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

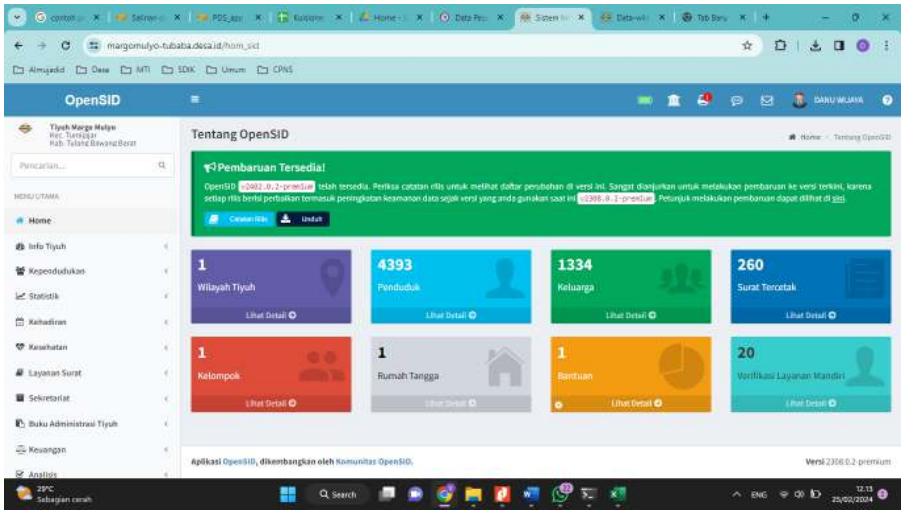
      Status Desa Menurut IDM : Mandiri

      Sebagai salah satu Desa yang telah mendapat predikat mandiri berdasarkan Indeks Desa Membangun (IDM) yang dikeluarkan oleh Kementerian Desa, Pembangunan Daerah Tertinggal, dan Tranmigrasi, 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 diatasnya 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

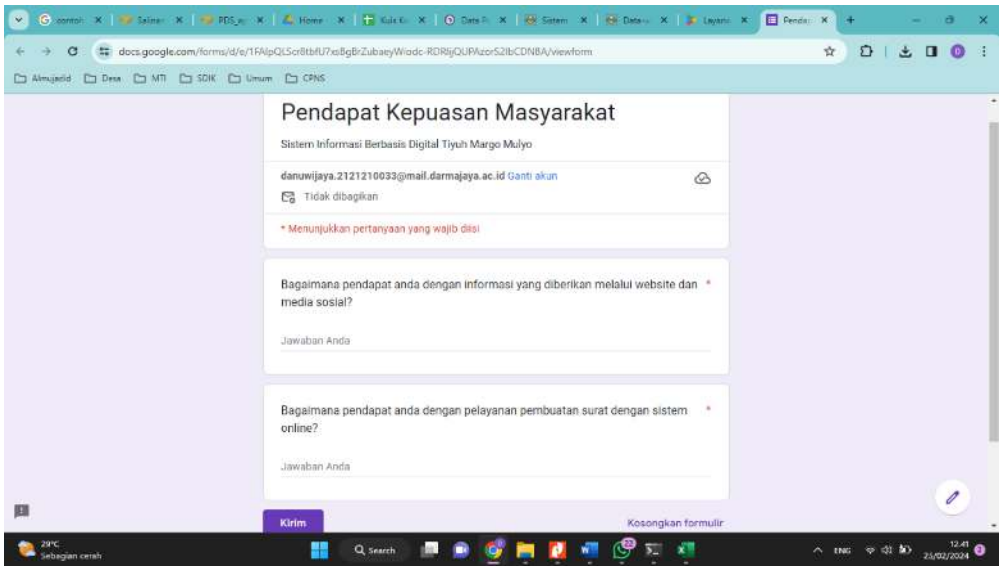


Gambar 4.1 Sistem Informasi Desa pada *OpenSID*

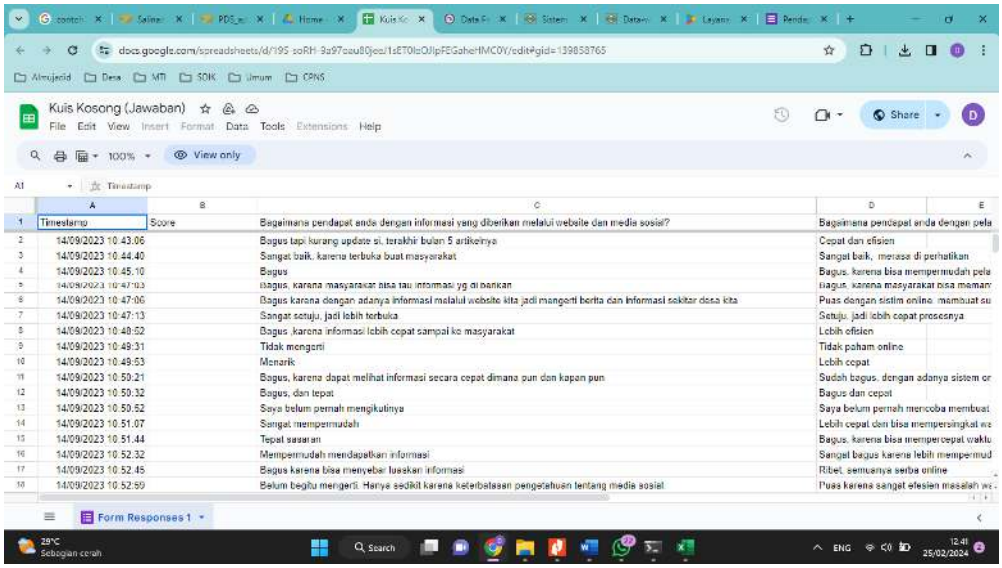
Beberapa fitur dalam *OpenSID* yang memudahkan perangkat desa yaitu dalam pelaporan keuangan, pelaporan jumlah penduduk, pelaporan kegiatan pembangunan, layanan surat, dan lain sebagainya. Dan aplikasi *mobile* juga telah tersedia di *Play Store* yang dapat diunduh di gawai dan telah diunduh sebanyak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

c. Opini Masyarakat

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



Gambar 4.2 Tampilan Google Form



Gambar 4.3 Tampilan Hasil *Google Form*

4.2. Analisis Data

1. Kecerdasan Buatan

   Kecerdasan buatan atau *Artificial Intelegence (AI)* adalah teknologi yang memungkinkan mesin atau komputer untuk melakukan tugas yang biasanya membutuhkan kecerdasan menusia, 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.

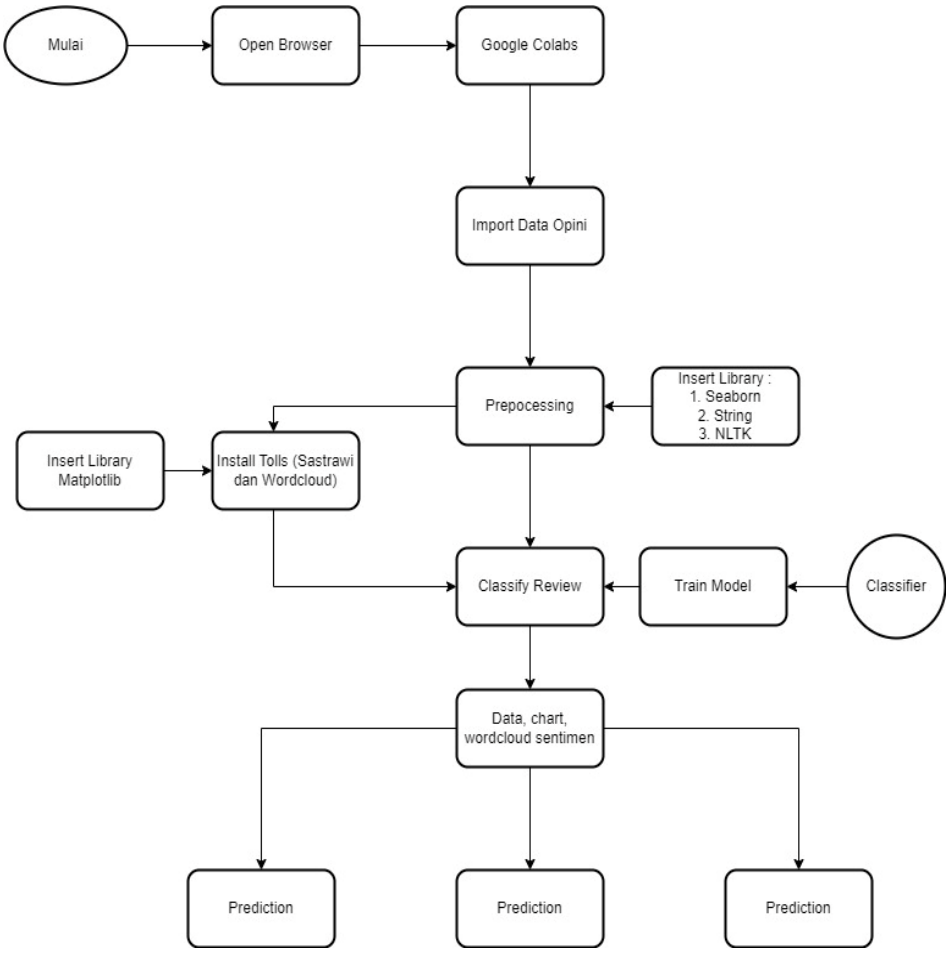
2. *Machine Learning* dan *Natural Language Processing (NLP)*

   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 permu 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 sematik 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.

Flow Chart Penggunaan *Machine Learning*



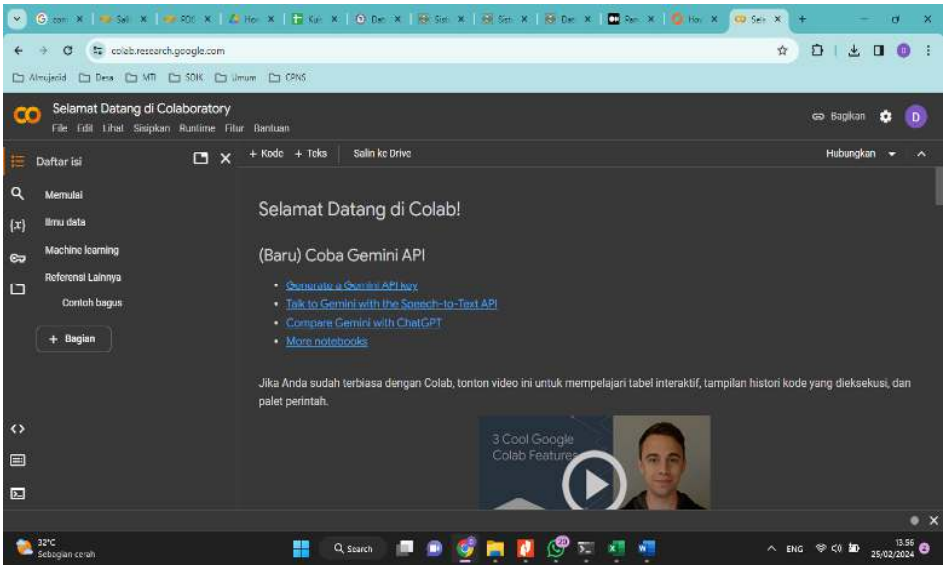
Sumber : Olah data Penulis

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 unti 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*

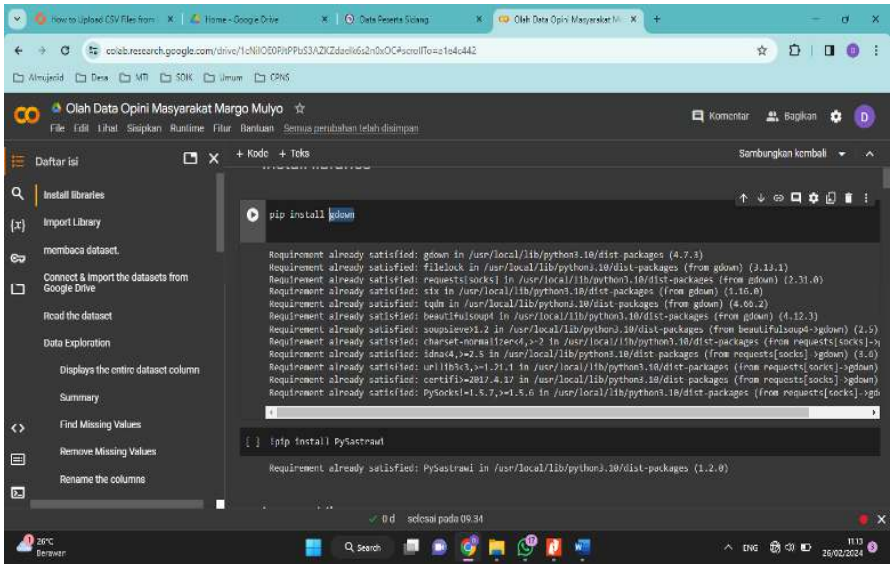


Gambar 4.5 Laman google colabs

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*

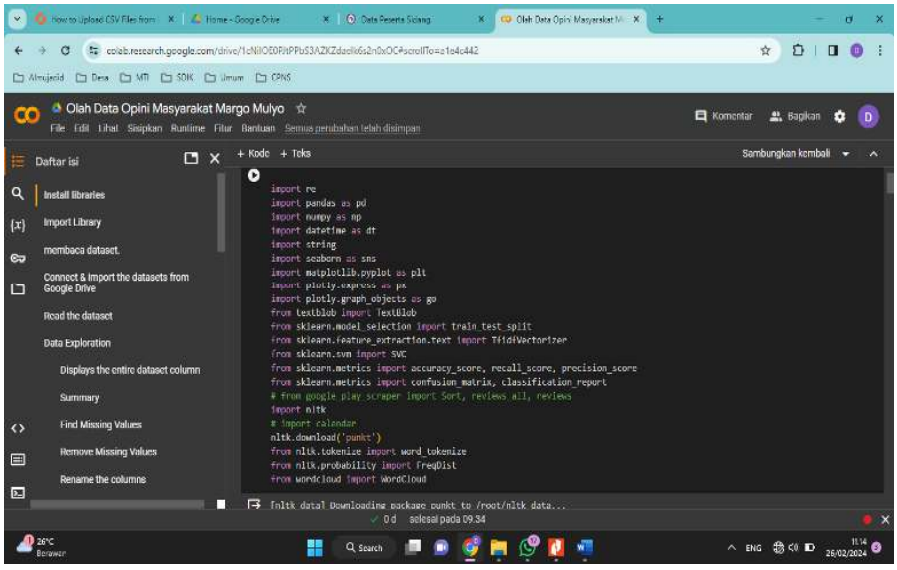
Ada beberapa *library* yang perlu di*install* untuk analisis yang akan penulis lakukan, diantaranya *gdown* dan Sastrawi. *Gdown* adalah kependekan dari *Google Drive Public File Downloader* adalah *library* yang memungkinkan *colabs* untuk mengambil data/dokumen dari *google drive* dan *library* sastrawi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



Gambar 4.6 *Install Library*

b. Import Library

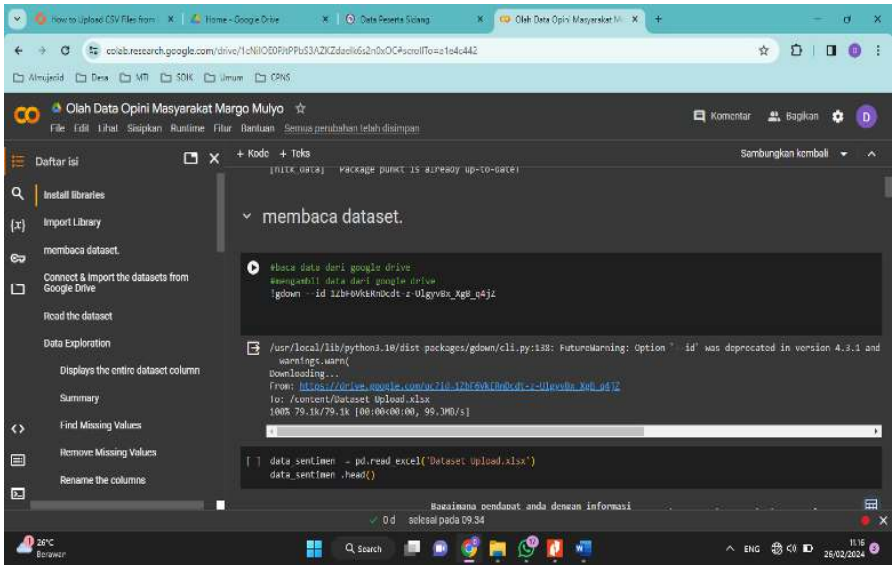
Setelah berhasil ter*install*, Langkah selanjutnya adalah *import data library* seperti pada gambar berikut



Gambar 4.7 *Import Library*

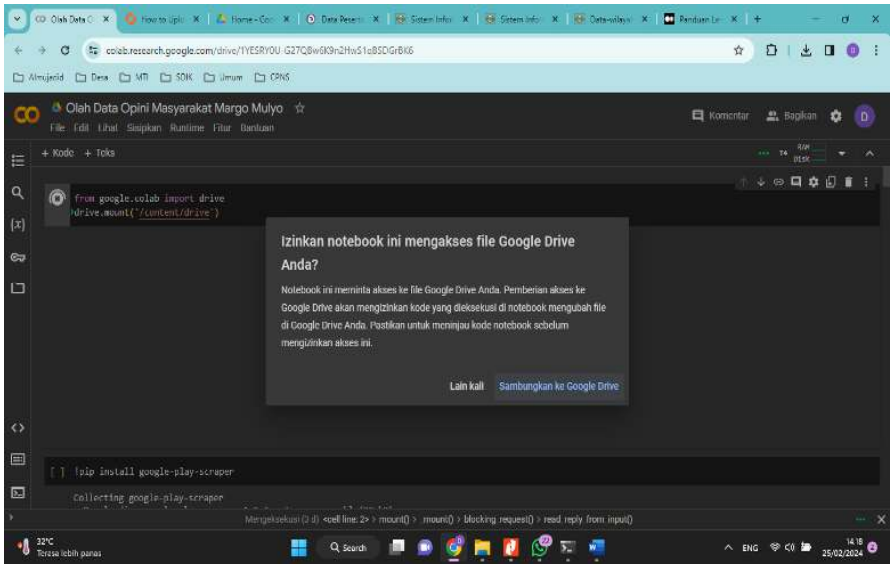
c. Import Data

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



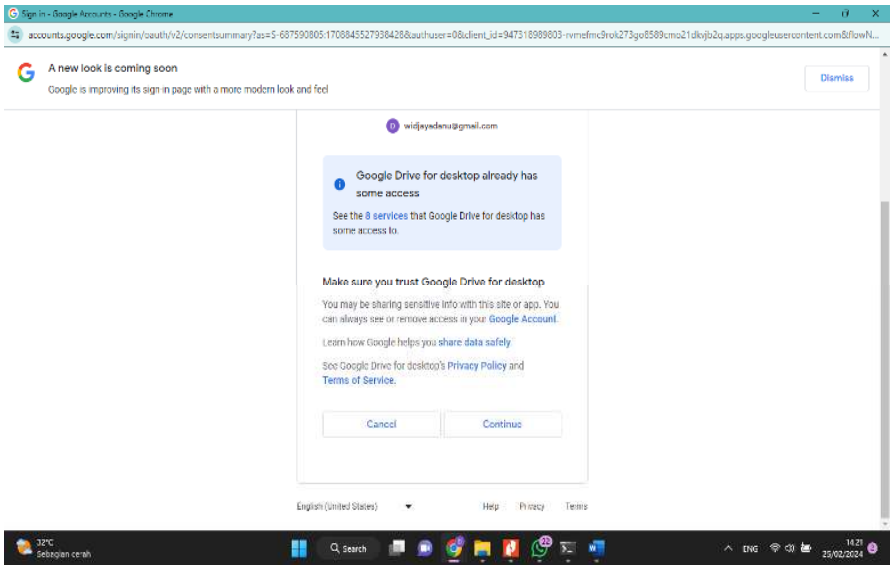
Gambar 4.8 Printah Import data

Kemudian hubungkan ke akun google drive seperti gambar berikut



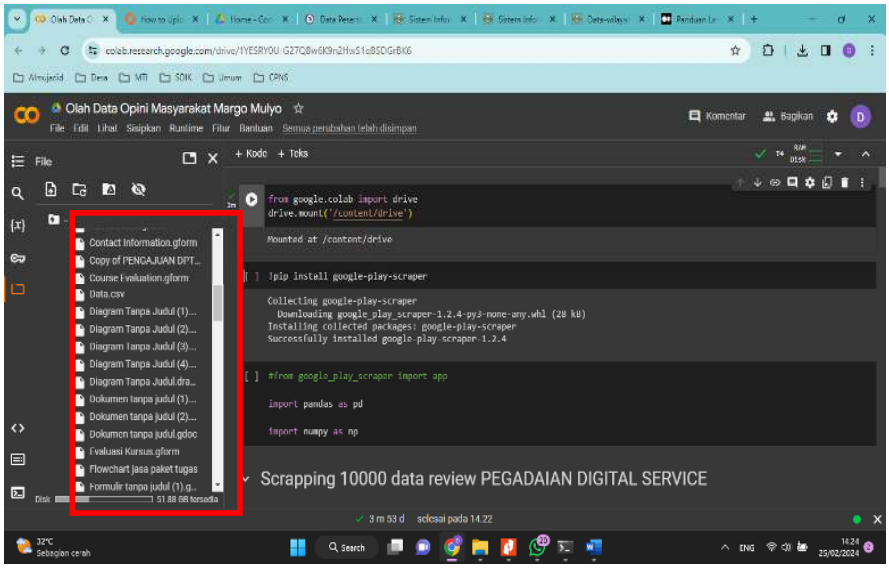
Gambar 4.9 *Colabs* Meminta Ijin Akses Ke *Google Drive*

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



Gambar 4.10 Login Akun Google

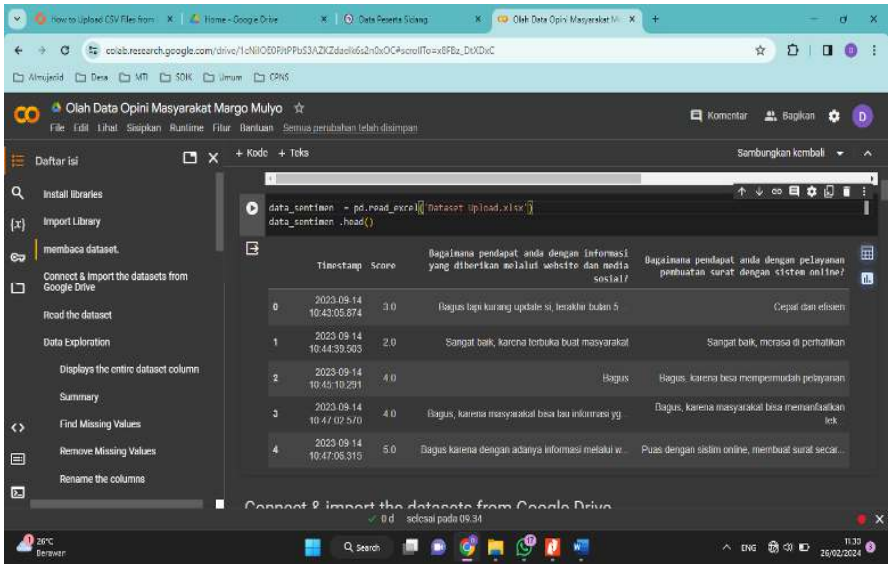
Jika berhasil maka akan tampil drive kita seperti gambar dibawah ini



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

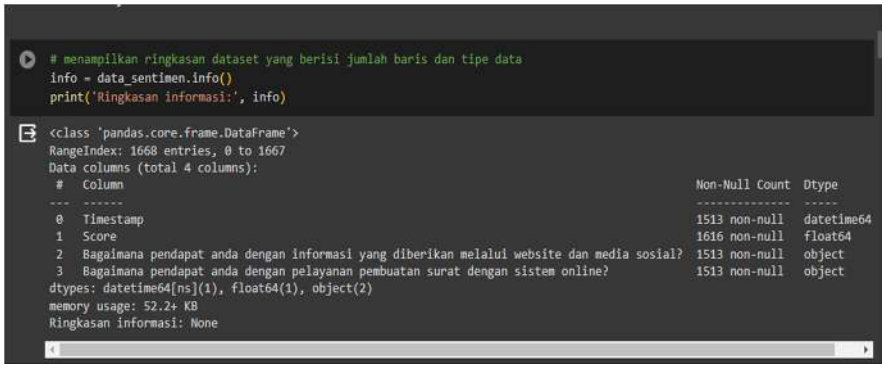
Upload.xlsx' dan perintah "*data_sentimen .head()*" untuk menampilkan lima data teratas dalam file 'Dataset Upload.xlsx' seperti gambar berikut



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



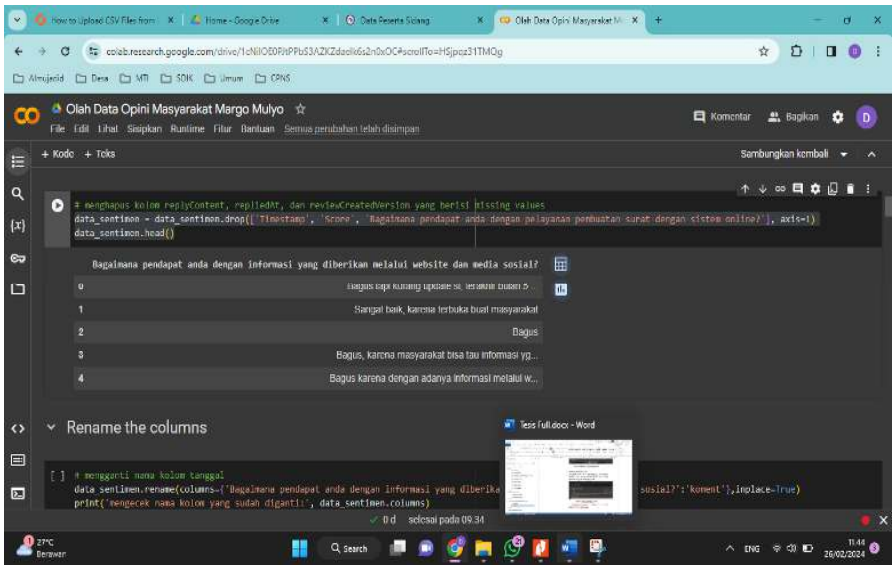
```
# menampilkan ringkasan dataset yang berisi jumlah baris dan tipe data
info = data_sentimen.info()
print('Ringkasan informasi:', info)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1668 entries, 0 to 1667
Data columns (total 4 columns):
 #   Column                                                                Non-Null Count  Dtype
---  ------                                                                --------------  -----
 0   Timestamp                                                             1513 non-null   datetime64
 1   Score                                                                 1616 non-null   float64
 2   Bagaimana pendapat anda dengan informasi yang diberikan melalui website dan media sosial?  1513 non-null   object
 3   Bagaimana pendapat anda dengan pelayanan pembuatan surat dengan sistem online?  1513 non-null   object
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 52.2+ KB
Ringkasan informasi: None
```

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

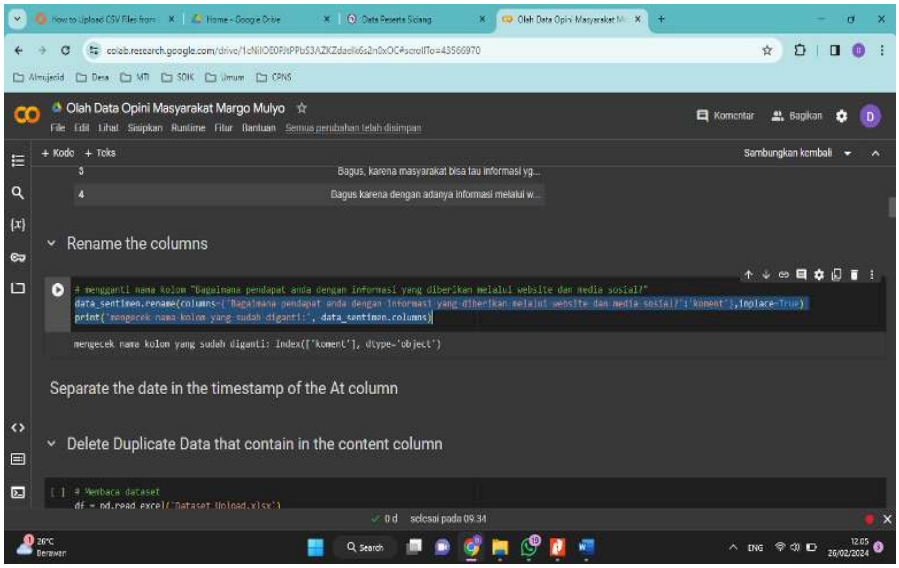


Gambar 4.13 Menampilkan Hasil Remove Missing Dataset

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*

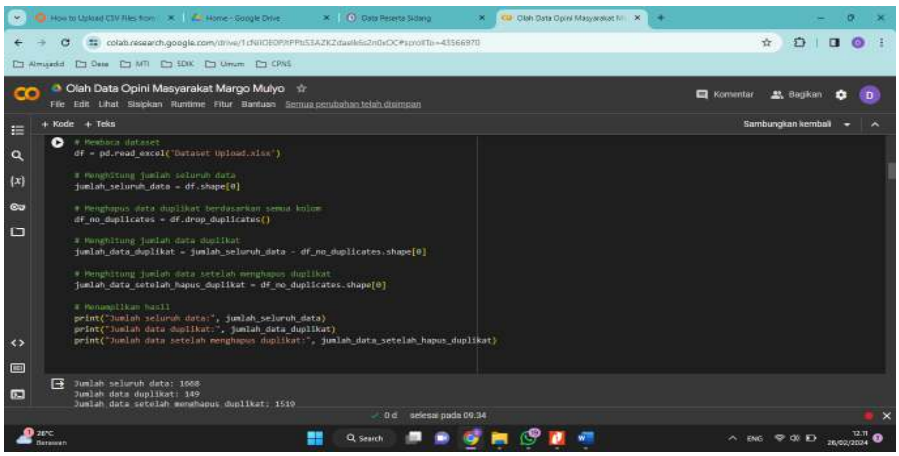
sosial?':'koment'},inplace=True) print('mengecek nama kolom yang sudah diganti:', data_sentimen.columns)" seperti gambar berikut



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



Gambar 4.15 Menghapus Data Ganda

Berdasarkan perintah diatas maka diperolah hasil sebagai berikut

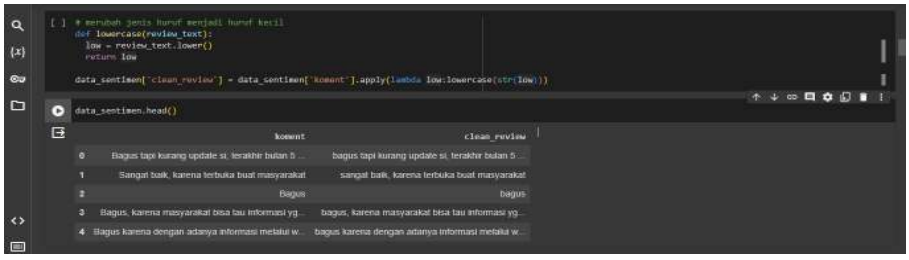
Jumlah seluruh data	: 1668
Jumlah data duplikat	: 149
Jumlah data setelah menghapus duplikat	: 1519

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*

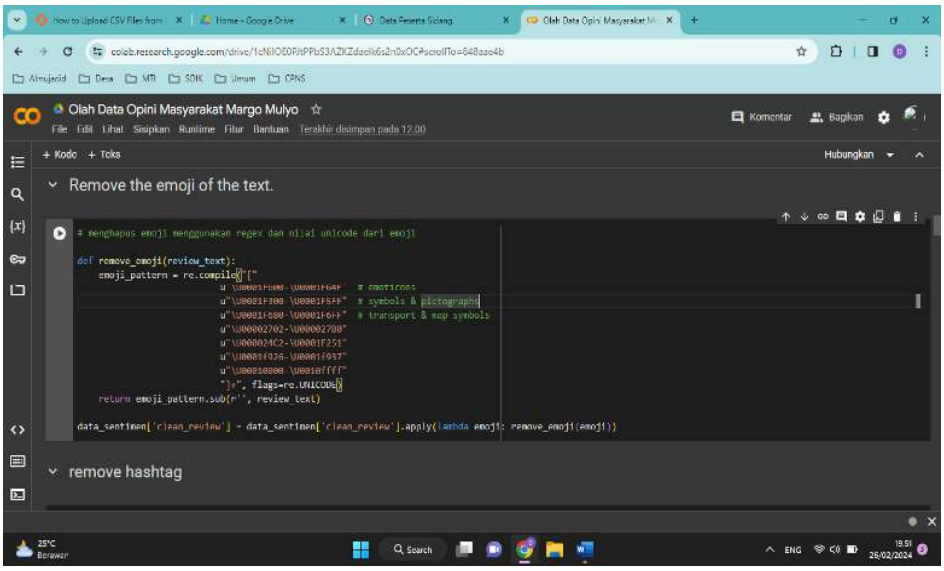
   Case Folding adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf a sampai z yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter.* Untuk melakukannya perlu dimasukkan perintah sebagai berikut *"def lowercase(review_text): low = review_text.lower() return low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



Gambar 4.16 Proses *Case Folding*

2. *Cleaning Data*

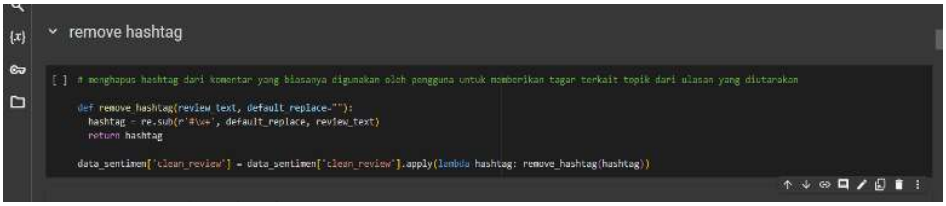
Proses selanjutnya adalah membersihkan data yaitu dengan menghapus emoji menggunakan regex dan nilai unicode dari emoji dengan perintah seperti pada gambar berikut



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

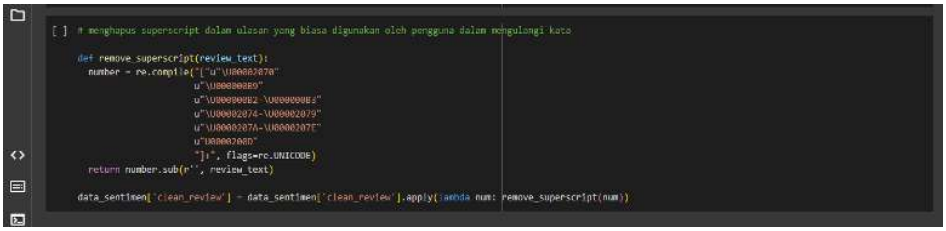


Gambar 4.18 Proses *Cleaning Data (Hashtag)*

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

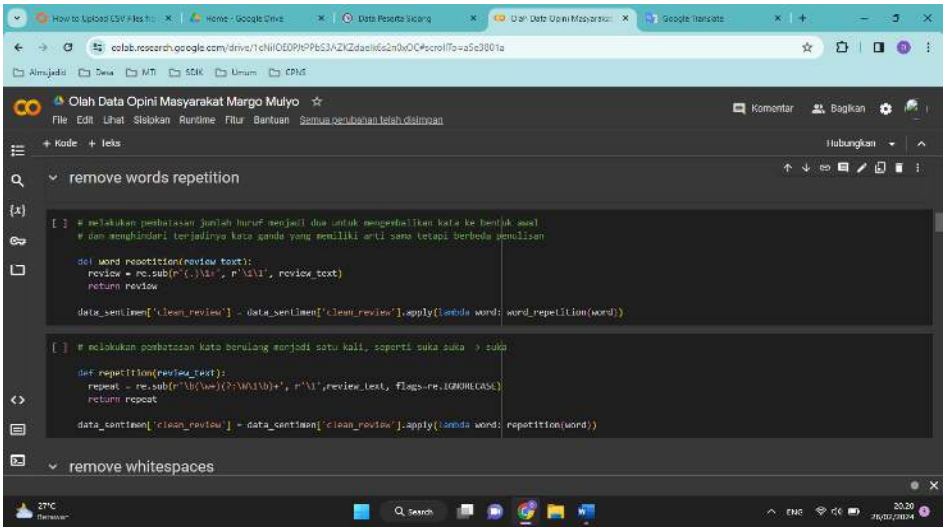
Gambar 4.19 Proses *Cleaning Data (Angka dan Tanda Baca)*



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



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_whitespaces(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_whitespaces(extra_spaces))*"

7. Membersihkan Kata-kata yang tidak Perlu

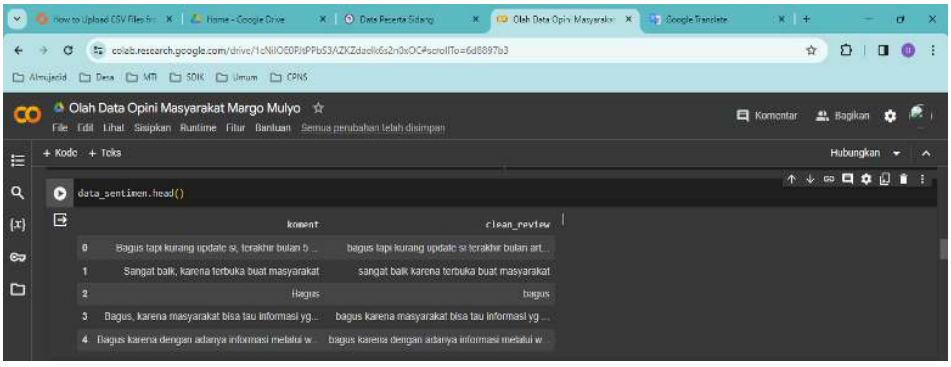
   Komentar tidak hanya berisi kata yang bisa ditafsirkan atau dimengerti oleh manusia. Ada kalanya pengguna memberikan informasi berupa ketawa, atau kata abstrak yang tidak bisa/sulit diterjemahkan. Dimana kata tersebut bisa membingungkan orang yang menganalisisnya karena tidak tahu arti dari kata tersebut. 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 'uqjy', 'dehdje', 'jwdnmb', 'wxvlezuvws', 'ircel', 'swjjw', 'dmv', 'dxrgz','qjtajuq','zzhwntdft','xywrfw' ,'fztnmdrrduy' ,'xsezuz', 'kbyv', 'iyyveyzqkdgm', 'arlrmdjy', 'sfarif', 'sfomar', 'nzxx', 'mtsco','wkwk', 'wkwkw', 'wkwkwk', 'hihi', 'hihihii', 'hihihi', 'hehehe', 'hehehehe', 'hehe', huhu', 'huhuu', 'ancok', 'guak', 'cokcok', 'hhmm', 'annya', 'hufft', 'href'.

Gambar 4.22 Proses Hapus Data tidak digunakan

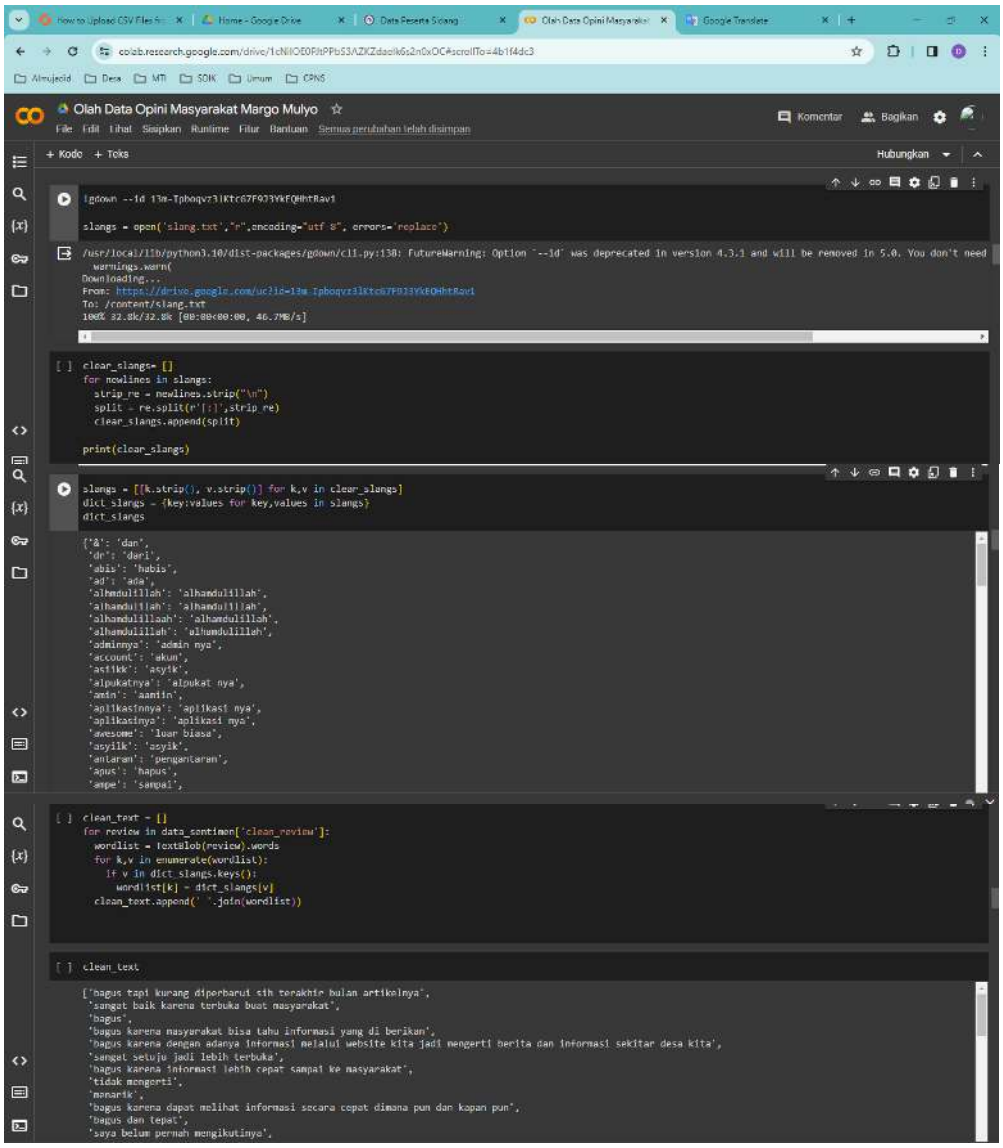
Setelah melakukan pembersihan data, untuk melihat hasilnya masukkan perintah berikut "*data_sentimen.head()*" maka akan tampil lima data teratas yang telah dilakukan pembersihan



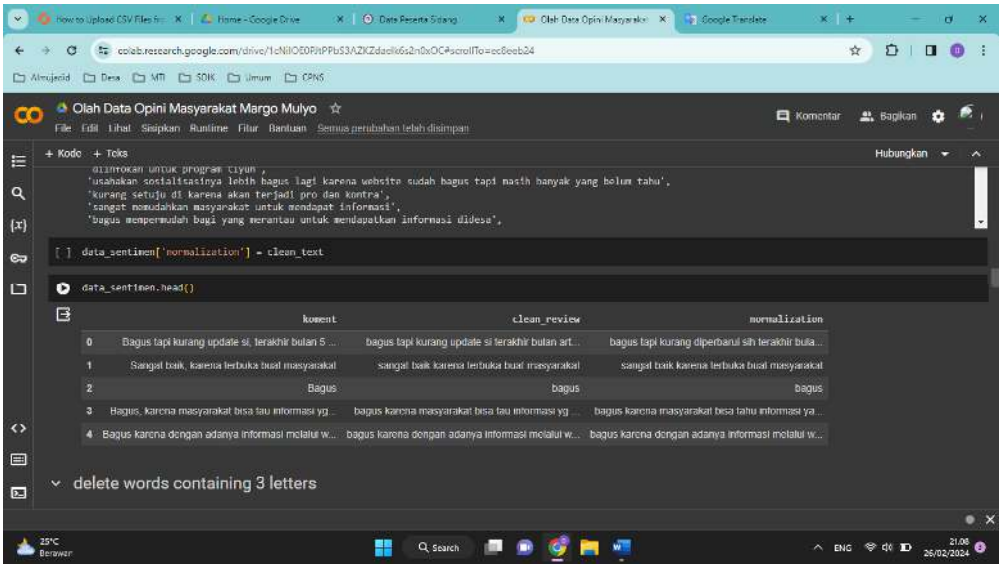
Gambar 4.23 Menampilkan Data Hasil Pembersihan

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



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

Gambar 4.26 Perintah *Word Tokenizing*

Hasil dari *Tokenizing* seperti tabel berikut

	koment	**clean_review**	**normalization**	**final_text**	**token**
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 melalui w...	bagus karena dengan adanya informasi melalui w...	bagus karena dengan adanya informasi melalui w...	[bagus, karena, dengan, adanya, informasi, mel...

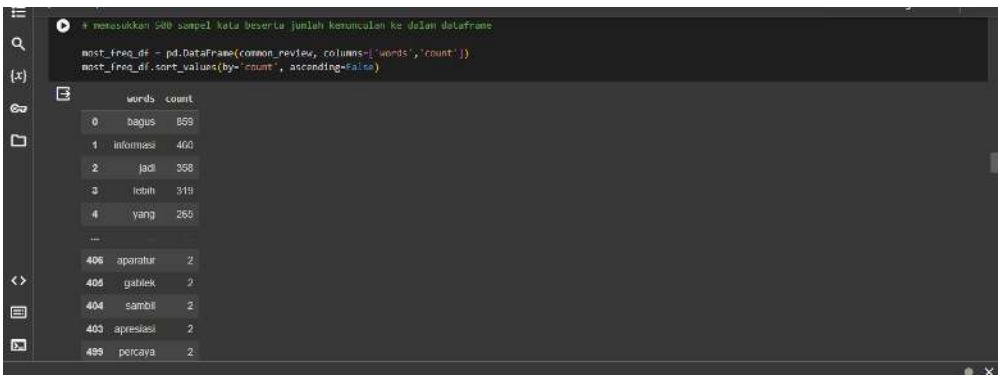
Tabel 4.1 Word Tokenizing

4.6. *Wordcloud Visualization*

Setelah proses tokenisasi dilakukan, kata-kata yang dikumpulkan menjadi token kemudian ditampilkan dalam bentuk gambar sehingga menjadikan penyajian data menjadi lebih menarik, namun kata-kata yang ditampikan dibatasi hanya sebanyak 500 kata, dengan perintah sebagai berikut *"review_all = [] for review in data_sentimen['final_text']: text = review.split() for token_review in text: print(token_review) review_all.append(token_review)"*

Gambar 4.27 Perintah *Wordcloud Visualization*

Penulis juga menampilkan data frame dengan memasukkan 500 kata beserta jumlah kemunculannya dengan perintah berikut

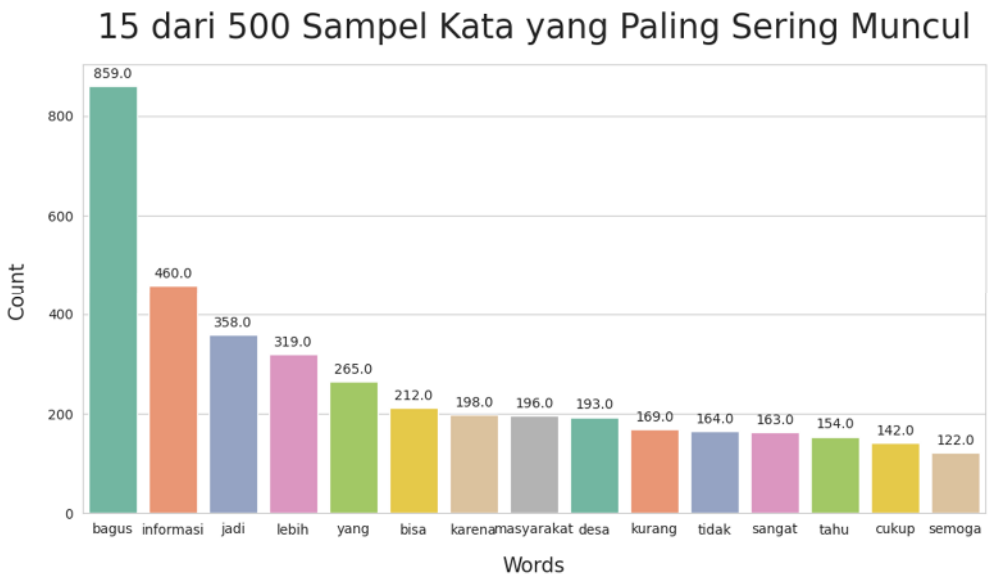


Gambar 4.28 Menempilkan 500 kata dan jumlah kemunculannya

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()
```

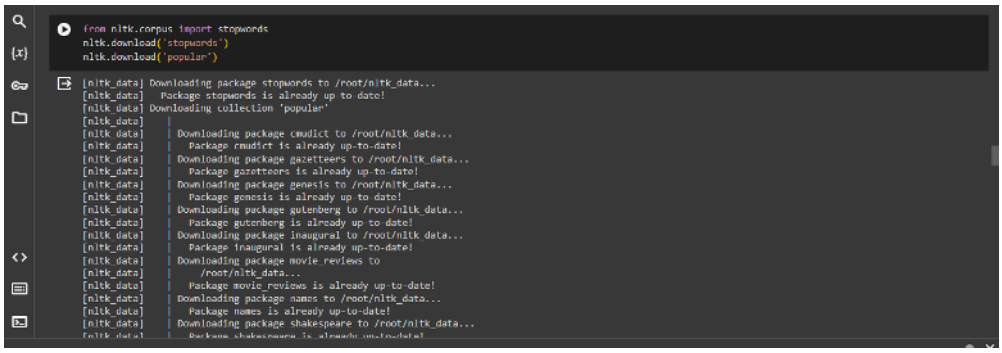
Gambar 4.29 Perintah Visualisasi 15 Kata Paling Sering Muncul



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.



Gambar 4.31 Proses Install dan Unduh Stopword

Gambar 4.32 Proses Install, memasukkan, dan menyimpan dalam dataset

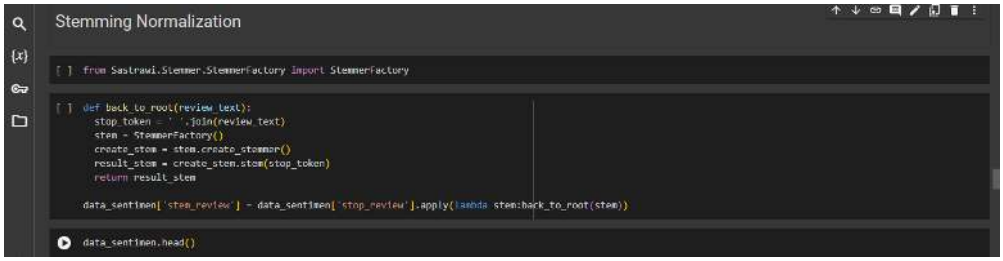
Gambar 4.33 Fungsi Yang Digunakan untuk Menghapus *Stopwords*

Sebelum	Sesudah *Remove Stopwords*
[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.



Gambar 4.34 Stemming Normalization

Sebelum	**Sesudah *Stemming Normalization***
[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*

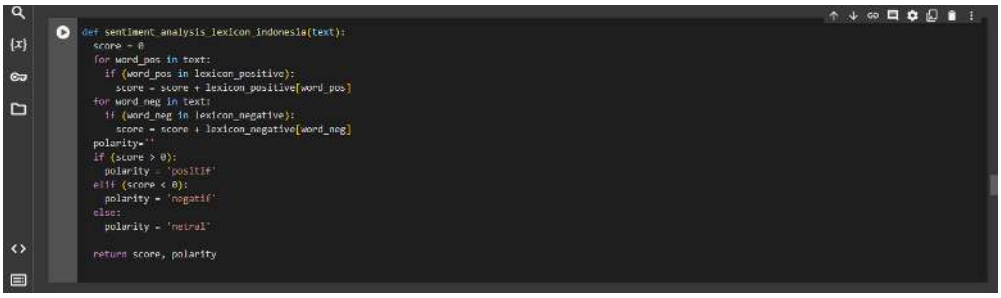
4.9. Pelabelan

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.

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.



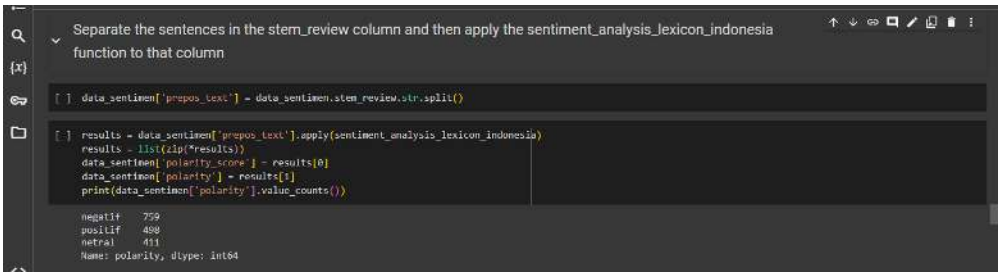
Gambar 4.35 Pembobotan sentimen

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

Gambar 4.36 *Conduct Functional Tests*

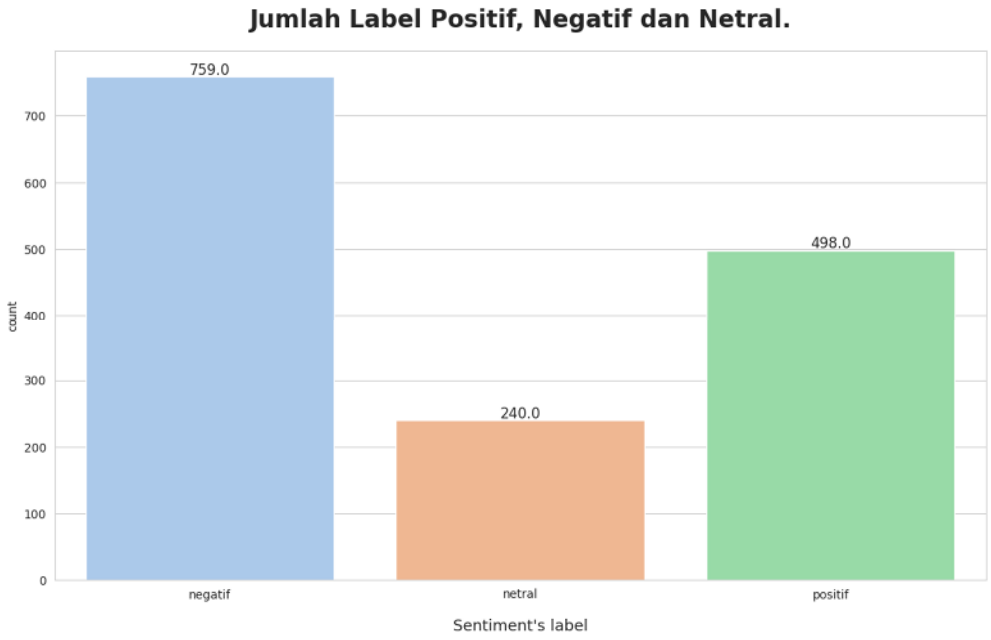
Pisahkan kalimat pada kolom stem_review lalu terapkan fungsi sentimen_analisis_lexicon_indonesia pada kolom yang baru yang bernama *polarity* sehingga diperoleh hasil negatif 759, positif 498, dan netral 411.



Gambar 4.36 Memisahkan Kalimat ke dalam kolom polarity

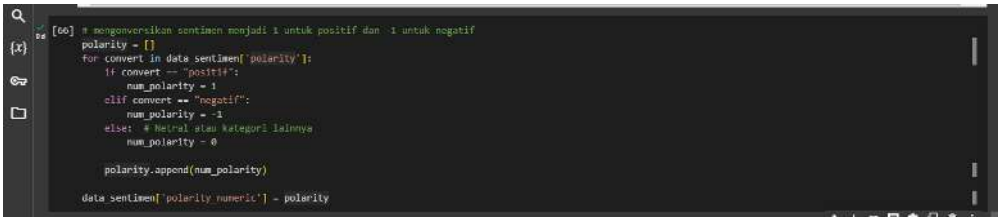
Pada tahap selanjutnya, penulis mengubah kembali spasi np.nan kemudian menghapusnya dengan menggunakan fungsi dropma dengan perintah "*data_sentimen = data_sentimen.replace (",np.nan, regex=True) data_sentimen =*

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 498dan, 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 disimpak 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



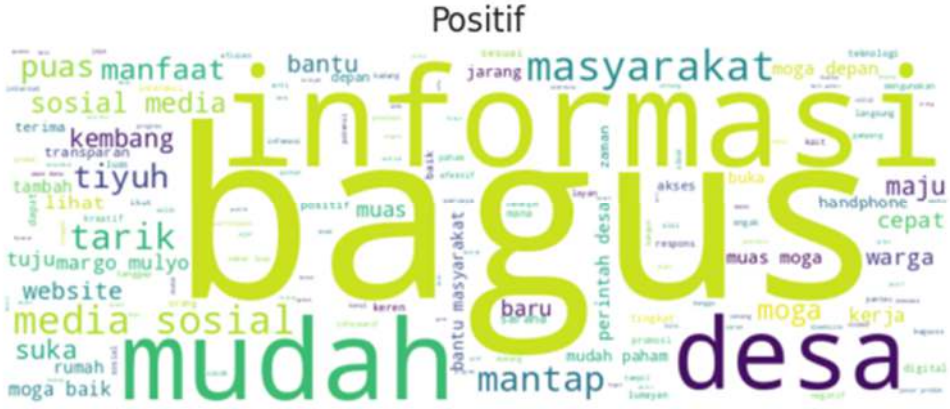
Gambar 4.37 mengonversikan sentimen

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%.

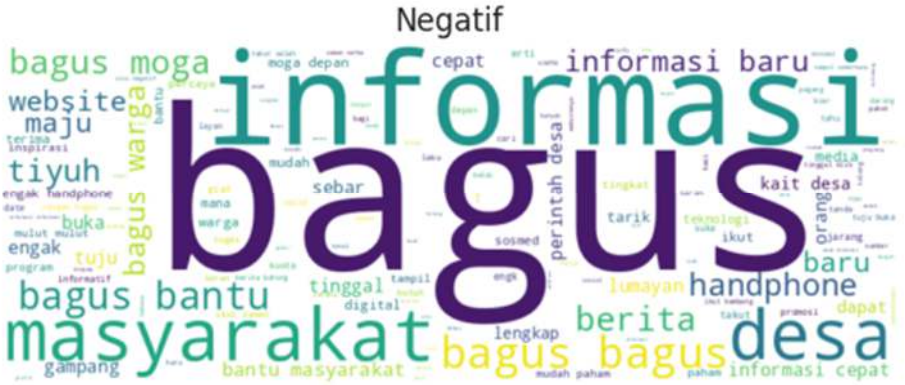
4.10. *Sentimen Wordcloud*

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.

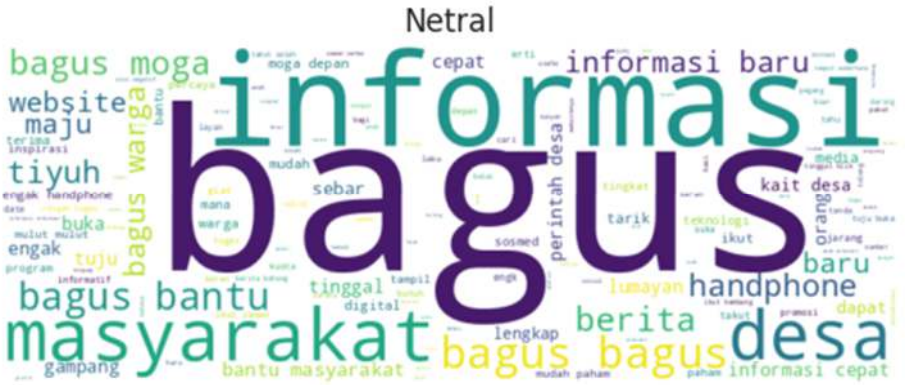
Gambar 4.38 *Wordcloud* Sentimen keseluruhan



Gambar 4.39 *Wordcloud* Sentimen Positif



Gambar 4.40 *Wordcloud* Sentimen Negatif



Gambar 4.40 *Wordcloud* Sentimen Netral

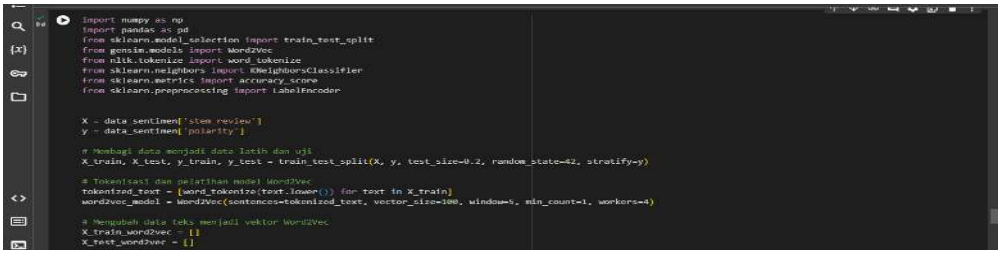
4.11. Klasifikasi Data dengan KNN

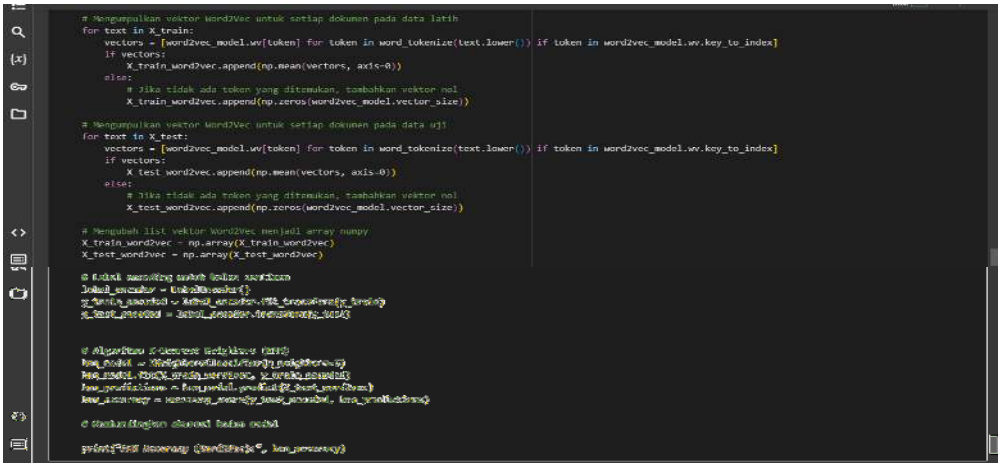
Sebelum data pelabelan dapat diolah menjadi data yang sesuai dengan klasifikasinya, KNN perlu melakukan ujicoba berupa data latih dan kemudian KNN akan melakukan tahapan uji setelah mendapatkan data dari ujicoba atau secara sederhana penulis membagi dua jenis data yaitu data latih dan data uji dengan 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

Gambar 4.41 *Wordcloud* Sentimen Netral

Kemudian dilanjutkan dengan perintah berikut





```
# 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 menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)
```

Gambar 4.41 Melihat Akurasi KNN

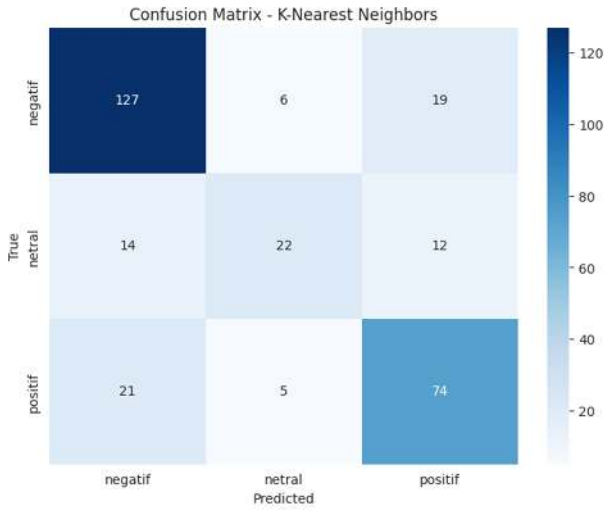
Dengan data latih diatas, KNN berhasil memperoleh akurasi sebanyak 0.7433333333333333.

4.12. Uji Performa dengan *Confusion Matrix*

Selanjutnya adalah pengujian performa algortima 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*

Predicat / Actual	Negatif	Netral	Positif
Negatif	127	6	19
Netral	14	22	12
Positif	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 hasil nya 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.

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

Sehingga dimasukkan nilainya menjadi

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

Dan hasilnya adalah **0,863945578**

Berikut adalah table hasil uji performa dengan *Confusion Matrix*

Confusion Matrix	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Hasil	**0,8340248963**	**0,858108108**	**0,8698630137**	**0,863945578**

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. Ektrak Fitur dengan TF-IDF

Setelah uji performa dilakukan dengan *Confusion Matrix* dilakukan dan diperoleh hasil yang baik, selanjutnya penulis melakukan ektrak 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