BAB 3
METODE PENELITIAN

Penelitian ini adalah penelitian analisis sentimen positif dan negatif pada data Twitter. Penelitian ini menggunakan metode *Lexicon Based* dan *Naïve Bayes Classifier*. Penelitian ini membutuhkan data *tweet* yang didapatkan dari Twitter terkait Jaminan Hari Tua (JHT) BPJS Ketenagakerjaan yang selanjutnya akan dilakukan pengolahan data berupa *pre-processing* untuk mendapatkan hasil yang diinginkan. Data tersebut nantinya digunakan untuk memetakan informasi atau sentimen dari masyarakat di Twitter terkait Jaminan Hari Tua (JHT) BPJS Ketenagakerjaan sehingga didapatkan informasi yang sesuai.

Penelitian ini berawal dari latar belakang permasalahan yang ada, mengolah data yang sudah didapatkan dan menentukan sentimen yang tepat sehingga informasi yang diperoleh sesuai dengan apa yang diharapkan. Berikut ini adalah bahan, alat dan jalannya penelitian analisis sentimen BPJS Ketenagakerjaan mengenai jaminan hari (JHT) serta tahapan penelitian guna menyelesaikan proses analisis sentimen menggunakan data *tweet*.

3.1 BAHAN DAN ALAT PENELITIAN

Bahan penelitian ini akan menggali data dan informasi dari *tweet* yang ada di dalam *website* Twitter terkait Jaminan Hari Tua (JHT) BPJS Ketenagakerjaan menggunakan kata kunci bpjstk, bpjs dan jht.

Alat yang digunakan dalam penelitian ini adalah komputer dengan spesifikasi cukup untuk menjalankan sistem operasi dan *software* pengembangan serta koneksitas Internet. Sistem Operasi dan program-program aplikasi yang dipergunakan dalam pengembangan aplikasi ini adalah:

1. Sistem Operasi: Windows 10.
2. Anaconda versi 3.
3. Jupyter Notebook.
4. *Microsoft Office Excel* 2019.
5. Bahasa Pemrograman Python 3.7.4

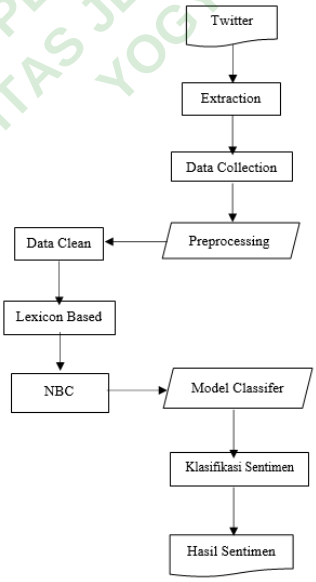
6. Sublime Text.

3.2 JALAN PENELITIAN

Jalan penelitian ini adalah penelitian analisis sentimen positif dan negatif pada data Twitter. Penelitian ini menggunakan metode *Lexicon Based* dan *Naïve Bayes Classification*. Penelitian ini membutuhkan data *tweet* yang didapatkan dari Twitter yang berkaitan tentang BPJS Ketenagakerjaan mengenai Jaminan Hari Tua (JHT), selanjutnya dilakukan pengolahan data berupa *pre-processing* untuk mendapatkan hasil yang diinginkan.

Penelitian ini menggunakan bahasa pemrograman Python, dan dimodelkan dengan bantuan *library* pada bahasa pemrograman Python, Anaconda 3 dan Jupyter Notebook untuk melakukan pengambilan data yang akan di tampilkan di Microsoft Office Excel dan akan dilakukan *Pre-processing* data, serta menganalisi nilai-nilai sentimen pertopik secara otomatis melalui *website dashboard* yang akan dibuat menggunakan bahasa pemrograman Python.

Adapun langkah-langkah pada metode penelitian tersebut. Alur penelitian dapat dilihat pada gambar 3.1.



Gambar 3. 1 Alur Penelitian

3.2.1 Extraction

Extraction merupakan sebuah tahapan mengambil data dari Twitter mengenai Jaminan Hari Tua (JHT) BPJS Ketenagakerjaan dengan menggunakan kode Python dan diproses di Anaconda Promt dan menghasilkan data yang berupa file csv. Kode untuk melakukan *extraction* dapat dilihat dibawah ini.

```
import snscrape.modules.twitter as sntwitter
import pandas as pd
from time import sleep

tweet_data = []

username = input('Enter your keyword: ')
number = int(input('How many tweets do you to scrape: '))

for i,tweet in enumerate(sntwitter.TwitterSearchScraper('jaminan
hari tua since:2022-02-14 until:2022-07-12 lang:id').get_items()):
      if i > number:
            break
      tweet_data.append([tweet.date,tweet.content,tweet.username)

df=pd.DataFrame(tweet_data,columns=['Date','Content','Username'])
df.to_csv(f'{username}.csv', index=False, encoding='utf-8')
```

Data dari hasil *exraction* tersebut dapat dilihat pada tabel 3.1

Tabel 3. 1 Data Tweet

NO	Content
1	@BPJSTKinfo Lalu jika bpjstk nya dilanjut lg setelah diambil uangnya kemudian dgn nomer yg sama ketika sudah bekerja kembali apakah bisa?
2	Bayar BPJS pakai autodebit BRI bulan ini (juli) terjadi dua kali transaksi. Waktu dan nomor transaksi sama persis begitu pula dengan autofee bpjs nya. Kok bisa? @BPJSKesehatanRI @bpjskesehatan3 @BPJSTK_Mobile@ BANKBRI_ID a thread
3	@BPJSTKinfo halo min, mau tanya kenapa setelah pencairan dana bpjstk dan masuk ke perusahaan baru no kpj sudah tidak dapat digunakan kembali? Cek DM.
4	@barcastuff_idn Gpp, porotin aja psg, cari dana buat jaminan hari tua ðŸ˜
5	Selain itu secara otomatis, saldo Jaminan Hari Tua (JHT) yang dimiliki oleh peserta juga turut dibayarkan, serta manfaat JHT

3.2.2 Data Collection

Data collection merupakan data yang sudah terkumpul dari beberapa kata kunci yang dimasukan yaitu BPJSTK, BPJS dan JHT, yang masing-masing kata kunci menghasilkan 10.000 data, dan data tersebut digabung dan menghasilkan 30.000 data yang nantinya akan masuk ke tahap *pre-processing*.

3.2.3 Data Pre-Processing

Data *pre-processing* adalah teknik *data mining* yang melibatkan transformasi data mentah menjadi format yang mudah dimengerti. Sebelum memproses data masukan *library* yang akan digunakan, *library* untuk melakukan *data pre-processing* dapat dilihat dibawah ini.

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

Adapun langkah-langkah dalam data pre-processing, yaitu :

1. Cleaning

Melakukan pembersihan atribut-atribut yang tidak penting yang terdapat di dalam dokumen yang sudah di dapatkan, seperti hastag, angka, emoji dan url. Kode untuk melakukan *cleaning* dapat dilihat dibawah ini.

```
def url_remove(tweet):
    t = re.sub(r'http\S+', '',tweet)
    return t

def punc_remove(tweet):
    t = re.sub(r'[^\w\s]','',tweet)
    return t

def rt_remove(tweet):
    t = re.sub(r'RT[\s]+', '', tweet)
    return t

def number_remove(tweet):
    t = re.sub('[0-9]+', '', tweet)
    return t

def slang_remove(tweet):
    t = re.sub(r'\\n', " ", tweet)
    return t
def regex_remove(tweet):
```

```
    t = re.sub("b'", " ", tweet)
    return t

def remove_user(tweet):
    t = re.sub('@[^\s]+','',tweet)
    return t

def emoji_remove(tweet):
    return emoji.get_emoji_regexp().sub("", tweet).strip()

def hashtag_remove(tweet):
    reg = "#(\w+:\/\/S+)"
    return re.sub(reg, " ", tweet)

cleaned = []

def clean_text(tweet):
    for i in tweet:

cleaned.append(url_remove(punc_remove(number_remove(remove_user(r
egex_remove(hashtag_remove(rt_remove(emoji_remove(slang_remove(
            re.sub("[\n\r\t\xa0]"," ",i).strip()))))))))))
clean_text(data["Content"])
```

2. Case Folding

Melakukan konversi dari bentuk awal menjadi bentuk standar (huruf kecil atau *lowercase*). Kode untuk melakukan c*ase folding* dapat dilihat dibawah ini.

```
def lowercase():
    lower_word = data['Content'].str.lower()
    return lower_word

lower_tweet = lowercase()

print(lower_tweet)
```

3. Tokenizing

Melakukan pemecahan atau pemisahan karakter dalam suatu teks yang didefinisikan sebagai pemisah kata atau bukan. Kode untuk melakukan *tokenizing* dapat dilihat di bawah ini.

```
def clean_text(tweet):
    for i in tweet:

cleaned.append(url_remove(punc_remove(number_remove(remove_user(r
egex_remove(hashtag_remove(rt_remove(emoji_remove(slang_remove(
            re.sub("[\n\r\t\xa0]"," ",i).strip()))))))))))
```

```
clean_text(data["Content"])
```

4. Stopword removal

Melakukan penghapusan kata-kata yang memiliki informasi rendah dari sebuah teks (“yang”, “dan”, “di”, “dari” dll). Kode untuk melakukan *Stopword Removal* dapat dilihat di bawah ini.

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory

factory = StopWordRemoverFactory()
more_stopword =

stopword = factory.create_stop_word_remover()
stopwords = factory.get_stop_words()+more_stopword
print(stopwords)
```

Contoh kata *stopword* dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Stopword

Stopword
['anjing','anjir','pantek','yg','utk','cuman','deh','id','nih', 'ecusli','dpt','xixixixi','hahaha','dr','kpn','kok','kyk','donk','eh','sih', 'eh','bang','br','kyk','rp','jt','kan','gpp','usah','sob','thx','ato','je','gu','ukuk','mak','haha','haha','duh','ye','wkwkwk','syg','btw','nerjemahin','gaes','guys','moga','nemu','yukkk','wkwkw','klas','iw','ew','lho','bwt','clau', 'cpet','ku','wke','mba','mas','oi','sh','wakakaka','sihhh','hehe','ih','la','mana', 'knna','tuh','dah','kek','ko','pls','mah','dhhh','tuh','kzl','si','sii','cm', 'hahahaha','weh','kntl','gblk','dlu','tuhh','bgst','ajg','pantat','mw','lw','jooyul','kidr','larka','jst','khara','camery','ky','agy','arha','or','apny','thapr','mar', 'kam','bolo','bibi','or','mafi','mango','lollll','lol','cmiiw','fyi','Anw','kr','easier', 'hueee','jeti','kuuu','care','dede','thread','cutoff','nada','dansa','tks','dear','wehhh', 'cw','wrk','guis','nett','umbi','mjb','yawww','fyi','geh','fck','hhh','fuck','brengsek', 'bajingan','kontol','bangke','tkstedi','act','ombudsman','bit','CC','noh','Juaanccokkkkkk', 'lahalaaaahhhh','nder','njirrr','njir','bro','oiya','ar','kunyuk','nmdi','momyettttt','alahhhhh', 'waduuuh','hyunsuk','seh','iwan','goyang','wkwk','xxxx','askrl','ecek','spn','jsh','kanwil',

5. Stemming

Proses untuk merubah semua kata menjadi kata dasar. Kode untuk melakukan *stemming* dapat dilihat di bawah ini.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()
stemmer = factory.create_stemmer()

def stemmed_wrapper(term):
    return stemmer.stem(term)

term_dict = {}

for document in stopwords_tweet:
    for term in document:
        if term not in term_dict:
            term_dict[term] = " "

print(len(term_dict))
print("------------------------")

for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
    print(term,":" ,term_dict[term])

print(term_dict)
print("------------------------")

def get_stemmed_term(document):
    return [term_dict[term] for term in document]

stem_tweet = stopwords_tweet.apply(get_stemmed_term)

print(stem_tweet)
```

6. Normalization

Proses menormalkan kata-kata yang disingkat seperti kata “yg” di ubah menjadi “yang”, kata “mnjd” menjadi kata “menjadi”. Kode untuk melakukan *normalization* dapat dilihat dibawah ini.

```
normalizad_word = pd.read_excel("normalization.xlsx")

normalizad_word_dict = {}

for index, row in normalizad_word.iterrows():
    if row[0] not in normalizad_word_dict:
        normalizad_word_dict[row[0]] = row[1]
```

```
def normalized_term(document):
    return     [normalizad_word_dict[term]     if     term     in
normalizad_word_dict else term for term in document]

normal_tweet = stem_tweet.apply(normalized_term).str.join(" ")

print(normal_tweet)
```

Contoh tabel *normalization* dapat dilihat pada tabel 3.3.

Tabel 3. 3 Normalization

Before	After
yg	Yang
mnjd	Menjadi
gak	Tidak
pake	Pakai
utk	Untuk

7. Data Clean

Merupakan data yang benar-benar sudah bersih dan bisa digunakan untuk tahap berikutnya. Kode untuk menyimpan data bersih dalam bentuk csv dapat dilihat di bawah ini.

```
tt={"Date":data['Date'],"Content":data['Content'],
"Username":data['Username'] ,"Cleaning_Text":data_preprocess}

df=pd.DataFrame(tt)
df.to_csv("data.csv")
```

contoh data bersih dapat dilihat pada tabel 3.4.

Tabel 3. 4 Data Bersih

NO	Cleaning_Text
1	gus dur dagang rakyat kalau bumn rugi rakyat perintah kompeni usaha ikat nama negara rakyat beli sehat bayar bpjs modus asuransi
2	protes subsidi bpjs diilangin saya kategori warga subsidi cmn ngelus dada saja anggap cmn mikir saja harap subsidi beneran tidak
3	rakyat elus dada berfikir cari solusi hadap naik bahan pokok pmk listrik bbm bpjs anak sekolah kuliah kuat sabar problematika
4	Iya soalnya saya udah di phk dari mei lalu udah claim JHT juga pas Juni udah buat akun siapkerja juga tapi pas mau ajukan JKP di sso bpjs malah tidak bisa
5	jht gak bisa langsung cairsedangkan BPJS gak boleh nunggak org nganggur disuruh berfikir buat bayar bpjssedangkan buat makan sehari aja susah negara dikelola oleh keparat bermental rakus

3.2.4 Pelabelan

Setelah tahap *pre-processing*, maka tahap selanjutnya adalah proses pelabelan data untuk mengetahui yang mana saja tanggapan positif dan negatif. Pelabelan manual adalah proses memberikan label terhadap kata atau kalimat yang ada pada dokumen sehingga dapat dianalisis lebih lanjut mengenai sifatnya yang positif atau negatif. Contoh data pelabelan manual dapat dilihat pada Tabel

Lexicon based ini dapat digunakan pada proses pelabelan data atau tanggapan dengan cara menghitung setiap skor sentiment untuk mempermudah dalam proses klasifikasi. Proses menghitung setiap kata berdasarkan kamus positif serta negatif dalam suatu kalimat tersebut, yaitu dengan menjumlahkan nilai opini. Jumlah nilai opini buat sentimen positif yaitu nilai 1 ataupun lebih dan untuk sentimen negatif bernilai -1 ataupun lebih. (Prianto, C., Rahayua, W.I., & Hamka, N.I., 2020). Contoh data pelabelan manual dapat dilihat pada Tabel 3.5.

Tabel 3. 5 Data Pelabelan Manual

No	Content	Label
1	mincottt banget lagi kendala cairin bpjstk jaminan hari tua usaha banget resign tapi gpunya paklaring perushaan th pas usaha blg tidak bisa paklaring karena tidak tahun krja bulan bagaimana	Negatif
2	gapapa tidak dapat bonus tidak dapat bpjs ketenagakerjaan admin	Positif
3	treasure keroyok kaya gini untung bpjs	Positif
4	maju kamu treasure tidak takut bpjs	Positif
5	kasihan teman tidak panjang kontrak paklaringnya tidak kasih alas banget butuh cair bpjstk pegang smp kerja alhamdulillah skr barutp paklaringnya belum jugamuke gile memang itu kantor	Negatif

Contoh pelabelan otomatis dapat dilihat pada Tabel 3.6.

Tabel 3. 6 Data Pelabelan Otomatis

No	**Content**	**Label**	**Score**	**Sentimen**
1	mincottt banget lagi kendala cairin bpjstk jaminan hari tua usaha banget resign tapi gpunya paklaring perushaan th pas usaha blg tidak bisa paklaring karena tidak tahun krja bulan bagaimana	Negatif	-0.34	Negatif
2	gapapa tidak dapat bonus tidak dapat bpjs ketenagakerjaan admin	Positif	0.5423	Positif
3	treasure keroyok kaya gini untung bpjs	Positif	0.296	Positif
4	maju kamu treasure tidak takut bpjs	Positif		Positif
5	kasihan teman tidak panjang kontrak paklaringnya tidak kasih alas banget butuh cair bpjstk pegang smp kerja alhamdulillah skr barutp paklaringnya belum	Negatif	-0.2732	Negatif

	jugamuke gile memang itu kantor			

3.2.5 Proses Lexicon Based

Setelah semua tahapan *pre-processing* selesai dilakukan, tahap selanjutnya yaitu mengambil data bersih sebanyak 600 data yang siap untuk dilakukan tahap *testing* sentimen menggunakan kamus *Lexicon based*. Tahapan ini memegang peranan penting dalam klasifikasi. Karena penelitian ini menggunakan pendekatan pada *word level*, dimana data yang diproses adalah kata untuk memperoleh skor sentimen.

Kamus *Lexicon* yang digunakan pada penelitian ini adalah kamus *InSet Lexicon* dari penelitian sebelumnya yang dilakukan untuk mengklasifikasi sentimen terhadap data Twitter. Kamus *InSet Lexicon* ini berisi daftar kata yang mengandung sentimen positif maupun negatif serta sudah memiliki bobot nilai untuk kata nya.

Kamus *lexicon* ini terdiri dari 3609 kata positif dan 6609 kata negatif. Bobot pada kamus *lexicon* ini memiliki nilai dengan rentang skor dari -5 sampai +5. Pada penelitian ini dilakukan beberapa penambahan kata yang berkaitan dengan topik BPJS Ketenagakerjaan. (Prasetya, Y. N., Winarso. D., & Syahril, 2021)

Install VaderSentiment dan *Import library* yang di gunakan untuk *pre-procesing* data. Kode *library Lexicon Based* dapat dilihat di bawah ini.

```
pip install VaderSentiment
```

Kalimat yang sudah dilakukan pelabelan manual, yang nantinya akan di bandingan dengan kalimat yang dilabeli oleh mesin, yang nantinya akan di hitung akurasinya.

Polarity score berfungsi untuk menampilkan *score* atau bobot dari setiap kalimat yang terdeteksi oleh mesin, dimana jika kalimat tersebut memiliki nilai atau score dibawah 0 maka kalimat tersebut terlabeli negatif begitupun sebaliknya, apabila kalimat tersebut memiliki nilai atau score di atas 0 maka kalimat tersebut terlabeli positif. Kode untuk *Polarity Score* dapat dilihat di bawah ini.

```
nilai = [analyser.polarity_scores(x) for x in df['content']]
print(nilai)
df['Score'] = [x['compound'] for x in nilai]
```

Data yang sudah mendapatkan *polarity score* dapat dilihat pada Tabel 3.7.

Tabel 3. 7 Polarity Score

No	Content	Label	Score
1	mincottt banget lagi kendala cairin bpjstk jaminan hari tua usaha banget resign tapi gpunya paklaring perushaan th pas usaha blg tidak bisa paklaring karena tidak tahun krja bulan bagaimana	Negatif	-0.34
2	gapapa tidak dapat bonus tidak dapat bpjs ketenagakerjaan admin	Positif	0.5423
3	treasure keroyok kaya gini untung bpjs	Positif	0.296
4	maju kamu treasure tidak takut bpjs	Positif	
5	kasihan teman tidak panjang kontrak paklaringnya tidak kasih alas banget butuh cair bpjstk pegang smp kerja alhamdulillah skr barutp paklaringnya belum jugamuke gile memang itu kantor	Negatif	-0.2732

Berilutnya menentukan prediksi sentimen berdasarkan nilai *score* nya, apabila nilai *score* nya negatif maka hasil sentimen juga negatif dan apabila nila *score* nya positif maka hasil sentimen positif. Kode untuk melakukan prediksi sentimen dapat dilihat dibawah ini.

```
df.loc[df['Score'] < 0, 'Sentimen'] = 'negatif'

df.loc[df['Score'] > 0, 'Sentimen'] = 'positif'

df.nsmallest(600, ['Score'])
```

Setelah mendapatkan pelabelan otomatis langkah berikutnya adalah menghitung nilai *accuracy, precision, recal* dan *f1-score* menggunakan *concfusion matrix* dari data tersebut. Proses *confusion matrix* dapat dilihat pada Tabel 3.8.

Tabel 3. 8 Proses Confusion Matrix

	Positif	**Negatif**
Positif	TP	FP
Negatif	FN	TN

Kode untuk menghitung nilai *accuracy* dapat dilihat di bawah ini.

```
accuracy = (TP+TN)/(TP+TN+FP+FN)
print('Accuracy =', accuracy)
```

Kode untuk menghitunga nilai *precission* dapat dilihat di bawah ini.

```
precission = (TP) / (TP+FP)
print('Precission =', precission)
```

Kode untuk menghitung nilai *recall* dapat dilihat di bawah ini.

```
recall = (TP) / (TP + FN)
print('Recall =', recall)
```

Kode unutk menghitung nilai *F1-Score* dapat dilihat di bawah ini.

```
F1_Score = 2 * (recall*precission) / (recall + precission)
print('F1Score =', F1_Score)
```

3.2.6 Proses Naïve Bayes Classifer

1. Training Data

Training data adalah proses *training* pada data dengan menggunakan metode *Naive Bayes Classification*. Tahapan proses *training* data diawali dengan fitur ektraksi pada data teks menggunakan TF-IDF (*Term Frequency-Inverse Document Frequency*), kemudian dilakukan proses *training* data dengan 1000 data untuk

membuat model klasifikasi yang dapat digunakan untuk melakukan klasifikasi sentimen secara otomatis. Kode dataset tersebut dapat dilihat di bawah ini.

```
import math
import random
from collections import defaultdict
from pprint import pprint

# Prevent future/deprecation warnings from showing in output
import warnings
warnings.filterwarnings(action='ignore')

import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Set global styles for plots
sns.set_style(style='white')
sns.set_context(context='notebook',                font_scale=1.3,
rc={'figure.figsize': (16,9)})
```


Untuk melatih *classifier*, kita perlu mengubah kata-kata dalam *tweet* menjadi angka, karena algoritma hanya tahu bagaimana bekerja dengan angka. Kalimat untuk menghitung TF-IDF manual dapat dilihat pada tabel 3.9.

Tabel 3. 9 Kalimat TF-IDF

Dokumen (d)	Kalimat
d1	dana jaminan hari tua bpjs ketenagakerjaan cari tunggu usia tahun syarat tentu cair offline online via
d2	lama amat ngurusin bpjs
d3	logika resign kalau butuh uang opsi akhir cair bpjstk aneh banget bijak sungguh negara wakanda
d4	jaminan hari tua kalau masih kalau sudah phk nunggu umur tahun uang jht pakai untuk modal usaha dasar otak idiot kamu

Tabel 3.9 menjelaskan kalimat yang akan digunakan untuk menghitung TF-IDF secara manual. Perhitungan TF menggunakan beberapa komponen seperti t yaitu *term* (kata), d yaitu dokumen terdiri dari d1, d2, d3, d4 dan df yaitu banyaknya dokumen dimana suatu term muncul. Perhitungan DF dapat dilihat pada Tabel 3.10.

Tabel 3. 10 Perhitungan DF

t	d1	d2	d3	d4	DF
dana	1				1
jaminan	1			1	2
hari	1			1	2
tua	1			1	2
bpjs		1			1
ketenagakerjaan	1				1
cari	1				1
tunggu	1				1
usia	1				1
tahun	1				1
syarat	1				1
tentu	1				1
cair	1		1		2
offline	1				1
online	1				1
via	1				1
lama		1			1
amat		1			1
ngurusin		1			1
logika			1		1
resign			1		1
kalau			1	2	3
butuh			1		1
uang			1		1
opsi			1		1
akhir			1		1
bpjstk			1		1
aneh			1		1

banget			1		1
bijak			1		1
sungguh			1		1
negara			1		1
wakanda			1		1
masih			1		1
sudah			1		1
phk			1		1
nunggu			1		1
umur			1		1
jht			1		1
pakai			1		1
untuk			1		1
modal			1		1
usaha			1		1
dasar			1		1
otak			1		1
idiot			1		1
kamu			1		1

Tabel 3.10 menjelaskan distribusi tiap term atau kata pada kalimat yang terdapat pada dokumen. Perhitungan IDF menggunakan beberapa komponen seperti *term* (kata) , df dan idf, yaitu hubungan tersedianya sebuah *term* (kata) dalam seluruh dokumen yang dihitung dengan N (banyaknya dokumen), seperti menggunakan rumus pada persamaan (1). Hasil perhitungan IDF dapat dilihat pada tabel 3.11.

Tabel 3. 11 Perhitungan IDF

t	DF	IDF		idf(N=4)	idf(N=1000)
dana	1	1		0.602059991	3
jaminan	2	0.5		0.301029996	2.698970004
hari	2	0.5		0.301029996	2.698970004
tua	2	0.5		0.301029996	2.698970004
bpjs	1	1		0.602059991	3
ketenagakerjaan	1	1		0.602059991	3
cari	1	1		0.602059991	3
tunggu	1	1		0.602059991	3
usia	1	1		0.602059991	3
tahun	1	1		0.602059991	3
syarat	1	1		0.602059991	3
tentu	1	1		0.602059991	3
cair	2	0.5		0.301029996	2.698970004
offline	1	1		0.602059991	3
online	1	1		0.602059991	3
via	1	1		0.602059991	3
lama	1	1		0.602059991	3
amat	1	1		0.602059991	3
ngurusin	1	1		0.602059991	3
logika	1	1		0.602059991	3
resign	1	1		0.602059991	3
kalau	3	0.333333		0.124938737	2.522878745
butuh	1	1		0.602059991	3
uang	1	1		0.602059991	3
opsi	1	1		0.602059991	3
akhir	1	1		0.602059991	3
bpjstk	1	1		0.602059991	3
aneh	1	1		0.602059991	3

banget	1	1		0.602059991	3
bijak	1	1		0.602059991	3
sungguh	1	1		0.602059991	3
negara	1	1		0.602059991	3
wakanda	1	1		0.602059991	3
masih	1	1		0.602059991	3
sudah	1	1		0.602059991	3
phk	1	1		0.602059991	3
nunggu	1	1		0.602059991	3
umur	1	1		0.602059991	3
jht	1	1		0.602059991	3
pakai	1	1		0.602059991	3
untuk	1	1		0.602059991	3
modal	1	1		0.602059991	3
usaha	1	1		0.602059991	3
dasar	1	1		0.602059991	3
otak	1	1		0.602059991	3
idiot	1	1		0.602059991	3
kamu	1	1		0.602059991	3
butuh	1	1		0.602059991	3
uang	1	1		0.602059991	3
opsi	1	1		0.602059991	3
akhir	1	1		0.602059991	3
bpjstk	1	1		0.602059991	3
aneh	1	1		0.602059991	3
banget	1	1		0.602059991	3
bijak	1	1		0.602059991	3
sungguh	1	1		0.602059991	3
negara	1	1		0.602059991	3
wakanda	1	1		0.602059991	3

masih	1	1		0.602059991	3
sudah	1	1		0.602059991	3
phk	1	1		0.602059991	3
nunggu	1	1		0.602059991	3
umur	1	1		0.602059991	3
jht	1	1		0.602059991	3
pakai	1	1		0.602059991	3
untuk	1	1		0.602059991	3
modal	1	1		0.602059991	3
usaha	1	1		0.602059991	3
dasar	1	1		0.602059991	3
ota	1	1		0.602059991	3
idiot	1	1		0.602059991	3
kamu	1	1		0.602059991	3

Selanjutnya melakukan perhitungan TF-IDF dapat dilihat pada tabel 3. 12.

Tabel 3. 12 Perhitungan TF-IDF

	TF*IDF			
t	**d1**	**d2**	**d3**	**d4**
dana	1	0	0	0
jaminan	0.5	0	0	0.5
hari	0.5	0	0	0.5
tua	0.5	0	0	0.5
bpjs	0	1	0	0
ketenagakerjaan	1	0	0	0
cari	1	0	0	0
tunggu	1	0	0	0
usia	1	0	0	0
tahun	1	0	0	0

syarat	1	0	0	0
tentu	1	0	0	0
cair	0.5	0	0.5	0
offline	1	0	0	0
online	1	0	0	0
via	1	0	0	0
lama	0	1	0	0
amat	0	1	0	0
ngurusin	0	1	0	0
logika	0	0	1	0
resign	0	0	1	0
kalau	0	0	0.333333	0.666666
butuh	0	0	1	0
uang	0	0	1	0
opsi	0	0	1	0
akhir	0	0	1	0
bpjstk	0	0	1	0
aneh	0	0	1	0
banget	0	0	1	0
bijak	0	0	1	0
sungguh	0	0	1	0
negara	0	0	1	0
wakanda	0	0	1	0
masih	0	0	1	0
sudah	0	0	1	0
phk	0	0	1	0
nunggu	0	0	1	0
umur	0	0	1	0
jht	0	0	1	0
pakai	0	0	1	0

untuk	0	0	1	0
modal	0	0	1	0
usaha	0	0	1	0
dasar	0	0	1	0
otak	0	0	1	0
idiot	0	0	1	0
kamu	0	0	1	0

Tabel 3.12 menjelaskan tentang perhitungan TF-IDF secara manual dengan rumus hasil perkalian antara TF dengan IDF.

Berikutnya yaitu klasifikasi menggunakan fitur ekstraksi TF-IDF yang menghasilkan perhitungan secara otomatis dengan pembobotan pada kata yang ada pada term atau dokumen pada data *training*. Perhitungan TF-IDF menggunakan *library* pada Python yaitu *Sklearn* dan *TfidfVectorizer* untuk menghitung hasil perhitungan secara otomatis. Pada perhitungan TF-IDF didukung dengan *library Multinomial Naive Bayes* dengan menggunakan data *training* yang tersedia. Kode perhitungan TF-IDF otomatis dapat dilihat di bawah ini.


```
from sklearn.feature_extraction.text import TfidfVectorizer

s1 = "dana jaminan hari tua bpjs ketenagakerjaan cari tunggu usia
tahun syarat tentu cair offline online via"
s2 = "lama amat ngurusin bpjs"

vect = TfidfVectorizer()
X = vect.fit_transform([s1, s2])

X.toarray()

list(zip(X.toarray()[0], vect.get_feature_names())
```

Hasil perhitungan TF-IDF otomatis dapat dilihat di bawah ini.

```
[(0.0, 'amat'),
 (0.18068688270171665, 'bpjs'),
 (0.2539491091300954, 'cair'),
 (0.2539491091300954, 'cari'),
 (0.2539491091300954, 'dana'),
 (0.2539491091300954, 'hari'),
 (0.2539491091300954, 'jaminan'),
```

```
 (0.2539491091300954, 'ketenagakerjaan'),
 (0.0, 'lama'),
 (0.0, 'ngurusin'),
 (0.2539491091300954, 'offline'),
 (0.2539491091300954, 'online'),
 (0.2539491091300954, 'syarat'),
 (0.2539491091300954, 'tahun'),
 (0.2539491091300954, 'tentu'),
 (0.2539491091300954, 'tua'),
 (0.2539491091300954, 'tunggu'),
 (0.2539491091300954, 'usia'),
 (0.2539491091300954, 'via')]
```

Setelah itu tahap berikutnya adalah *Cross-Validation*, yaitu cara untuk menghitung hasil akurasi sebanyak beberapa kali (*k-fold*) dengan menggunakan parameter yang sama. Proses *Cross-Validation* digunakan untuk mencari nilai akurasi dengan melakukan percobaan beberapa kali agar dapat diketahui tingkat performa dari model dan data yang digunakan. Kode untuk menghitung *Cross-Validation* dapat dilihat di bawah ini.


```
from sklearn.model_selection import ShuffleSplit

X = df.content
y = df.actual

ss = ShuffleSplit(n_splits=10, test_size=0.2)
sm = SMOTE()

accs = []
f1s = []
cms = []

for train_index, test_index in ss.split(X):

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit vectorizer and transform X train, then transform X test
    X_train_vect = vect.fit_transform(X_train)
    X_test_vect = vect.transform(X_test)

    # Oversample
    X_train_res,   y_train_res   =   sm.fit_resample(X_train_vect,
y_train)

    # Fit Naive Bayes on the vectorized X with y train labels,
    # then predict new y labels using X test
    nb.fit(X_train_res, y_train_res)
    y_pred = nb.predict(X_test_vect)
```

```
    # Determine test set accuracy and f1 score on this fold using
the true y labels and predicted y labels
    accs.append(accuracy_score(y_test, y_pred))
    f1s.append(f1_score(y_test, y_pred, average='weighted'))
    cms.append(confusion_matrix(y_test, y_pred))

print("\nAverage accuracy across folds: {:.2f}%".format(sum(accs)
/ len(accs) * 100))
print("\nAverage F1 score across folds: {:.2f}%".format(sum(f1s) /
len(f1s) * 100))
print("\nAverage    Confusion    Matrix    across    folds:    \n
{}".format(sum(cms) / len(cms)))
```

Hasil perhitungan *cross validatioin* dapat dilihat di bawah ini.

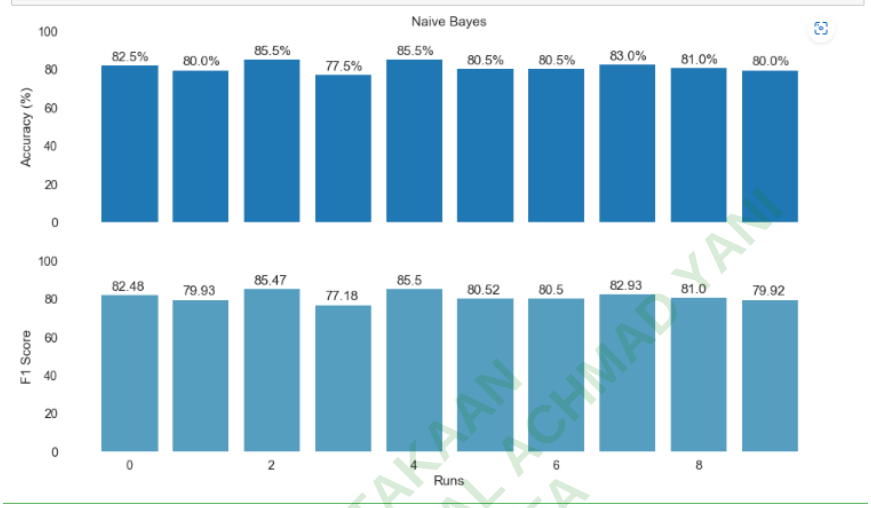
```
Average accuracy across folds: 81.60%

Average F1 score across folds: 81.54%

Average Confusion Matrix across folds:
[[77.8 20.7]
[16.1 85.4]]
```

Setelah itu tampilkan dalam bentuk diagram presentase. Kode untuk menampilkan diagram presentase dapat dilihat pada gambar

```
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(list(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()
```

Diagram presentase dapat dilihat pada Gambar 3.2.



Gambar 3. 2 Diagram Presentase

Setelah *cross-validation* maka dilanjutkan dengan pembuatan model klasifikasi menggunakan variabel x dan y dengan data *training* yang sudah tersedia. Model dibuat dalam sebuah fungsi agar nantinya memudahkan dalam pemanggilan dan eksekusi pada tahap berikutnya sehingga lebih efektif dan efisien. Kode untuk pembuatan model klasifikasi dapat dilihat di bawah ini.

```
import numpy as np
import pandas as pd
import os
import pickle
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer

X = df.content
y = df.actual

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))
```

```
#save ke PICKLE
files = open('model_classifier_nbc.pickle', 'wb')
pickle.dump(text_classifier, files)
files.close()

print('Proses Training Naive Bayes Selesai!')
```

Model yang diberi nama pipeline selanjutnya disimpan dalam bentuk file *pickle* agar dapat dibuka kembali dan dapat digunakan lagi. Kode untuk membuka file *pickle* dapat dilihat di bawah ini.

```
model = open('model_classifier_nbc.pickle', 'rb')
nbc_classifier = pickle.load(model)
print(nbc_classifier)
```

File *pickle* yang menjadi model klasifikasi digunakan untuk melakukan eksekusi pada data *training* sebanyak 1000 *tweet*. Yang nantinya akan dieksekusi menggunakan *classifier* yang sudah dibuat sehingga dapat dilakukan prediksi sifat positif dan negatif dari *term* atau dokumen yang tersedia. Hasil prediksi dapat dilihat pada Tabel 3.13.

Tabel 3. 13 Hasil Prediksi

No	Content	Sentimen
1	urus dl dijadiin bpjstk resign buka usaha tapi tidak modal	-1
2	suru resign lagi biar ambil bpjstk	-1
3	anti alamat email daftar bpjstk via telpon email sulit diresponya	-1
4	selamat sore ka kartu ganti kartu bpjs sehat ambil terimakasih	1
5	selamat sore admin nanya kalau cair bpjstk paklaringnya lengkap	1

2. Testing.

Testing adalah tahapan untuk mengetahui tingkat keakuratan pemodelan yang dibangun pada tahap *training* yang digunakan untuk memprediksi label atau kelas dari data uji yang tersedia. Pada proses *testing* data yang dilakukan untuk pengujian adalah 500 data *tweet* yang sudah dilakukan pelabelan manual sehingga sudah didapatkan label positif dan negatif pada tiap data *tweet* yang tersedia.

Tahap berikutnya masukan data untuk *testing* yang sudah terlabeli manual. Kode untuk membuka data dapat dilihat di bawah ini.

```
def load_data():
    data = pd.read_csv('testing_NBC.csv')
    return data
```

Berikutnya jalankan kode untuk memprediksi data *testing* untuk mendapatkan nilai negatif dan positif. Kode untuk melakukan prediksi dapat dilihat di bawah ini.

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

predicted

 result_tweet=[]
for i in range(len(predicted)):
    if(predicted[i]==2):
        sentiment_result='puas'
    elif(predicted[i]==1):
        sentiment_result='senang'
    elif(predicted[i]==-1):
        sentiment_result='sedih'
    elif(predicted[i]==-2):
        sentiment_result='kecewa'
    result_tweet.append({'tweet':data_tweet[i],
'class':predicted[i]})
    #                 result_tweet.append({'tweet':data_tweet[i],
'class':predicted[i], 'result_nbc':sentiment_result})
```


Data hasil dari pelabelan otomatis dapat dilihat pada Tabel 3.14.

Tabel 3. 14 Hasil Pelabelan Otomatis

No	Content	Sentimen
1	urus dl dijadiin bpjstk resign buka usaha tapi tidak modal	-1
2	suru resign lagi biar ambil bpjstk	-1
3	anti alamat email daftar bpjstk via telpon email sulit diresponya	-1
4	selamat sore ka kartu ganti kartu bpjs sehat ambil terimakasih	1
5	selamat sore admin nanya kalau cair bpjstk paklaringnya lengkap	1

Setelah mendapatkan pelabelan otomatis maka dilakukan perhitungan nilai *accuracy*, *precision*, *recal* dan *f1-score* pada data *testing naïve bayes classifier*, proses *confusion matrix* dapat dilihat pada persamaan .

Kode untuk menghitung *accuracy* dapat dilihat di bawah ini.

```
accuracy = (TP+TN)/(TP+TN+FP+FN)
print('Accuracy =', accuracy)
```

Kode untuk menghitung nilai *precision* dapat dilihat di bawah ini.

```
precission = (TP) / (TP+FP)
print('Precission =', precission)
```

Kode untuk menghitung nilai *recall* dapat dilihat di bawah ini.

```
recall = (TP) / (TP + FN)
print('Recall =', recall)
```

Kode untuk menghitung nilai *F1-Score* dapat dilihat di bawah ini.

```
F1_Score = 2 * (recall*precission) / (recall + precission)
print('F1Score =', F1_Score)
```

3. Klasfikasi Sentimen

Pada proses klasifikasi data menggunakan keseluruhan data yang ada yaitu 4.154 data yang nantinya akan di labeli oleh mesin. Data *training* yang belum terlabeli dapat dilihat pada Tabel 3.15.

Tabel 3. 15 Data Training

No	Content
1	with benefit sudah biaya bpjs kah
2	gratis kyknya kalau bpjs sana
3	booby sehat kalau obat habis buru obat kalau sanggup bayar bpjs beli bbm paling ojol gratis
4	wiw opname dapat kelas vip bpjs
5	kalau halodok murah dapat tapi kalau domter an smpe juga kalau bpjs gratis

Setelah data yang digunakan untuk *training* sudah di buka tahap selanjutnya adalah membuka file *pickle*. Kode untuk membuka file *pickle* dapat dilihat di bawah ini.

```
model = open('model_classifier_nbc.pickle', 'rb')
nbc_classifier = pickle.load(model)
print(nbc_classifier)
```

Tahap selanjutnya adalah melakukan prediksi untuk memperoleh pelabelan negatif dan positif menggunakan mesin. Kode untuk pelabelan otomatis dapat dilihat di bawah ini.

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

predicted

 result_tweet=[]
for i in range(len(predicted)):
    if(predicted[i]==2):
        sentiment_result='puas'
    elif(predicted[i]==1):
        sentiment_result='senang'
    elif(predicted[i]==-1):
        sentiment_result='sedih'
    elif(predicted[i]==-2):
        sentiment_result='kecewa'
    result_tweet.append({'tweet':data_tweet[i],
'class':predicted[i]})
    #                 result_tweet.append({'tweet':data_tweet[i],
'class':predicted[i], 'result_nbc':sentiment_result})
```

Hasil prediksi otomatis dapat dilihat pada Tabel 3.16.

Tabel 3. 16 Hasil Prediksi Otomatis

No	Content	Sentimen
1	with benefit sudah biaya bpjs kah	-1
2	gratis kyknya kalau bpjs sana	-1
3	booby sehat kalau obat habis buru obat kalau sanggup bayar bpjs beli bbm paling ojol gratis	-1
4	wiw opname dapat kelas vip bpjs	1
5	kalau halodok murah dapat tapi kalau domter an smpe juga kalau bpjs gratis	1

Berikutnya tampilkan jumlah sentimen negatif dan positif dari data 4.154. Kode untuk menampilkan jumlah tersebut dapat dilihat di bawah ini.

```
data.groupby(by='class').agg('count')
```

Jumlah data negatif dan positif dapat dilihat pada Tabel 3.17.

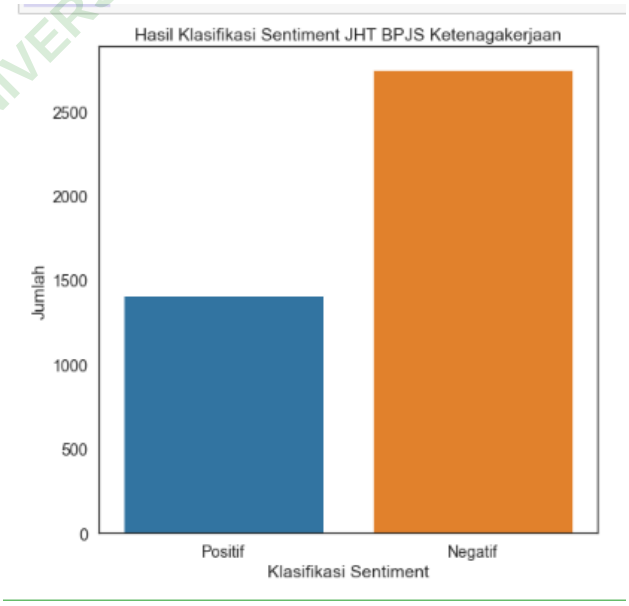
Tabel 3. 17 Jumlah Negatif dan Positif

Tweet	
Negatif	402
Positif	98

Berikutnya tampikan dalam bentuk *histogram*. Kode unutk menampilkan *histogram* dapat dilihat pada gambar 3.46.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(8, 8))
sentiment = ['Positif','Negatif']
jumlah_sentiment = [1408, 2746]
sns.barplot(sentiment,jumlah_sentiment)
ax.set_ylabel('Jumlah')
ax.set_xlabel('Klasifikasi Sentiment')
ax.set_title('Hasil Klasifikasi Sentiment JHT BPJS
Ketenagakerjaan')
plt.show()
```

Tampilan *histogram* dapat dilihat pada Gambar 3.3.



Gambar 3. 3 Histogram