BAB 3
METODE PENELITIAN

Penelitian ini adalah penelitian rancang-bangun. Penelitian berawal dari latar belakan permasalahan yang ada, memetakan proses-proses, mencari sumber permasalahan, dan akhirnya merancang dan mengembangkan suatu sistem yang dapat digunakan untuk mereduksi atau mengeliminasi permasalahan yang ada.

3.1 BAHAN DAN ALAT PENELITIAN

Bahan yang akan dipergunakan pada penelitian ini adalah data *tweet* pengguna dengan keyword tentang layanan ojek *online* dan pengambilan data dilakukan menggunakan Jupyter Notebook dengan dicari berapa banyak data yang diperlukan.

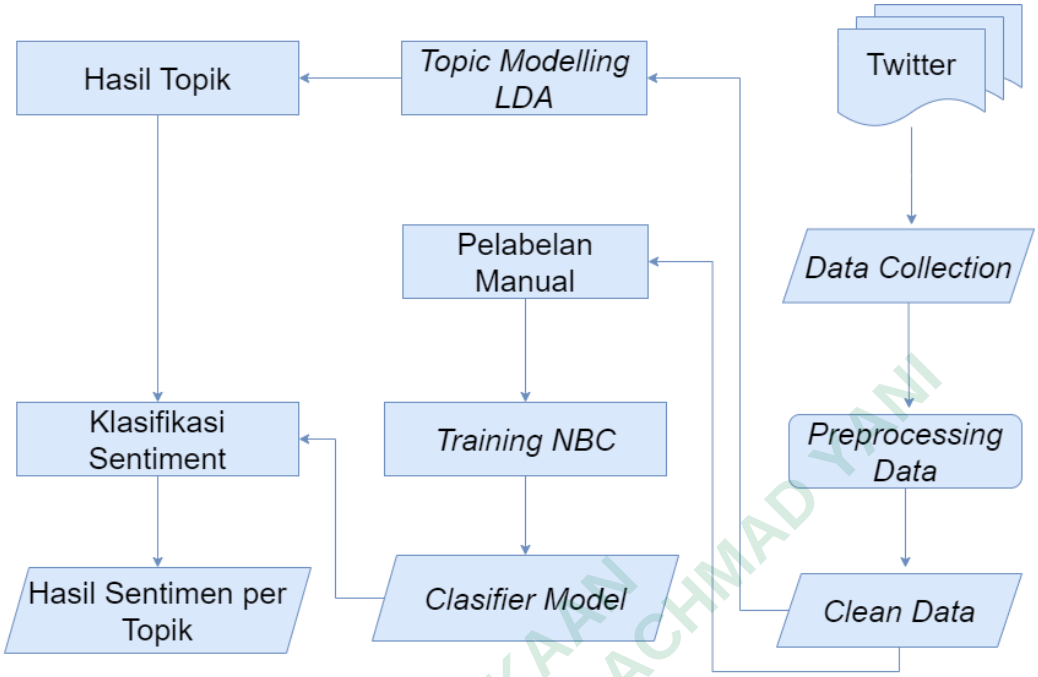
Alat yang akan digunakan untuk melakukan penelitian ini adalah laptop dengan kualitas dan spesifikasi yang cukup baik dan bisa untuk melakukan proses mengolah data serta sanggup untuk terkoneksi ke jaringan internet.

Sistem operasi dan program aplikasi yang akan digunakan dalam pembuatan web dashbord dalam pemodelan topik dan analisis sentimen adalah:

1. Sistem Operasi Windows 11 64-bit.
2. Bahasa Pemrograman Python 3.1.0.
3. Framework Flask
4. Microsoft Office Excel 2019.
5. Anaconda 3 64-bit.
6. Jupyter Notebook 3.5.2

3.2 JALAN PENELITIAN

Penelitian ini adalah penelitian tentang pemodelan topik yang nanti akan dilakukan analisis nilai sentimen pada masing-masing layanan dari beberapa *platform* ojek *online* untuk menjadi perbandingan. Pada penelitian ini nanti akan memakai algoritma Latent Dirichlet Allocation (LDA) dan Naïve Bayes Classifier (NBC), dapat dilihat alur penelitian tersebut pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

Hal pertama yang dilakukan pada penelitian ini adalah dari latar belakang permasalahan yang sudah dibahas, langkah awal akan melakukan data *collection* dari *tweet* pengguna Twitter yang membahas tentang jasa layanan ojek *online* dan selanjutnya dilakukan *preprocessing* pada data yang sudah dilakukan untuk mendapatkan *clean data*, kemudian akan dilakukan 2 tahapan yaitu pemodelan topik dengan menggunakan metode LDA dan melakukan pelabelan data yang diproses dengan *training* NBC, data yang dihasilkan dari 2 tahapan sebelumnya diproses untuk membentuk model klasifikasi yang nantinya dapat digunakan untuk mengklasifikasikan nilai sentimen dari beberapa layanan dengan hasil akhir dapat menghasilkan nilai sentimen pada layanan untuk menjadi bahan perbandingan dari beberapa *platform* ojek *online*. Maka berikut adalah hal yang dilakukan pada saat jalannya penelitian dalam melakukan pemodelan topik dan analisis nilai sentimen dari masing-masing layanan *platform* ojek *online*. langkah-langkah jalannya penelitian :

3.2.1 Pengumpulan data

1. *Data Collection* Gojek

Langkah awal dilakukan pengambilan dan pengolahan data dari *tweet* pengguna website Twitter dengan kata kunci Gojek Indonesia dengan kurun waktu pengambilan data dari tanggal 01 Janurai 2023 sampai 07 Mei 2023 dan didapatkan data sekitar 8000 *tweet* pengguna, menggunakan Anaconda Prompt dan kemudian data diproses di Jupyter Notebook yang akan menghasilkan kumpulan dokumen berbentuk file Microsoft Excel, dua data tersebut dapat dilihat pada Tabel 3.1.

Tabel 3.1 Data *Collection* Gojek

No	Waktu	Username	Teks
1	2023-05-17	,Snoopykeropi	Yang terbaru adalah hari ini saya kirim barang , di aplikasi saya membayar 15.500 tapi driver bilang saya dia hanya dpt 10.000 sambil nunjukin nilai rupiah di aplikasinya. Nah bagaimana itu jadi urusan saya? @gojekindonesia"
2	2023-05-17	Snoopykeropi	@gojekindonesia bisakah kalian tolong informasikan ke driver-driver kalian yg motor soal potongan dari kalian. Ini saya sudah 4 driver yg berbeda dng waktu dalam 2 minggu ini baik driver yg antar saya mauoun driver kiriman curhat dan mengeluh soal tarif yang mereka terima.

No	Waktu	Username	Teks
3	2023-05-17	ednaiseden	Hi@gojekindonesia dan @SpotifyID Saya agak bingung kok ada auto-payment untuk pembelian Spotify Premium sedangkan akun saya udah di-delete?Bulan sebelumnya juga terjadi hal serupa, sudah terpotong saldo tapi Spotify saya statusnya masih free.Bisa tolong dibantu pengecekan? Tks."
4	2023-05-17	izzahawardah	,"Aduh pelayanan kalian kok makin ngadi-ngadi sih makin jelek dan keliatan sekali pilih kasih, cari cuan cuan cuan terus @gojekindonesia
5	2023-05-17	irfanpuriindah	,@gojekindonesia Yg dibohongin mitra tapi ko minta gaji ke mitra dasar ga tau malu lanjutkan bohongmu sampai kiamat biar masuk neraka bareng bosmu gajinya mau tapi bantu yg gaji tiap bulan ga mau goride gue mana setan picek

Terlihat pada Tabel 3.1 terdapat beberapa jumlah data Gojek yang masih belum dilakukan *preprocessing* sehingga data tersebut belum dapat digunakan untuk pengolahan data lebih lanjut.

2. *Data Collection* Grab

Langkah kedua dilakukan pengambilan dan pengolahan data dari *tweet* pengguna website Twitter dengan kata kunci Grab dan Grab Indonesia dengan kurun waktu pengambilan data dari tanggal 01 Januari 2023 sampai 07 Mei 2023 dan didapatkan data sekitar 6000 *tweet* untuk Grab dan 1178 *tweet* untuk Grab

Indonesia pengguna menggunakan Anaconda Prompt dan kemudian data diproses di Jupyter Notebook yang akan menghasilkan kumpulan dokumen berbentuk file Microsoft Excel, data tersebut dapat dilihat pada Tabel 3.2.

Tabel 3.2 Data *Collection* Grab

No	Waktu	Username	Teks
1	2023-03-30	@bububibichan	iya bener sih ak aja seringnya pake grab skrng krn jegok gak ada promo lagi wkwkwkw ðŸ‘ŽðŸ‘ŽðŸ‘Ž kalah ama produk luar lol
2	2023-03-30	holaamerta	pakee saldo di sbux cardnya kak. sistemnya sama kek gojek grab gitu. dan kalo mau pake rewardnya setauku harus pake sbux card bayarnya. nah plusnya, kalo saldo kurang bisaa split paymentðŸ™†ðŸ • »â€• â™€ï¸ • "
3	2023-03-30	kadalindia	Dipikir2 wanian oge kamari. Pulang dr senayan ke st palmerah naik grab. Motornya pake motor repsol balap gt. Abangnya nanya ""pulang kemana kak"" ""ke bintaro xchnge bang"" ""itu yg lurus ke graha raya kan?"" ""Iya bang"" ""waahhh
4	2023-03-30	lilousdump	kalo disini gue bisa naek grab kalo disana gue kudu apa ajg? naik delman istimewa ku duduk di muka? kata gue mending acaranya dibikin perkecamatan aja biar gue tinggal jalan 5 langkah (re: rumah alpin ma bestie)
5	2023-03-30	bapakojoll	Jamane grab point e smpek jam 9 , gara2.iki gasido nutup 15orderan"

Terlihat pada Tabel 3.2 terdapat beberapa jumlah data Grab yang masih belum dilakukan *preprocessing* sehingga data tersebut belum dapat digunakan untuk pengolahan data lebih lanjut.

3.2.2 *Data Preprocessing*

Setelah melakukan tahapan pengambilan data dilakukan tahapan *prepocessing* data berikut ini tahapan-tahapan pada *preprocessing* data. Sebelum masuk ke tahap *preprocessing* data masukkan *library* yang dapat dilihat pada kode dibawah ini.

```
import numpy as np
import pandas as pd
from pandas import DataFrame
import nltk, re, string
from pandas import DataFrame
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

Pada kode diatas digunakan untuk memasukkan *library* serta modul yaitu nltk, emoji, nltk.tokenize, *stopwords* untuk melakukan *preprocessing*

1. Cleaning

Tahapan *cleaning* adalah tahapan untuk menghilangkan atribut-atribut yang tidak diperlukan seperti tag, emoji, dan tag url yang dapat dilihat pada kode dibawah ini.

```
def url_remove(Text):
    t = re.sub(r'http\S+', '',Text)
    return t
def punc_remove(Text):
    t = re.sub(r'[^\w\s]','', Text)
    return t
def rt_remove(Text):
    t = re.sub(r'RT[\s]+', '', Text)
    return t
```

```
def number_remove(Text):
    t = re.sub('[0-9]+', '', Text)
    return t
def slang_remove(Text):
    t = re.sub(r'\\n', " ", Text)
    return t
def regex_remove(Text):
    t = re.sub("b'", " ", Text)
    return t
def remove_user(Text):
    t = re.sub('@[^\s]+','', Text)
    return t
def hashtag_remove(Text):
    reg = "#(\w+:\/\/S+)"
    return re.sub(reg, " ", Text)
cleaned = []
def clean_text(Text):
   for i in Text:
cleaned.append(url_remove(punc_remove(number_remove
(remove_user(regex_remove(hashtag_remove(rt_remove
(slang_remove(re.sub("[\n\r\t\xa0]","",i).strip())))))
)))))clean_text(df["Text"])
```

Sudah terdapat berbagai macam fungsi pada kode diatas dari line pertama untuk menghilangkan url pada data *tweet*, line ke tujuh menghilangkan emoji, dan line delapan ada hastag pada Twitter setelah dilakukan *cleaning* pada data yang tersimpan data akan terlihat bersih.

a. Data Bersih Gojek

Terlihat pada Tabel 3.3 data Gojek yang sudah dilakukan cleaning sudah bersih dari url, emoji serta hastag disini data sudah bersih tetapi belum dilakukan proses-proses selanjutnya.

Tabel 3.3 Data Bersih Gojek

No	**Teks_Bersih**
1	maaf kak food sama ada doble order seperti systemnya sayang sekali keringet driver manfaat kan system
2	nggak bagus kalo semua company butuh balik modal buat survive kalo company nggak survive mau makan apa drivernya
3	potensi gelombang subvarian omicronn berat varian delta
4	dibales dong isi dmnya dan teman teman jangan cuma baca
5	bantu jawab enggak bang jarang banget banget banget driver dapet order udah
6	sejak gojek pegakng chindo makin keji potongaknnyakenapa si org chindo

b. Data Bersih Grab

Terlihat pada Tabel 3.4 data Grab yang sudah dilakukan cleaning sudah bersih dari url, emoji serta hastag disini data sudah bersih tetapi belum dilakukan proses-proses selanjutnya.

Tabel 3.4 Data Bersih Grab

No	**Teks_Bersih**
1	hampir nyungsep naik grab ugal
2	grab makin kedekut nampaknya
3	celaka kalian yang aku keluar grab rumah office vice versa tiap kali problem
4	ini gimana pesan dominos lewat grab garden shopping arcade uda jaman dateng pas telfon malah dimatiin chuakss
5	akhir dapet grab electric jadi tau nyata ini sewa grab per hari buset mihil terus isi baterai lewat token alfamartmidi batu baterai dipake segera isi ulang
6	iyasih aku dengernya kayak gitu naik grab susah mana mahal untung aku nginep regentown jadi tinggal jalan kaki aja deket

2. Tokenizing

Pada tahapan ini berfungsi untuk memisahkan sebuah kata dari kalimat agar menjadi kata tunggal agar kata tersebut dapat berdiri sendiri *library* yang dapat digunakan untuk proses *tokenizing* serta perintah-perintah yang dapat digunakan untuk *tokenizing* dapat dilihat pada kode dibawah ini.

```
def removeStopWords(Text):
    dictionary = ArrayDictionary(stopwords)
    str = StopWordRemover(dictionary)
    token = word_tokenize(str.remove(Text))
    return ' '.join(token)
df['text'] = df.text.apply(removeStopWords)
df
```

Pada kode diatas digunakan untuk memasukkan proses *tokenizing* dan berfungsi untuk memisahkan kata dari kalimat.

3. *Case folding* dan *Stopwords*

Menghilangkan kata-kata yang tidak diperlukan serta atribut-atribut yang sering muncul serta merubah semua huruf menjadi huruf kecil perintah-perintah tersebut dapat dilihat pada kedua kode dibawah untuk perintah *case folding* untuk kode dibawahnya lagi untuk perintah *stopwords*.

```
def lowercase(text):
    lower_word = text.lower()
    return lower_word
df['text'] = df.text.apply(lowercase)
df
```

Dari kode yang sudah terlihat diatas terdapat perintah-perintah untuk merubah kata-kata menjadi huruf kecil.

```
From Sastrawi.StopWordRemover.StopWordRemoverFactory import
```

```
StopWordRemoverFactory,ArrayDictionary,StopWordRemover
factory = StopWordRemoverFactory()
more_stopword=['krn','nsa','ic','dw','shl','jhn','wtb','hyne']
stopword = factory.create_stop_word_remover()
stopwords = factory.get_stop_words()+more_stopword
print(stopwords)
```

Dari kode yang terlihat diatas sudah terlihat kata-kata yang tidak perlu lagi digunakan akan dibersihkan dalam proses ini.

a. *Stopwords* Gojek

Daftar kata pada Gojek yang sudah bersih hasilnya dapat dilihat pada Tabel 3.5.

Tabel 3.5 Daftar *Stopwords* Gojek

Kata
, 'krn', 'g', 'rb', 'ngebid', 'maja', 'maulana', 'adnan', 'tksibl', 'lai', 'clavinova', 'clara', 'ryan', 'sindy', 'indra', 'sali', 'yohan', 'd', 'pasu', 'kasihnsa', 'kait', 'log', 'tksari', 'kasihbona', 'andreo', 'ya', 'ara', 'bs', 'ga', 'gin', 'amp', 'tp', 'sen', 'sm', 'hr', 'ara', 'kasihfeb', 'goto', 'x', 'yb', 'kasihaln', 'kasihyun', 'kasihyni', 'kasihekw', 'gw', 'ka', 'rp', 'dm', 'b', 'lg', 'jg', 'sy', 'tksjhn', 'dr', 'dah', 'tdk', 'tks', 'tksari', 'tksfeb', 'tksast', 'fee', 'nsa', 'yni', 'ibl', 'nsa', 'ic', 'dw', 'shl', 'jhn', 'wtb', 'wts', 'qr', 'tbk', 'rt', 'dips', 'bp', 'kkl', 'ml', 'pk', 'sc', 'gbs', 'cp', 'hyne', 'hya', 'nu', 'pkkmb', 'ltkm', 'un', 'cm', 'tj', 'uk'

Tabel 3.5 adalah daftar kata-kata pada Gojek yang tidak lagi digunakan atau kata- kata yang tidak memiliki makna sehingga perlu dilakukan proses *stopwords*.

b. *Stopwords* Grab

Daftar kata pada Gojek yang sudah bersih hasilnya dapat dilihat pada Tabel 3.6.

Tabel 3.6 Daftar *Stopwords* Grab

Kata
'hlo', 'g', 'c', 'fd', 'grgr', 'rbu', 'bs', 'rprb', 'dtg', 'jd', 'sy', 'sm', 'gw', 'eh', 'ga', 'nct', 'gin', 'taeil', 'pasta', 'porto', 'murce', 'itv', 'aa', 'aab', 'mawar', 'ka', 'ma', 'bu', 'ka', 'x', 'matchy', 'tu', 'byk', 'eysisi', 'pools', 'esok', 'mino', 'kt', 'rajut', 'lebak', 'jugan', 'pa', 'mas', 'ss', 'zarzou', 'lawa', 'shio', 'lepastu', 'yaaa', 'rbpcs', 'sulam', 'la', 'takde', 'korang', 'tak', 'cekiceki', 'yb', 'ht', 'jp', 'shawl', 'dedi', 'tu', 'tak', 'ye', 'tk', 'mcm', 'pastu', 'nk', 'kat', 'dah', 'mmg' 'utk', 'dg', 'joy', 'ava', 'suga', 'ph', 'lg', 'peluk', 'chinese', 'wak', 'hannan', 'nang', 'pct', 'lalamove', 'medispa', 'tdk', 'sd', 'rb', 'tp', 'lglsg', 'dr', 'gin', 'ko', 'asa', 'kr', 'ken', 'sembah', 'encik', 'putus', 'geng', 'kisah', 'yaw', 'taeil', 'minit', 'doyoung', 'lrt', 'sgitu', 'xiaojun', 'sop', 'kr', 'nak', 'mak', 'sia', 'yuta', 'msh', 'exp', 'si', 'km', 'dah', 'rp', 'ak', 'bni', 'kl', 'tng', 'utk', 'nder', 'ga', 'dm', 'pc', 'km', 'utk', 'ava', 'pct', 'min', 'makcik', 'dine', 'uolls', 'dedi', 'slot', 'rprb', 'rm', 'amp', 'aab', 'aaabbbb', 'thailand', 'aaaabbbbbb', 'wts', 'svt', 'sbb', 'jb', 'shl', 'jhn', 'wtb', 'wts', 'qr', 'tbk', 'rt', 'dips', 'bp', 'kkl', 'ml', 'pk', 'sc', 'gbs', 'cp', 'hyne', 'hya', 'nu', 'pkkmb', 'ltkm', 'un', 'cm', 'tj', 'uk', 'ama', 'ringgit', 'nge', 'png', 'p', 'pkp', 'drpd', 'gel', 'jaring', 'yaa', 'bini', 'mei', 'nge', 'qris', 'kea', 'ni', 'last', 'sk', 'jela', 'mikael', 'header', 'g', 'now', 'berbaloi', 'taw', 'kurta', 'plat', 'now', 'ka', 'pak', 'aju', 'lil', 'cair', 'bot', 'sila', 'cdm', 'di', 'gilir', 'serpong', 'harini', 'abg', 'go', 'ckp', 'bas', 'new', 'haa', 'tengok', 'set', 'pon', 'meh', 'tiktok', 'haritu', 'jer', 'try', 'cakap', 'taknak', 'dh', 'n', 'rani', 'ha', 'mostly', 'leh', 'pes', 'org','smlm', 'indonesia', 'ka', 'sbtsell', 'da', 'spt', 'think', 'ah', 'matte', 'silah', 'ic'.

Tabel 3.6 adalah daftar kata-kata pada Grab yang tidak lagi digunakan atau kata- kata yang tidak memiliki makna sehingga perlu dilakukan proses *stopwords*.

4. *Stemming*

Proses untuk merubah semua kata menjadi kata dasar perintah tersebut dapat dilihat pada kode dibawah ini.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()
def stemming(text):
   hasil = stemmer.stem(text)
          return hasil
df['text'] = df.text.apply(stemming)
df
```

Kode diatas digunakan untuk melakukan proses *stemming* pada data *tweet*.

5. *Normalization*

Proses untuk merubah kata-kata yang disingkat menjadi sebuah kata yang utuh seperti : “yg” :“yang”, “dgn”:”dengan”, dan lain-lain perintah tersebut dapat dilihat pada kode dibawah ini.

```
normalizad_dict={'jd':'jadi','sdh':'sudah','dariiver
':'driver','kyk':'kayak','yg':'yang','bsk':'besok','
utk':'untuk','bgt':'banget','dgn':'dengan','gmn':'gi
mana'}
import re
normalizad=re.compile('(%s)'%'|'.join(normalizad_dic
t.keys()))
def expand_normalizad(s, normalizad_dict=normalizad_dict):
      def replace(match): return
normalizad_dict[match.group(0)]
            return normalizad.sub(replace, s)
df['text'] = df.text.apply(expand_normalizad)
```

Hasil perintah diatas ada sebuah kata-kata yang diubah menjadi kalimat yang baku dan kalimat tersebut berisi sebuah kata-kata yang perlu dinormalisasikan.

a. Kata Normalisasi Gojek

Kata-kata yang sudah dinormalisasikan pada data Gojek dapat dilihat pada Tabel 3.7.

Tabel 3.7 Tabel Kata Normalisasi Gojek

Sebelum dinormalisasi	Sesudah dinormalisasi
dgn	dengan
gmn	gimana
bgt	banget
utk	untuk
bsk	besok
utk	untuk
yg	yang
jg	juga
klo	kalau
kpd	kepada

b. Kata Normalisasi Grab

Kata-kata yang sudah dinormalisasikan pada data Grab dapat dilihat pada Tabel 3.8.

Tabel 3.8 Tabel Kata Normalisasi Grab

Sebelum dinormalisasi	Sesudah dinormalisasi
Pdhl	padahal
blm	belum
trs	terus
skrng	sekarang
dpt	dapat
utk	untuk
yg	yang

3.2.3 *Clean Data*

Tahapan dimana data sudah benar-benar bersih dan sudah dapat digunakan untuk proses selanjutnya perintah untuk menyimpan *clean data* yang sudah benar-benar bersih tersebut dapat dilihat pada kode dan kata-kata tersebut di Tabel 3.9 dan Tabel 3.10.

```
df.to_excel('gojekfix.xlsx', index=False)
```

Pada perintah diatas hasil dari data yang sudah di *preprocessing* akan disimpan atau diunduh dalam bentuk file Excel.

1. Data Bersih Gojek

Hasil dari proses *preprocessing* pada data Gojek dapat dilihat pada Tabel 3.9.

Tabel 3.9 Daftar Data Bersih Gojek

No	**Teks_Bersih**
1	mau transaksi nyaman tidak repot kok malah jadi beginii pusing di telpon driver minta uang lebih bilang buat susu anak bikin jadi orang jahat kalau gakk kasih
2	sama grab metode kirim jadi cepet lambat hemat lama kirim reguler hari biar basi tuh makan
3	kejam kali aplikator di boikot aja kalo sistem gakk manusiawi
4	makin gakk ngehargain kerja org lapangan banget
5	memang sekarang makin serakah gojek driver makin siksa
6	kalian terlalu dzolim driver dapat double order gofood
7	iya bang sebagai driver mau ngadu gimana tetap driver selalu salah
8	sip catat resto ngeklik pesan siap gobiznya masih batalin order
9	masuk driver cuma segitu sadis gue yakin kalo pesan masuk driver customer pasti bayar normal kasian banget driver gojek
10	padahal gojek ongkirnya lebih mahal banyak tambah biaya promonya susah banget

2. Data Bersih Grab

Hasil dari proses *preprocessing* pada data Grab dapat dilihat pada Tabel 3.10.

Tabel 3.10 Daftar Data Bersih Grab

No	**Teks_Bersih**
1	mau pesan makan terus dibatalin sama resto tutup saldo ovo gak balik gimana dong kak plis
2	grabfood nya tolong lebih baik
3	sekarang grab mahal
4	grab layanan anter jemput nya agak lama
5	pesen makan tapi datang nya lama banget
6	halo grab mau tanya ada cara beri tip driver kalau lanjut beri rating
7	halo kak mohon maaf kak jadi gak nyaman bisa aku bantu cek infoin dm ketik bantu dm hubung langsung agent terima kasih
8	ekspresi ku order kopi grabfood
9	udah telpon cs berkali-kali hasil nihil percuma
10	maaf kak bikin tidak nyaman aku bantu cek dm lebih lanjut terkait kakak bisa langsung klik tombol bawah ini dan ketik bantu

3.2.4 *Topic Modelling* Menggunakan LDA

Setelah melakukan *preprocessing* pada data akan dilakukan pemodelan topik menggunakan algoritma LDA dengan *library* yang dapat dilihat pada kode kode dibawah ini.

```
NUM_TOPICS = 30
ldamodel = gensim.models.ldamodel.LdaModel
(corpus, num_topics = NUM_TOPICS, id2word=
dictionary, passes=30)
ldamodel.save('modelLDA_30.gensim')
 topics = ldamodel.show_topics(num_topics=3
 0,num_words=250)
 for topic in topics:
 print(topic)
```

Pada kode diatas memasukkan *library* gensim, lalu menyimpan modelLDA ke dalam file modelLDA 5.gensim, serta menampilkan model LDA.

```
x=ldamodel.show_topics(num_topics=5,
num_words=250,formatted=False)
topics_words = [(tp[0], [wd[0] for wd in tp
      [1]]) for tp in x]
#Below Code Prints Topics and Words
for topic,words in topics_words:
print(str(topic)+ "::"+ str(words))
print()
#Below Code Prints Only Words
topik=[]
for topic,words in topics_words:
      topik.append(" ".join(words))
```

Pada kode diatas untuk menampilkan topik yang telah didapatkan dari perulangan dari “for topic,words in topics_words:” dan “print(str(topic)+ "::"+ str(words))”

1. *Visualisasi*

Topik-topik yang sudah terkumpulkan yang telah didapatkan dari tahapan *topic modelling* akan di*visualisasikan* dengan library, dan file yang telah tersimpan, serta perintah yang dapat dilihat pada kode dibawah ini.

```
dictionary_lda = gensim.corpora.Dictionary.load
('dictionary_lda_gojek.gensim')
corpus_lda = pickle.load(open
('corpus_lda_gojek.pkl', 'rb'))
lda = gensim.models.ldamodel.LdaModel.load
('modelLDA_5_gojek.gensim')
import pyLDAvis.gensim
lda_display = pyLDAvis.gensim.prepare
(lda, corpus, dictionary, sort_topics=True)
```

```
pyLDAvis.display(lda_display)
```

Pada kode ditas dipergunakan untuk memasukkan modul *library* pyLDAvis.gensim, serta memuat file yaitu ada dictionary_lda.gensim, *corpus*_lda.pkl dan modelLDA_5.gensim, akan ditampilkan *topic modelling* dari file yang telah dimuat. Selain itu, topik dapat divisualisasikan dengan kata-kata dalam topik, dimana ukuran setiap kata akan menampilkan frekuensi atau pentingnya kata dalam topik. Sebelum memasuki *library* dan perintah yang dapat dilihat di kode-kode dibawah ini.

```
from os import path
from PIL import Image

from Sastrawi.StopWordRemover.StopWordRemoverFactory
Import
StopWordRemoverFactory,StopWordRemover,ArrayDictionary
import matplotlib.pyplot as plt
```

Pada kode diatas akan digunakan untuk memasukkan modul *library* path, image, *wordCloud*, *stopwords*, *imagecolorgenerator*, serta *matplotlib.pyplot*

```
my_list=tops
wordcloud   =   WordCloud(width   =   1000,
        height   =   600,
background_color="white").generate(my_list)
plt.figure(figsize=(15,8)) plt.imshow(wordcloud)
plt.axis("off")
plt.savefig("wordcloud_all"+".png", bbox_inches
='tight')
plt.show()
plt.close()
```

Pada kode diatas dipergunakan untuk menampilkan *wordcloud* dari keseluruhan topik serta menyimpanya dengan nama file *wordcloud*_all.png.

```
tops=""
for top in topik:
    tops=tops+"".join(top)
       my_list=tops
              wordcloud = WordCloud(width = 1000,
              height = 600,
               background_color="white").
              generate(my_list)
       plt.figure(figsize=(15,8))
       plt.imshow(wordcloud)
       plt.axis("off")
              plt.savefig("wordcloud_all"+".png",
       bbox_inches='tight')
       plt.show()
       plt.close()
```

Pada kode diatas dipergunakan untuk menampilkan *wordcloud* dari setiap masing-masing topik serta menyimpannya dengan nama file your_file_name.png.

2. Analisis

Pada tahap akhir, kualitas topik yang dihasilkan dari *topic modelling* diuji dengan menggunakan hasil *topic coherence* yang disajikan dalam grafik. Berikut ini adalah beberapa perintah dan *library* yang digunakan, seperti terlihat pada kode- kode dibawah ini.

```
ldatopics = ldamodel.show_topics(formatted=False)
```

Pada kode diatas tujuannya untuk membuat variable ldatopics dan memanggil topik dari proses *topic modelling*.

```
from gensim.models import CoherenceModel, LdaModel,
LsiModel, HdpModel
def evaluate_graph(dictionary, corpus, texts, limit):
 """
 Function to display num_topics - LDA graph using
    c_v coherence
 Parameters:
```

```
----------
dictionary : Gensim dictionary
corpus : Gensim corpus
limit : topic limit
Returns:
-------
lm_list : List of LDA topic models
       c_v : Coherence values corresponding to the
   LDA model with respective number of topics
"""
c_v = []
lm_list = []
for num_topics in range(1, limit):
   lm=LdaModel(corpus=corpus,
   num_topics=num_topics,
   id2word=dictionary) lm_list.append(lm)
   cm=CoherenceModel(model=lm,texts=texts,dictionary=
   dictionary, coherence='c_v')
    c_v.append(cm.get_coherence())
x = range(1, limit)
plt.plot(x, c_v)
plt.xlabel("num_topics")
plt.ylabel("Coherence score")
plt.legend(("c_v"), loc='best')
plt.show()
return lm_list, c_v
```


Pada kode diatas dipergunakan untuk memasukkan *library* CohorenceModel, LdaModel, LsiModel, HdpModel serta membentuk fungsi untuk memproses topic cohorence dari "def evaluate_graph(dictionary, *corpus*, texts, limit):" sampai "return lm_list, c_v".

```
%%time
lmlist, c_v =
evaluate_graph(dictionary=dictionary,
corpus=corpus, texts=text_data, limit=30)
```

Pada kode diatas dibuat untuk menampilkan grafik dari *topic coherence* sehingga dapat diketahui jumlah topik idealnya dengan jumlah grafik tertinggi.

3.2.5 Naïve Bayes Classifier

Setelah melakukan pemodelan topik menggunakan algoritma LDA selanjutnya akan dilakukan pengolahan data menggunakan metode Naïve Bayes Classifier (NBC) yang akan melewati beberapa tahap seperti pelabelan manual, feature extraction TF-IDF, *training*, *testing*, dan klasifikasi data dari masing-masing *platform*.

3.2.6 Pelabelan Manual

Pelabelan manual adalah proses memberikan label terhadap kalimat dan kata yang terdapat pada sebuah dokumen sehingga nantinya dapat dilakukan analisis lebih lanjut mengenai sifat dari kalimat atau kata tersebut apakah memiliki sifat positif atau negatif. Dari data *tweet* yang sudah dilabeli pada tahap ini maka telah didapatkan data *training* sebanyak 1134 untuk data *tweet* Gojek dan data *tweet* Grab, dengan jumlah data *tweet* Gojek positif 646 dan 487 data *tweet* negatif, sedangkan Grab memiliki data positif 777 dan negatif 356.

1. Hasil Pelabelan Manual Gojek

Hasil dari pelabelan manual pada data Gojek dengan label positif dan negatif dapat dilihat pada Tabel 3.11.

Tabel 3.11 Pelabelan Manual Gojek

No	Teks_Bersih	Label	Kelas
1	jam pagi sekarang x drivernya cancel pickup paket ada sistem punishment buat driver yang cancel emang	negatif	0
2	sejak kapan gojek akun bodong beginii mba wkwkkw	positif	1
3	aneh banget sih jabodetabek gakk masuk promo	positif	1
4	gimana apakah emang udah sering kayak gini	positif	1
5	ayo susahin drivernya min biar makan cape doang wkwk	positif	1

Dari Tabel 3.11 terdapat informasi bahwa label positif diberi kelas dengan nilai 1 sedangkan untuk label negatif diberi kelas dengan nilai 0. Pelabelan manual ini dilakukan untuk memberi nilai sentimen terhadap kelas positif maupun negatif yang nantinya akan dihitung nilai akurasinya.

2. Hasil Pelabelan Manual Grab

Hasil dari pelabelan manual pada data Grab dengan label positif dan negatif dapat dilihat pada Tabel 3.12.

Tabel 3.12 Pelabelan Manual Grab

No	Teks_Bersih	Label	Kelas
1	ngapain ini sini naik grab nyampe bank	negatif	0
2	gimana grab udah beli voucher free ongkir tulis free tetep aja kena charge	negatif	0
3	siapa yang baru login grab no office aku dah tiga kali sms	negatif	0
4	Bisa bisanya telat bangun nyempetin order makan grab	positif	1
5	enak banget kak nyesel beli yang kecil padahal diskon lumayan grab	positif	1
6	jadi inget waktu mau beli aqua aja pesan pake grab	positif	1

Dari Tabel 3.12 terdapat informasi bahwa label positif diberi kelas dengan nilai sedangkan untuk label negatif diberi kelas dengan nilai 0. Pelabelan manual ini dilakukan untuk memberi nilai sentimen terhadap kelas positif maupun negatif yang nantinya akan dihitung nilai akurasinya.

Contoh hasil data yang sudah diberikan label otomatis dari sistem dapat dilihat pada Gambar 3.2 dan Gambar 3.3.

1. Hasil Pelabelan Otomatis Sistem Gojek

Data pelabelan manual pada Gojek dengan jumlah 1134 *tweet* untuk digunakan pada *training* dan dapat dilihat pada Gambar 3.2.

	text	Sentimen
0	wahahaha gua pernah dapet ni ambas satpol mera...	negatif
1	top up aja cepet gilir mau trafer gopay no rek...	negatif
2	baru hari kirim barang aplikasi bayar driver b...	negatif
3	berita lagi rame hoax kak	negatif
4	sama kayak td sore jalan batu tumbuh jati cemp...	negatif
...	...	...
1128	haha jam jam gakk order padahal cuma minus rib...	positif
1129	jadi tuh wegotyou banget punya fitur perjalana...	positif
1130	min cek dm ya thanks	positif
1131	sering lihat motor gojek bawa barang super bes...	positif
1132	hai kak aku saranin kamu untuk konfirmasi lebi...	positif

1133 rows × 2 columns

Gambar 3.2 Data Pelabelan Otomatis Gojek

2. Hasil Pelabelan Otomatis Sistem Grab

Data pelabelan manual pada Grab dengan jumlah 1134 *tweet* untuk digunakan pada *training* dan dapat dilihat pada Gambar 3.3.

	text	Sentimen
0	reckless nya driver i jarang bad rating grab d...	negatif
1	almost babe hujan stop je i terus blkg and jal...	negatif
2	mumpung diskon anjrittt tengah harga lebih bah...	negatif
3	pakee saldo sbux cardnya kak sistem sama kek g...	negatif
4	hai kak herri buat minta aju yang kakak sampai...	negatif
...	...	...
1128	iyanihh bener gua punya alam mau dispill kasi ...	positif
1129	nah dialihin biar happy pesan makan bareng gra...	positif
1130	wah nungguin promo grab disney hotstar masuk a...	positif
1131	hi moots lup pacar jual keychain rv hehe siapa...	positif
1132	pinter cara rayu sayaetannnn gosah sok knalll ...	positif

1133 rows × 2 columns

Gambar 3.3 Data Pelabelan Otomatis Grab

3.2.7 *Data Training*

Proses data *training* dimulai dengan ekstraksi pada data teks menggunakan TF-IDF, pada tahap ini digunakan pendekatan Naive Bayes Classifier, dan dilanjutkan dengan proses *training* data untuk membentuk model klasifikasi yang disimpan dengan format pickle yang nantinya dapat dipergunakan untuk mengklasifikasikan sentimen data dari masing-masing *platform* secara otomatis. Contoh penghitungan TF-IDF secara manual dengan Microsoft Office Excel ditunjukkan pada Tabel dibawah ini.

1. TF-IDF Gojek

Dokumen yang digunakan untuk TF-IDF untuk Gojek dapat dilihat pada Tabel 3.13.

Tabel 3.13 Dokumen TF-IDF Gojek

Dokumen (d)	Kalimat
d1	admin kok akun gak pernah ada promo voucher gofood
d2	pihak gojek membuat potongan biaya gofood dan goride menjadi rugi
d3	Harga bisa lebih murah dengan menggunakan promo voucher untuk gofood dan goride
d4	akun kena soft banned saya pernah dapet voucher goride gocar gofood gak bisa claim kode promo mohon bantu bisa nyaman dan aktif guna gojek terima kasih

Pada Tabel 3.13 terdapat beberapa kata yang nantinya akan digunakan untuk menghitung TF-IDF secara manual dengan jumlah 4 dokumen yaitu d1, d2, d3, d4 melakukan perhitungan TF ini akan digunakan beberapa komponen seperti term atau kata dan d adalah jumlah dari komponen data yang akan digunakan yaitu d1, d2, d3 dan d4 dan terdapat df untuk melakukan perhitungan jumlah term yang nantinya muncul dari setiap dokumen.

2. TF-IDF Grab

Dokumen yang digunakan untuk TF-IDF untuk Grab dapat dilihat pada Tabel 3.14.

Tabel 3.14 Dokumen TF-IDF Grab

Dokumen (d)	Kalimat
d1	Gak pernah dapet promo voucher buat ngegrabfood hilang tiba-tiba
d2	Order grabcar pagi payah susah sama jasa deliverynya mahal

Dokumen (d)	Kalimat
d3	cocok kok waktu pulang ngantor stasiun rumah pake grab electric nyaman banget
d4	driver grab gak chat dulu gitu juga langsung nelpon nyepam ganggu

Pada Tabel 3.14 terdapat beberapa kata yang nantinya akan digunakan untuk menghitung TF-IDF secara manual dengan jumlah 4 dokumen yaitu d1, d2, d3, d4 melakukan perhitungan TF ini akan digunakan beberapa komponen seperti term atau kata dan d adalah jumlah dari komponen data yang akan digunakan yaitu d1, d2, d3 dan d4 dan terdapat df untuk melakukan perhitungan jumlah term yang nantinya muncul dari setiap dokumen. Contoh dari perhitungan TF itu sendiri dapat dilihat pada dibawah.

3. Perhitungan pada TF Gojek

Pada kalimat dokumen Gojek yang telah dimasukan ke dalam perhitungan TF dapat dilihat pada Tabel 3.15, dan untuk penjelasan lengkapnya dapat dilihat pada lampiran 1.

Tabel 3.15 Perhitungan pada TF Gojek

***Term* (kata)**	**d1**	**d2**	**d3**	**d4**	**df**
admin	1				1
kok	1				1

Pada Tabel 3.15 memberikan penjelasan terhadap kemunculan term atau kata didalam sebuah kalimat yang telah dimasukan. Pada perhitngan IDF ini akan menggunakan beberapa komponen seperti term atau kata, serta tf dan idf yang terhubung dari kesedian suatu term dari semua dokumen, di hitung menggunakan N yaitu jumlah dokumen.

4. Perhitungan pada TF Grab

Pada kalimat dokumen Grab yang telah dimasukan ke dalam perhitungan TF bisa dilihat pada Tabel 3.16, dan untuk penjelasan lengkapnya dapat dilihat pada lampiran 2.

Tabel 3.16 Perhitungan pada TF Grab

***Term* (kata)**	**d1**	**d2**	**d3**	**d4**	**df**
gak	1			1	2
pernah	1				1

Pada Tabel 3.16 memberikan penjelasan terhadap kemunculan term atau kata didalam sebuah kalimat yang telah dimasukan. Pada perhitngan IDF ini akan menggunakan beberapa komponen seperti term atau kata, serta tf dan idf yang terhubung dari kesedian suatu term dari semua dokumen, dihitung menggunakan N yaitu jumlah dokumen, dan berikut ini adalah contoh perhitungan IDF yang dapat dilihat pada Tabel dibawah.

5. Perhitungan Pada IDF Gojek

Contoh perhitungan IDF pada dokumen Gojek dapat dilihat pada Tabel 3.17 dan penjelasan lengkapnya dapat dilihat pada lampiran 3.

Tabel 3.17 Perhitungan Pada IDF Gojek

***Term* (kata)**	**df**	**idf**	**idf(N=4**	**Idf(N=100**
admin	1	0,602	0,602059991	3
kok	1	0,602	0,602059991	3
akun	2	0,301	0,301029996	2,698970004

Pada Tabel 3.17 memperlihatkan perhitungan IDF secara manual dengan rumus pada persamaan (1).

6. Perhitungan Pada IDF Grab

Contoh perhitungan IDF pada dokumen Grab dapat dilihat pada Tabel 3.18 untuk penjelasan lengkapnya dapat dilihat pada lampiran 4.

Tabel 3.18 Perhitungan Pada IDF Grab

***Term* (kata)**	**df**	**idf**	**idf(N=4**	**Idf(N=100**
gak	2	0,176	0,301029996	2,698970004
pernah	1	0,477	0,602059991	3

Pada Tabel 3.18 memperlihatkan perhitungan IDF secara manual dengan rumus pada persamaan (1), sedangkan untuk penjelasan perhitungan TF-IDF dapat dilihat pada dibawah.

7. Perhitungan TF-IDF Gojek

Perhitungan dengan jumlah 4 dokumen pada Gojek dapat dilihat pada Tabel 3.19 dan penjelasan lengkapnya dapat dilihat pada lampiran 5.

Tabel 3.19 Perhitungan Pada TF-IDF Gojek

***Term* (kata)**	**d1**	**d2**	**d3**	**d4**
akun	0,033444444			0,012542
gak	0,033444444			0,012542

Pada Tabel 3.19 menjelaskan perhitungan manual pada TF-IDF secara manual menggunakan Microsoft Office Excel awal dari hasil perkalian tf dan idf.

8. Perhitungan TF-IDF Grab

Perhitungan dengan jumlah 5 dokumen pada Grab dapat dilihat pada Tabel 3.20 untuk detail penjelasan lengkapnya dapat dilihat pada lampiran 6.

Tabel 3.20 Perhitungan Pada TF-IDF Grab

***Term* (kata)**	**d1**	**d2**	**d3**	**d4**
gak	0,019536			
pernah	0,052947			

Pada Tabel 3.20 menjelaskan perhitungan manual pada TF-IDF secara manual menggunakan Microsoft Office Excel awal dari hasil perkalian tf dan idf. Klasifikasi pada penelitian ini menggunakan fitur ekstraksi TF-IDF yang memberikan hasil perhitungan secara otomatis pada pembobotan kata pada setiap dokumen. *Library* yang digunakan untuk perhitungan TF-IDF ini adalah *sklearn.feature_extraction*.text serta TfidfVectorizer untuk melakukan perhitungan secara otomatis. Dibantu dengan *library* Multinomial Naïve Bayes dimana akan membantu untuk mengklasifikasi teks pada sebuah data *training*. Dan berikut adalah kode untuk perhitungan TF-IDF didalam sebuah sistem pada kode-kode dibawah ini.

```
from sklearn.feature_extraction.text import TfidfVectorizer

s1 = "jam pagi sekarang x drivernya cancel pickup paket ada
sistem punishment buat driver yang cancel memang"
s2 = "ya min akun tidak bisa pesan antar menu gofood kendala
udah kemarin loh"

Vect = TfidfVectorizer()
# X = TfidfVectorizer().fit['s1', 's2']
X = Vect.fit_transform(['s1', 's2'])

X.toarray()
```

1. Perhitungan Otomatis Data Gojek

Perhitungan otomatis pada data Gojek dengan dua kali perhitungan, dapat dilihat pada Gambar 3.4 dan Gambar 3.5.

```
[(0.242535625036333, 'ada'),
 (0.0, 'akun'),
 (0.0, 'antar'),
 (0.0, 'bisa'),
 (0.242535625036333, 'buat'),
 (0.485071250072666, 'cancel'),
 (0.242535625036333, 'driver'),
 (0.242535625036333, 'drivernya'),
 (0.0, 'gofood'),
 (0.242535625036333, 'jam'),
 (0.0, 'kemarin'),
 (0.0, 'kendala'),
 (0.0, 'loh'),
 (0.242535625036333, 'memang'),
 (0.0, 'menu'),
 (0.0, 'min'),
 (0.242535625036333, 'pagi'),
 (0.242535625036333, 'paket'),
 (0.0, 'pesan'),
 (0.242535625036333, 'pickup'),
 (0.242535625036333, 'punishment'),
 (0.242535625036333, 'sekarang'),
 (0.242535625036333, 'sistem'),
 (0.0, 'tidak'),
 (0.0, 'udah'),
 (0.0, 'ya'),
 (0.242535625036333, 'yang')]
```

Gambar 3.4 Hasil Perhitungan Pertama TF-IDF Gojek

```
[(0.0, 'ada'),
 (0.27735009811261463, 'akun'),
 (0.27735009811261463, 'antar'),
 (0.27735009811261463, 'bisa'),
 (0.0, 'buat'),
 (0.0, 'cancel'),
 (0.0, 'driver'),
 (0.0, 'drivernya'),
 (0.27735009811261463, 'gofood'),
 (0.0, 'jam'),
 (0.27735009811261463, 'kemarin'),
 (0.27735009811261463, 'kendala'),
 (0.27735009811261463, 'loh'),
 (0.0, 'memang'),
 (0.27735009811261463, 'menu'),
 (0.27735009811261463, 'min'),
 (0.0, 'pagi'),
 (0.0, 'paket'),
 (0.27735009811261463, 'pesan'),
 (0.0, 'pickup'),
 (0.0, 'punishment'),
 (0.0, 'sekarang'),
 (0.0, 'sistem'),
 (0.27735009811261463, 'tidak'),
 (0.27735009811261463, 'udah'),
 (0.27735009811261463, 'ya'),
 (0.0, 'yang')]
```

Gambar 3.5 Hasil Perhitungan Kedua TF-IDF Gojek

2. Perhitungan Otomatis Data Grab

Perhitungan otomatis pada data Grab dengan dua kali perhitungan, dapat dilihat pada Gambar 3.6 dan Gambar 3.7.

```
[(0.447213595499958, 'baikk'),
 (0.0, 'boleh'),
 (0.0, 'dapat'),
 (0.0, 'grab'),
 (0.447213595499958, 'grabfood'),
 (0.0, 'grabpay'),
 (0.447213595499958, 'lebih'),
 (0.447213595499958, 'nya'),
 (0.0, 'pointslepas'),
 (0.0, 'redeem'),
 (0.447213595499958, 'tolong'),
 (0.0, 'voucher')]
```

Gambar 3.6 Hasil Perhitungan Pertama TF-IDF Grab

```
[(0.0, 'baikk'),
 (0.3779644730092272, 'boleh'),
 (0.3779644730092272, 'dapat'),
 (0.3779644730092272, 'grab'),
 (0.0, 'grabfood'),
 (0.3779644730092272, 'grabpay'),
 (0.0, 'lebih'),
 (0.0, 'nya'),
 (0.3779644730092272, 'pointslepas'),
 (0.3779644730092272, 'redeem'),
 (0.0, 'tolong'),
 (0.3779644730092272, 'voucher')]
```

Gambar 3.7 Hasil Perhitungan Kedua TF-IDF Grab

Setelah dilakukan perhitungan TF-IDF secara otomatis oleh sistem selanjutnya akan dilakukan pencarian akurasi dari data *training* yang sebelumnya sudah dilakukan pelabelan manual tujuannya adalah untuk mengetahui keakuratan dari sebuah dokumen atau data tersebut. Kode yang digunakan untuk mencari nilai dari akurasi pada data *training* dapat dilihat pada kode dibawah ini.

```
from sklearn.metrics import accuracy_score,
f1_score, confusion_matrix
print("Accuracy: {:.2f}%".format(accuracy_score
(y_test, y_pred) * 100))
print("\nF1 Score: {:.2f}".format(f1_score
(y_test, y_pred, average='weighted') * 100))
print("\nCOnfusion Matrix:\n", confusion_matrix
(y_test, y_pred))
```

Cross-validation adalah metode untuk mendapatkan hasil akurasi yang dihitung sampai beberapa kali dengan parameter yang sama. Untuk melakukan pengujian akurasi metode Naive Bayes agar dapat diketahui akurasinya. Konsep pada *cross-validation* ini adalah membagi dua data yaitu data latih dan data uji Dengan memakai *library* from *sklearn*.model_selection serta import *ShuffleSplit* untuk melakukan perhitungan rata-rata dalam 10 kali. Berikut kode untuk menghitung *cross validation* dapat dilihat pada kode dibawah ini.

```
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True,
 figsize=(16,9))
acc_scores = [round(a * 100, 1) for a in accs]
f1_scores = [round(f * 100, 2) for f in f1s]
x1 = np.arange(len(acc_scores))
x2 = np.arange(len(f1_scores))

ax1.bar(x1, acc_scores)
ax2.bar(x2, f1_scores, color='#559ebf')

# Place values on top of bars
for i, v in enumerate (lis(zip(acc_scores, f1_scores))):
```

```
        ax1.text(i - 0.25, v[0] + 2, str(v[0]) + '%')
        ax2.text(i - 0.25, v[1] + 2, str(v[1]))

    ax1.set_ylabel('Accuracy (%)')
    ax1.set_title('Naive Bayes')
    ax1.set_ylim([0, 100])

    ax2.set_ylabel('F1 Score')
    ax2.set_xlabel('Runs')
    ax2.set_ylim([0, 100])
    sns.despine(bottom=True, left=True)
     # Remove the ticks on axes for cleaner presentation
    plt.show()
```

Tahap selanjutnya adalah langkah untuk melakukan perhitungan keakuratan pemodelan pada tahapan *training* yang akan digunakan untuk memprediksi label (kelas) dari data yang sudah tersedia. Dilanjutkan dengan melakukan pembuatan model klasifikasi dengan variabel X serta variabel Y dengan data *training* yang sudah diproses sebelumnya. Model tersebut dibuatkan sebuah fungsi agar pada saat proses pemanggilannya lebih mudah saat dijalankan untuk tahap berikutnya, sehingga akan lebih efektif dan efisien.

Didalam proses pembentukan model klasifikasi digunakan *library sklearn*.pipeline dengan import pipeline yang fungsinya testable pada proses cross-validation, lalu import pickle memiliki fungsi untuk menyimpan serta membaca file berformat .pkl. Dan berikut kode untuk import *library* pembuatan model klasifikasi terdapat pada kode dibawah ini.

```
import os
import pickle
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text
import TfidfTransformer
```

Dan untuk pembuatan model klasifikasi dapat dilihat pada kode dibawah ini.

```
X = df.content
y = df.Sentimen
text_classifier=Pipeline([('vect', TfidfVectorizer()),
('tfidf', TfidfTransformer()),
('classifier', MultinomialNB(alpha=1.0)),])
X_train = np.asarray(X)
text_classifier=text_classifier.fit
(X_train, np.asarray(y))
```

Model yang diberikan nama text_classifier akan tersimpan ke dalam bentuk file.pickle sehingga nantinya dapat dibuka serta digunakan kembali. Berikut kode untuk menyimpan file pickle serta membuka file pickle dapat dilihat pada kode-kode dibawah ini.

```
files = open('model_classifier_nbc2.pickle', 'wb')
pickle.dump(text_classifier, files)
files.close()
model = open('model_classifier_nbc2.pickle', 'rb')
nbc_classifier = pickle.load(model)
print(nbc_classifier)
```

Bentuk file pickle yang sudah dibentuk model klasifikasi selanjutnya akan dipergunakan untuk menjalankan data *testing* dengan jumlah 400 *tweet* pada Gojek dan Grab yang sudah dilabeli secara manual dari jumlah data *training* yang berjumlah 1134 *tweet* pada Gojek serta Grab dari total data 8000 *tweet* untuk Gojek dan Grab sebanyak 4000 data yang sudah dilakukan filtering data di Microsoft Office Excel. Dan berikut kode untuk pemanggilan dataframe terdapat pada kode dibawah ini.

```
df_tweet = pd.read_excel (r'topicgojek.xlsx',
sheet_name='Sheet1')
df_tweet=pd.DataFrame(df_tweet)
df_tweet=df_tweet.dropna()
df_tweet.head()
```

Setelah melakukan pemanggilan dataframe maka langkah selanjutnya adalah melakukan prediksi dari metode Naïve Bayes. Kode pemanggilan untuk hasil prediksi dari Naïve Bayes dapat dilihat pada kode dibawah ini.

```
predicted=nbc_classifier.predict(np.asarray(data_tweet))
```

Lalu selanjutnya hasil prediksi yang sudah didapatkan dari Naïve Bayes akan disimpan didalam variabel result_*tweet* dalam sebuah bentuk data list. Kode untuk pemanggilan hasil prediksi dari Naïve Bayes terdapat pada kode dibawah ini.

```
result_tweet=[]
for i in range(len(predicted)):
    if(predicted[i]=='positif'):
        sentiment_result='positif'
#elif(predicted[i]=='netral'):
#sentiment_result='netral'
    elif(predicted[i]=='negatif'):
        sentiment_result='negatif'
#result_tweet.append
({'class':prediction_linear[i],'result_nbc':
sentiment_result})result_tweet.append
({'tweet':df_tweet['tweet'][i],'topic':df_tweet
['topic'][i],'class':predicted[i],'result_nbc'
:sentiment_result})
```

3.2.8 *Testing*

Pada tahapan *testing* ini untuk menentukan akurasi dari model yang telah dibuat pada tahapan *training*, bertujuan untuk menentukan label atau kelas dari data *testing* yang telah disediakan. Maka akan ditampilkan data *testing* dari masing-masing *platform* yang dilabeli secara manual (actual) dan dari metode Naïve Bayes (predicted) terdapat pada Gambar dibawah ini.

1. *Data Testing* Gojek

Untuk menentukan akurasi dari tahapan pada *training* maka dilakukan proses data *testing* pada Gojek dan hasilnya dapat dilihat pada Gambar 3.8.

	text	Score	Sentimen
0	wahahaha gua pernah dapet ni ambas satpol mera...	-0.4588	negatif
1	top up aja cepet gilir mau trafer gopay no rek...	-0.1027	negatif
2	baru hari kirim barang aplikasi bayar driver b...	-0.1027	negatif
3	berita lagi rame hoax kak	-0.2732	negatif
4	sama kayak td sore jalan batu tumbuh jati cemp...	-0.3612	negatif
...	...	...	...
1128	haha jam jam gakk order padahal cuma minus rib...	0.4588	positif
1129	jadi tuh wegotyou banget punya fitur perjalana...	0.2263	positif
1130	min cek dm ya thanks	0.4404	positif
1131	sering lihat motor gojek bawa barang super bes...	0.5994	positif
1132	hai kak aku saranin kamu untuk konfirmasi lebi...	0.4019	positif

1133 rows × 3 columns

Gambar 3.8 *Data Testing* Gojek

2. *Data Testing* Grab

Untuk menentukan akurasi dari tahapan pada *training* maka dilakukan proses data *testing* pada Grab dan hasilnya dapat dilihat pada Gambar 3.9.

	text	Score	Sentimen
0	reckless nya driver i jarang bad rating grab d...	-0.4767	negatif
1	almost babe hujan stop je i terus blkg and jal...	-0.2350	negatif
2	mumpung diskon anjrittt tengah harga lebih bah...	-0.4215	negatif
3	pakee saldo sbux cardnya kak sistem sama kek g...	-0.1027	negatif
4	hai kak herri buat minta aju yang kakak sampai...	-0.2960	negatif
...	...	...	...
1128	iyanihh bener gua punya alam mau dispill kasi ...	0.2263	positif
1129	nah dialihin biar happy pesan makan bareng gra...	0.5106	positif
1130	wah nungguin promo grab disney hotstar masuk a...	0.4019	positif
1131	hi moots lup pacar jual keychain rv hehe siapa...	0.6486	positif
1132	pinter cara rayu sayaetannnn gosah sok knalll ...	0.3182	positif

1133 rows × 3 columns

Gambar 3.9 *Data Testing* Grab

Selanjutnya akan dilakukan pencarian nilai perbandingan model dengan *confusion matrix* terdapat *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *True Negative* (TN). Yang nantinya hasil dari prediksi perbandingan tersebut akan

dilakukan perhitungan menggunakan Microsoft Office Excel sehingga akan mendapatkan nilai akurasi dari data *testing* tersebut.

3.2.9 Klasifikasi

Setelah didapatkan nilai akurasi yang baik dari proses *training* dan proses *testing* selanjutnya dilakukan proses klasifikasi data dari dua *platform* dari proses klasifikasi ini data yang digunakan adalah data dari keseluruhan dengan jumlah 8000 data untuk Gojek dan 6000 data untuk Grab yang sudah dilakukan filtering di Microsoft Office Excel menjadi 8000 untuk Gojek dan 4000 data untuk Grab dengan data bersih yang digunakan, data tersebut sebelumnya sudah diproses kedalam classification metode Latent Dirichlet Allocation sehingga mendapatkan data yang sudah terdapat penyebaran topiknya serta nilai scorenya. Selanjutnya didapat nilai akurasi yang baik dari proses analisis sentimen dan proses *testing* dengan menggunakan 2 tahapan yaitu pertama dengan metode (LDA) dan tahap kedua dengan (NBC), sehingga akan menghasilkan nilai sentimen perlayanan dari masing masing *platform* ojek *online*, untuk menjadi bahan perbandingan antara dua *platform* tersebut mana yang lebih baik.