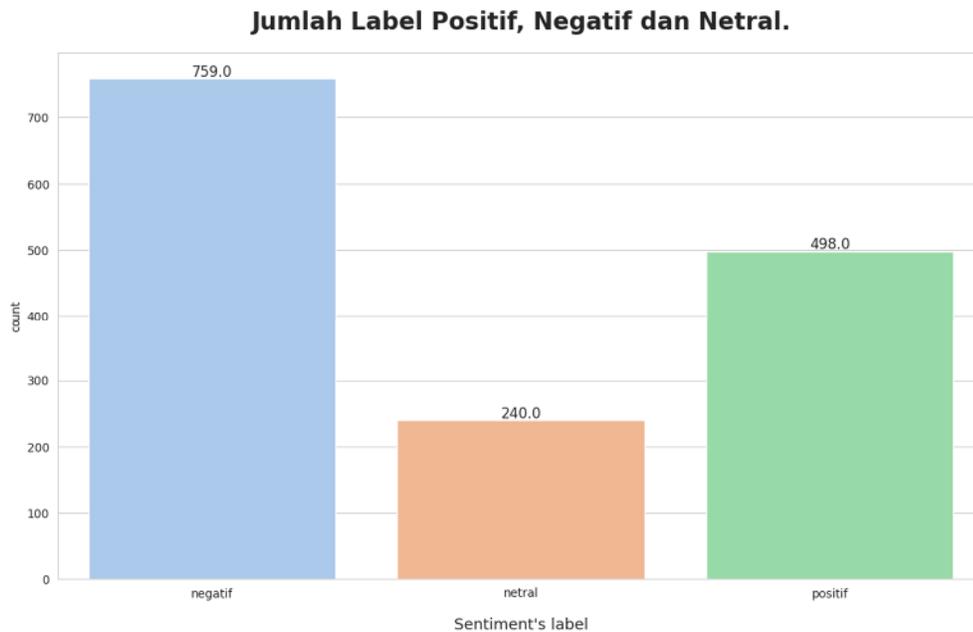
negatif 759
positif 498
netral 411
Name: polarity, dtype: int64

```

**Gambar 4.36 Memisahkan Kalimat ke dalam kolom polarity**

Pada tahap selanjutnya, penulis mengubah kembali spasi `np.nan` kemudian menghapusnya dengan menggunakan fungsi `dropna` dengan perintah `"data_sentimen = data_sentimen.replace ("np.nan, regex=True) data_sentimen =`

`data_sentimen.dropna()`” dan reset index tabel agar urutan menjadi penomoran menjadi lebih rapih “`data_sentimen = data_sentimen.reset_index(drop=True)`” untuk melihat hasilnya masukkan perintah “`print(data_sentimen ['polarity'].value_counts())`” sehingga hasilnya menjadi negatif 759, positif 498 dan, netral 240.



**Gambar 4.36 Gambar Diagram Label Negatif, Netral dan Positif**

Setelah dilakukan *polarity\_score* maka data tersebut diklasifikasikan kedalam komentar positif, netral, atau negatif. Kemudian akan dijadikan data numerik dimana untuk komentar positif menjadi bernilai '1', netral '0' dan negatif '-1' kemudian data hasilnya disimpan kembali kedalam *google drive* dengan perintah “`data_sentimen.to_excel("danu-ok.xlsx", index=False)`” dengan nama file 'hasil opini.xlsx' perintah mengonversikan sentimen menjadi 1 untuk positif, -1 untuk negatif dan netral 0 adalah sebagai berikut

```

[66] # mengonversikan sentimen menjadi 1 untuk positif dan -1 untuk negatif
polarity = []
for convert in data_sentimen['polarity']:
    if convert == "positif":
        num_polarity = 1
    elif convert == "negatif":
        num_polarity = -1
    else: # Netral atau kategori lainnya
        num_polarity = 0

    polarity.append(num_polarity)

data_sentimen['polarity_numeric'] = polarity

```

Gambar 4.37 mengonversikan sentimen

Setelah dikonversi maka diperoleh data negatif 759, positif 498, dan netral 240. Persentase Sentimen Positif: 33.27%, Persentase Sentimen Negatif: 50.7%, Persentase Sentimen Netral: 16.03%, Perbedaan antara Sentimen Positif dan Sentimen Negatif: -17.43%, Perbedaan antara Sentimen Positif dan Sentimen Netral: 17.23%, Perbedaan antara Sentimen Negatif dan Sentimen Netral: 34.67%.

#### 4.10. Sentimen Wordcloud

Setelah proses pelabelan selesai, hasil pelabelan berupa data sentimen positif, netral atau negatif dapat ditampilkan dalam bentuk *wordcloud* yang berisi kata-kata sesuai label yang telah dipisahkan.



Gambar 4.38 Wordcloud Sentimen keseluruhan



## 4.11. Klasifikasi Data dengan KNN

Sebelum data pelabelan dapat diolah menjadi data yang sesuai dengan klasifikasinya, KNN perlu melakukan ujicoba berupa data latih dan kemudian KNN akan melakukan tahapan uji setelah mendapatkan data dari ujicoba atau secara sederhana penulis membagi dua jenis data yaitu data latih dan data uji dengan perintah sebagai berikut “X = data\_sentimen['stem\_review'] y = data\_sentimen['polarity]” dengan keterangan data pada kolom *stem\_review* adalah X dan kolom *polarity* adalah kolom Y atau dapat dimisalkan X adalah fitur, sementara Y adalah label.

Penulis menggunakan 80% data menjadi data latih dan 20% sisanya adalah data uji didapat hasil jumlah data latih: 1.197, jumlah data uji: 300 seperti pada gambar berikut



```

{x} In [ ]: from sklearn.model_selection import train_test_split
          #split data
          # Misalnya, jika 'X' adalah fitur dan 'y' adalah label
          X = data_sentimen['stem_review']
          y = data_sentimen['polarity']

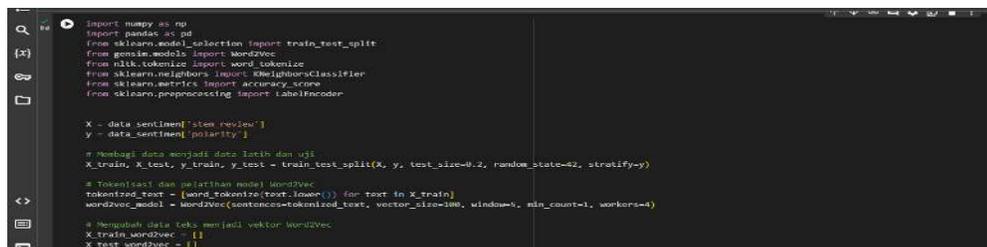
          # Pisahkan data menjadi data latih (80%) dan data uji (20%)
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

          # Tampilkan ukuran data latih dan data uji
          print("Jumlah data latih:", len(X_train))
          print("Jumlah data uji:", len(X_test))

Out [ ]: Jumlah data latih: 1197
        Jumlah data uji: 300
  
```

Gambar 4.41 Wordcloud Sentimen Netral

Kemudian dilanjutkan dengan perintah berikut



```

{x} In [ ]: import numpy as np
          import pandas as pd
          from sklearn.model_selection import train_test_split
          from gensim.models import Word2Vec
          from nltk.tokenize import word_tokenize
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.metrics import accuracy_score
          from sklearn.preprocessing import LabelEncoder

          X = data_sentimen['stem_review']
          y = data_sentimen['polarity']

          # Membagi data menjadi data latih dan uji
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

          # Tokenisasi dan pelatihan model Word2Vec
          tokenized_text = [word_tokenize(text.lower()) for text in X_train]
          word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=1, min_count=1, workers=4)

          # Mengubah data teks menjadi vektor Word2Vec
          X_train_word2vec = []
          X_test_word2vec = []
  
```

```

# Mengumpulkan vektor word2vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengumpulkan vektor word2vec untuk setiap dokumen pada data uji
for text in X_test:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_test_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_test_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor word2vec men jadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)

# Untuk memvisualisasikan hasil prediksi
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.fit_transform(y_test)

# Menghitung akurasi dengan menggunakan K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_word2vec, y_train_encoded)
y_pred = knn.predict(X_test_word2vec)

# Menghitung akurasi
accuracy = sum(y_test_encoded == y_pred) / len(y_test_encoded)
print("Akurasi: %.4f" % accuracy)

```

Gambar 4.41 Melihat Akurasi KNN

Dengan data latih diatas, KNN berhasil memperoleh akurasi sebanyak 0.7433333333333333.

## 4.12. Uji Performa dengan *Confusion Matrix*

Selanjutnya adalah pengujian performa algoritma yang digunakan dengan *Confusion Matrix* pada dasarnya *Confusion Matrix* memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. *Confusion Matrix* berbentuk tabel matriks yang menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui.

```

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Confusion matrix untuk K-Nearest Neighbors
cm_knn = confusion_matrix(y_test_encoded, knn_predictions)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_knn, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix - K-Nearest Neighbors")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

```

Gambar 4.42 Perintah *Confusion Matrix*



**Gambar 4.43 Hasil *Confusion Matrix***

Untuk melihat hasil dalam bentuk angka perintah yang dituliskan adalah sebagai berikut

```

import pandas as pd
from sklearn.metrics import confusion_matrix

# Confusion matrix untuk K-Nearest Neighbors
cm_knn = confusion_matrix(y_test_encoded, knn_predictions)

# Membuat DataFrame dari confusion matrix
label_names = label_encoder.classes_
cm_df = pd.DataFrame(cm_knn, index=label_names, columns=label_names)

# Menampilkan DataFrame
print("Confusion Matrix - K-Nearest Neighbors:")
print(cm_df)

```

**Gambar 4.44 *Dataframe Confusion Matrix***

<b>Predicat</b> <b>Actual</b>	<b>Negatif</b>	<b>Netral</b>	<b>Positif</b>
<b>Negatif</b>	127	6	19
<b>Netral</b>	14	22	12
<b>Positif</b>	21	5	74

**Tabel 4.5 Confusion Matrix**

Berdasarkan table diatas, penulis dapat melakukan beberapa tahapan pengujian diantaranya

1. Akurasi

Rumus untuk melihat akurasi adalah

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sehingga dapat dimasukkan data sebagai berikut ini

$$Accuracy = \frac{127 + 74}{127 + 74 + 21 + 19}$$

Dan hasil nya diperoleh akurasi sebesar **0,8340248963**

## 2. Presisi

*Precision* adalah perbandingan antara data yang terdeteksi benar dengan seluruh data prediksi pada suatu kelas atau tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem

$$Precision = \frac{TP}{TP + FP}$$

Sehingga dapat dimasukkan data sebagai berikut ini

$$Precision = \frac{127}{127 + 21}$$

Dan hasilnya diperoleh presisi sebesar **0,858108108**

## 3. Recall

*Recall* adalah perbandingan hasil klasifikasi dengan kelas sesungguhnya atau tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi.

$$Recall = \frac{TP}{TP + FN}$$

Sehingga setelah dimasukkan menjadi

$$Recall = \frac{127}{127 + 19}$$

Dan hasilnya *Recall* adalah **0,8698630137**

## 4. F1-Score

*F1-Score* menggambarkan perbandingan rata-rata precision dan recall yang dibobotkan.

$$F1 - Score = \frac{2 * recall * precision}{recall + precision}$$

Sehingga dimasukkan nilainya menjadi

$$F1 - Score = \frac{2 * 0,8698630137 * 0,858108108}{0,8698630137 + 0,858108108}$$

Dan hasilnya adalah **0,863945578**

Berikut adalah table hasil uji performa dengan *Confusion Matrix*

<i>Confusion Matrix</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
Hasil	<b>0,8340248963</b>	<b>0,858108108</b>	<b>0,8698630137</b>	<b>0,863945578</b>

Tabel 4.4 Hasil Keseluruhan *Confusion Matrix*

Berdasarkan uji performa menggunakan *Confusion Matrix*, diperoleh hasil yang baik dengan hasil rata-rata diatas 8,5.

#### 4.13. Ekstrak Fitur dengan TF-IDF

Setelah uji performa dilakukan dengan *Confusion Matrix* dilakukan dan diperoleh hasil yang baik, selanjutnya penulis melakukan ekstrak fitur menggunakan TF-IDF dengan memberikan perintah sebagai berikut

```

#menghitung akurasi KNN jika vektorisasi dengan TF-IDF
# Memilih fitur dan label
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Membuat TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000) # Sesuaikan dengan kebutuhan Anda

# Melakukan vektorisasi pada data latih dan uji
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# K-Nearest Neighbors (KNN) dengan TF-IDF
knn_tfidf_model = KNeighborsClassifier(n_neighbors=5)
knn_tfidf_model.fit(X_train_tfidf.toarray(), y_train_encoded)
knn_tfidf_predictions = knn_tfidf_model.predict(X_test_tfidf.toarray())
knn_tfidf_accuracy = accuracy_score(y_test_encoded, knn_tfidf_predictions)

# Menampilkan hasil akurasi
print("KNN Accuracy (TF-IDF):", knn_tfidf_accuracy)

```

KNN Accuracy (TF-IDF): 0.7466666666666667

Gambar 4.45 Perintah Uji Performa TF-IDF

Dan hasil uji performa menggunakan TF-IDF adalah 0.7466666666666667, sehingga dapat dikatakan mendapatkan hasil yang cukup baik.

#### 4.14. Uji Performa dengan *Cross Validation*

Uji Performa terakhir menggunakan *Cross Validation*, menurut penelitian sebelumnya[16], hasil *Cross Validation* dalam melakukan uji performa untuk algoritma KNN mendapatkan hasil yang baik, untuk itu penulis melakukan uji performa dengan *Cross Validation* sebagai pembanding. Perintah yang dimasukkan adalah sebagai berikut