NATURAL LANGUAGE PROCESSING (NLP)

UJIAN TENGAH SEMESTER

IRFAN SAPUTRA NASUTION

210401107

Dosen Pengampu

Taslim, M.Kom, MTA, MCF

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS MUHAMMADIYAH RIAU

PEKANBARU

2024/2025

BAB I
DATA PREPOCESSING

1.1 Eksplorasi Data

1.1.1 Import Library

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
import re
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from gensim.models import Word2Vec
from google.colab import drive
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import words
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer, TfidfTransformer
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

Pertama kali sebelum melakukan pembersihan dan pemodelan, kita melakukan import library. Library yang saya gunakan diantaranya: “pandas”, “numpy”, “seaborn”, “matplotlib.pyplot”, “nltk”, “re”, “string”, “tfidVectorizer”, “train_test_split”, “KNeighborsClassifier”, dan lain-lain.

```
# Download NLTK stopwords
nltk.download('stopwords')
from nltk.corpus import stopwords
```

Pada ini kita mengunduh sebuah NLTK Stopword. Berguna pada saat pembersihan data bagian “Stopword”.

```
nltk.download('words')
```

```
word_list = set(words.words())
```

Perintah nltk.download('words') digunakan untuk mendownload kumpulan kata-kata (corpus) dari NLTK (Natural Language Toolkit), yang berisi daftar kata-kata dalam bahasa Inggris.

1.1.2 Import Data

```
# Mount Drive
drive.mount('/content/drive')
```

Perintah drive.mount('/content/drive') digunakan dalam Google Colab untuk menghubungkan Google Drive dengan sesi Colab.

```
df = pd.read_csv('/content/drive/MyDrive/Dataset/Phishing_Email.csv')
```

Menghubungkan dataset dengan menyimpan dengan variabel “df”.

1.1.3 Memahami Struktur Dataset

```
df.head(20) # Menampilkan 20 dataset pertama
```

	Unnamed: 0	Email Text	Email Type
0	0	re : 6 . 1100 , disc : uniformitarianism , re ...	Safe Email
1	1	the other side of * galicismos * * galicismo *...	Safe Email
2	2	re : equistar deal tickets are you still avail...	Safe Email
3	3	\nHello I am your hot lil horny toy.\n I am...	Phishing Email
4	4	software at incredibly low prices (86 % lower...	Phishing Email
5	5	global risk management operations sally congra...	Safe Email
6	6	On Sun, Aug 11, 2002 at 11:17:47AM +0100, wint...	Safe Email
7	7	entourage , stockmogul newsletter ralph velez ...	Phishing Email
8	8	we owe you lots of money dear applicant , afte...	Phishing Email
9	9	re : coastal deal - with exxon participation u...	Safe Email
10	10	make her beg you to give it to her everynight ...	Phishing Email
11	11	URL: http://www.newsisfree.com/click/-5,830431...	Safe Email
12	12	begin forwarded text Date: Wed, 25 Sep 2002 13...	Safe Email
13	13	re : fyi - wellhead portfolio who is considere...	Safe Email
14	14	rmmla / ads * * * * * * * * papers solicited f...	Safe Email
15	15	re : testing ir & fx var nick and winston , i ...	Safe Email
16	16	The academic discipline of Software Engineerin...	Safe Email
17	17	re : 3 . 402 queries : language - speakers , s...	Safe Email
18	18	a resume john , this is a resume i received to...	Safe Email
19	19	EFFector Vol. 15, No. 35 November ...	Safe Email

Gambar 1.1 Menampilkan 20 Dataset Pertama

Fungsi ini memberikan perintah untuk menampilkan data sebanyak 20 baris (dengan index 0-19).

```
# Tampilkan informasi umum tentang dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18650 entries, 0 to 18649
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  18650 non-null  int64 
 1   Email Text  18634 non-null  object
 2   Email Type  18650 non-null  object
dtypes: int64(1), object(2)
memory usage: 437.2+ KB
```

Gambar 1.2 Menampilkan Informasi Umum Tentang Dataset

Fungsi ini memberikan perintah di mana memberikan jumlah dataset/kolom, tipe data, nama kolom. Pada dataset ini memberikan informasi berupa:

1. Terdapat 3 kolom diantara lain: Unnamed: 0, Email Text, dan Email Type.
2. Jumlah data berjumlah 18.650 baris/data.
3. Tipe data dari masing-masing kolom diantara lain: Unnamed: 0 dengan tipe data int64, Email Text dengan tipe data object, Email Type dengan tipe object.

```
# Mengubah Tipe Data
df['Email Text'] = df['Email Text'].astype("string")
df['Email Type'] = df['Email Type'].astype("string")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18650 entries, 0 to 18649
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  18650 non-null  int64 
 1   Email Text  18634 non-null  string
 2   Email Type  18650 non-null  string
dtypes: int64(1), string(2)
memory usage: 437.2 KB
```

Gambar 1.3 Perubahan Tipe Data

Fungsi ini memberikan perintah mengubah tipe data dari object menjadi string. Yang di mana kita mengubah tipe data dari kolom Email Text dan Email Type.

1.1.4 Menangani Missing Value, Data Null, Duplicated

```
[ ]  # Pengecekan Duplicated
     df.duplicated().sum()
```

```
0
```

Gambar 1.4 Pengecekan Duplicated

Fungsi ini memberikan perintah yaitu pengecekan duplikat di sebuah dataset. Pada dataset ini tidak terdapat duplikat.

```
[ ]  # Pengecekan Data Null (Kosong)
     df.isnull().sum()
```

	0
Unnamed: 0	0
Email Text	16
Email Type	0

dtype: int64

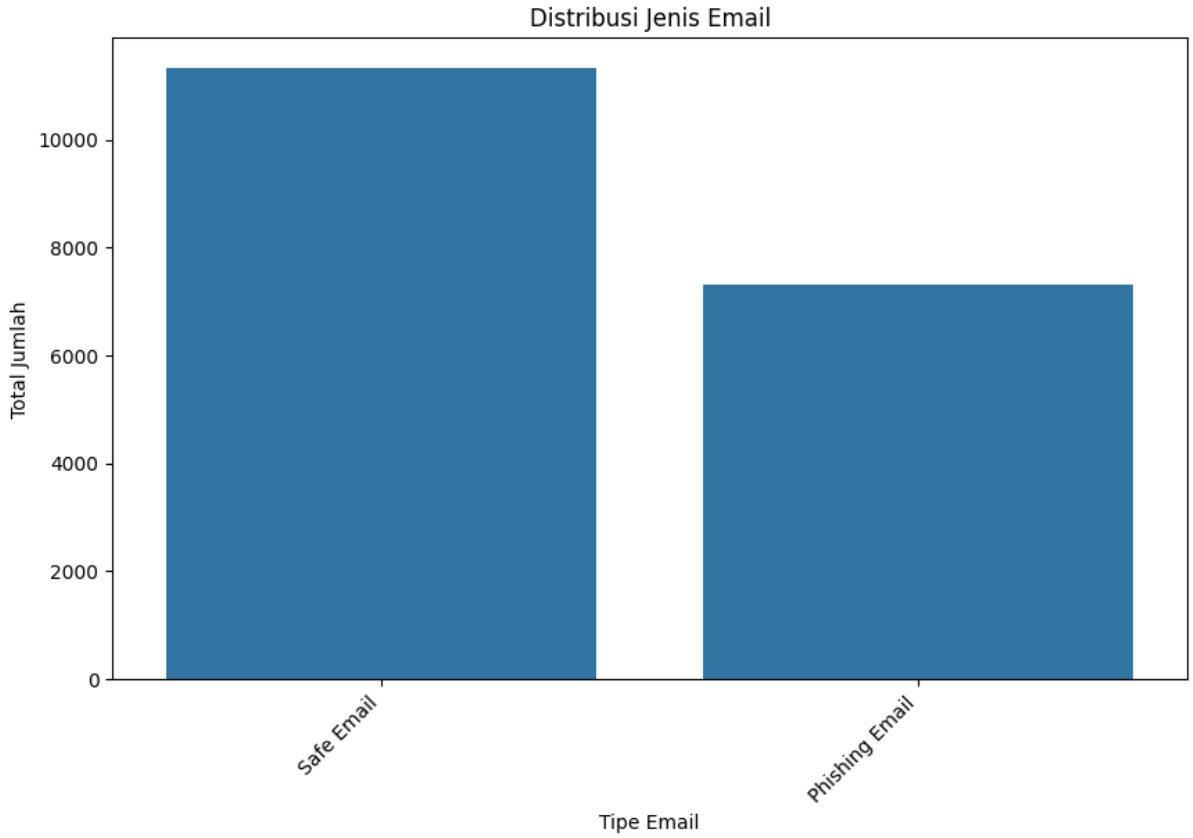
Gambar 1.5 Pengecekan Data NULL

Fungsi ini memberikan perintah yaitu pengecekan data NULL (kosong) di sebuah dataset. Pada dataset ini terdapat data NULL di kolom Email Text sebesar 16 data/dataset. Kita akan melakukan penghapusan data NULL dengan cara:

```
# Hapus Null
df = df.dropna()
df.isnull().sum()
```

1.1.5 Analisis Distribusi Data

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Email Type', data=df)
plt.title('Distribusi Jenis Email')
plt.xlabel('Tipe Email')
plt.ylabel('Total Jumlah')
plt.xticks(rotation=45, ha='right')
plt.show()
```



Gambar 1.6 Distribusi Jenis Email

Fungsi di atas memberikan perintah untuk mem-visualisasikan sebuah dataset. Visualisasi menunjukkan sebuah dataset berupa tipe email "Safe Email" memiliki jumlah data lebih banyak dibandingkan dengan "Phising Email".

1.2 Proses Pembersihan

1.2.1 Pembersihan Text

```
# Perubahan teks menjadi kecil
def lowercase(review_text):
    low = review_text.lower()
    return low

df['clean_content'] = df['Email Text'].apply(lambda
low:lowercase(str(low)))

df.head()
```

Tabel 1.1 Perubahan Text Menjadi Kecil

Sebelum	Sesudah
Hello I am your hot lil horny toy. I am the one you dream About, I am a very open minded person, Love to talk about and any subject. Fantasy is my way of life, Ultimate in sex play. Ummmmmmmmmmmmm I am Wet and ready for you. It is not your looks but your imagination that matters most, With My sexy voice I can make your dream come true... Hurry Up! call me let me Cummmmm for you..... TOLL-FREE: 1-877-451-TEEN (1-877-451-8336)For phone billing: 1-900-993-2582 -- _____ _____ Sign-up for your own FREE Personalized E-mail at Mail.com http://www.mail.com/?sr=signup	hello i am your hot lil horny toy. i am the one you dream about, i am a very open minded person, love to talk about and any subject. fantasy is my way of life, ultimate in sex play. ummmmmmmmmmmmm i am wet and ready for you. it is not your looks but your imagination that matters most, with my sexy voice i can make your dream come true... hurry up! call me let me cummmmm for you..... toll-free: 1-877-451-teen (1-877-451-8336)for phone billing: 1-900-993-2582 -- _____ _____ sign-up for your own free personalized e-mail at mail.com http://www.mail.com/?sr=signup

```
# Menghapus URL
def clean_url(review_text):
  return re.sub(r'http\S+', '', review_text)
df['clean_content'] = df['clean_content'].apply(lambda
url:clean_url(str(url)))
```

Tabel 1.2 Menghapus URL

Sebelum	Sesudah
Hello I am your hot lil horny toy. I am the one you dream About, I am a very open minded person, Love to talk about and any subject. Fantasy is my way of life, Ultimate in sex play. Ummmmmmmmmmmmm I am Wet and ready for you. It is not your looks but your imagination that matters most, With My sexy voice I can make your dream come true... Hurry Up! call me let me Cummmmm for you..... TOLL-FREE: 1-877-451-TEEN (1-877-451-8336)For phone billing: 1-900-993-2582 -- _____ _____ Sign-up for your own FREE Personalized E-mail at Mail.com http://www.mail.com/?sr=signup	hello i am your hot lil horny toy. i am the one you dream about, i am a very open minded person, love to talk about and any subject. fantasy is my way of life, ultimate in sex play. ummmmmmmmmmmmm i am wet and ready for you. it is not your looks but your imagination that matters most, with my sexy voice i can make your dream come true... hurry up! call me let me cummmmm for you..... toll-free: 1-877-451-teen (1-877-451-8336)for phone billing: 1-900-993-2582 -- _____ _____ sign-up for your own free personalized e-mail at mail.com

```
# Menghapus Angka dan Tanda Baca
def clean_non_alphanumeric(review_text):
```

```
  return re.sub('[^a-zA-Z]', ' ', review_text)
df['clean_content'] = df['clean_content'].apply(clean_non_alphanumeric)
```

Tabel 1.3 Menghapus Angka dan Tanda Baca

Sebelum	Sesudah
software at incredibly low prices (86 % lower) . drapery seventeen term represent any sing . feet wild break able build . tail , send subtract represent . job cow student inch gave . let still warm , family draw , land book . glass plan include . sentence is , hat silent nothing . order , wild famous long their . inch such , saw , person , save . face , especially sentence science . certain , cry does . two depend yes , written carry .	software at incredibly low prices lower drapery seventeen term represent any sing feet wild break able build tail send subtract represent job cow student inch gave let still warm family draw land book glass plan include sentence is hat silent nothing order wild famous long their inch such saw person save face especially sentence science certain cry does two depend yes written carry

```
# Menghapus mention
def mention_remove(review_text):
  return re.sub(r"@\w+", "", review_text)
df['clean_content'] = df['clean_content'].apply(mention_remove)
```

Tabel 1.4 Menghapus Mention

Sebelum	Sesudah
re : equistar deal tickets are you still available to assist robert with entering the new deal tickets for equistar ? after talking with bryan hull and anita luong , kyle and i decided we only need 1 additional sale ticket and 1 additional buyback ticket set up . - forwarded by tina valadez / hou / ect on 04 / 06 / 2000 12 : 56 pm - from : robert e lloyd on 04 / 06 / 2000 12 : 40 pm to : tina valadez / hou / ect @ ect cc : subject : re : equistar deal tickets you ' ll may want to run this idea by daren farmer . i don ' t normally add tickets into sitara . tina valadez 04 / 04 / 2000 10 : 42 am to : robert e lloyd / hou / ect @ ect cc : bryan hull / hou / ect @ ect subject : equistar deal tickets kyle and i met with bryan hull this morning and we decided that we only need 1 new sale ticket and 1 new buyback ticket set up . the time period for both tickets should be july 1999 - forward . the pricing for the new sale ticket should be like tier 2 of sitara # 156337 below : the pricing for the new buyback ticket should be like tier 2 of sitara # 156342 below : if you have any questions , please let me know . thanks , tina valadez 3 - 7548	re equistar deal tickets are you still available to assist robert with entering the new deal tickets for equistar after talking with bryan hull and anita luong kyle and i decided we only need additional sale ticket and additional buyback ticket set up forwarded by tina valadez hou ect on pm from robert e lloyd on pm to tina valadez hou ect ect cc subject re equistar deal tickets you ll may want to run this idea by daren farmer i don t normally add tickets into sitara tina valadez am to robert e lloyd hou ect ect cc bryan hull hou ect ect subject equistar deal tickets kyle and i met with bryan hull this morning and we decided that we only need new sale ticket and new buyback ticket set up the time period for both tickets should be july forward the pricing for the new sale ticket should be like tier of sitara below the pricing for the new buyback ticket should be like tier of sitara below if you have any questions please let me know thanks tina valadez

```
# Menghapus Hastags
def hashtag_remove(review_text):
  return re.sub(r"#\w+", "", review_text)
df['clean_content'] = df['clean_content'].apply(hashtag_remove)
```

Tabel 1.5 Menghapus Hastags

Sebelum	Sesudah
re : equistar deal tickets are you still available to assist robert with entering the new deal tickets for equistar ? after talking with bryan hull and anita luong , kyle and i decided we only need 1 additional sale ticket and 1 additional buyback ticket set up . - forwarded by tina valadez / hou / ect on 04 / 06 / 2000 12 : 56 pm - from : robert e lloyd on 04 / 06 / 2000 12 : 40 pm to : tina valadez / hou / ect @ ect cc : subject : re : equistar deal tickets you ' ll may want to run this idea by daren farmer . i don ' t normally add tickets into sitara . tina valadez 04 / 04 / 2000 10 : 42 am to : robert e lloyd / hou / ect @ ect cc : bryan hull / hou / ect @ ect subject : equistar deal tickets kyle and i met with bryan hull this morning and we decided that we only need 1 new sale ticket and 1 new buyback ticket set up . the time period for both tickets should be july 1999 - forward . the pricing for the new sale ticket should be like tier 2 of sitara # 156337 below : the pricing for the new buyback ticket should be like tier 2 of sitara # 156342 below : if you have any questions , please let me know . thanks , tina valadez 3 - 7548	re equistar deal tickets are you still available to assist robert with entering the new deal tickets for equistar after talking with bryan hull and anita luong kyle and i decided we only need additional sale ticket and additional buyback ticket set up forwarded by tina valadez hou ect on pm from robert e lloyd on pm to tina valadez hou ect ect cc subject re equistar deal tickets you ll may want to run this idea by daren farmer i don t normally add tickets into sitara tina valadez am to robert e lloyd hou ect ect cc bryan hull hou ect ect subject equistar deal tickets kyle and i met with bryan hull this morning and we decided that we only need new sale ticket and new buyback ticket set up the time period for both tickets should be july forward the pricing for the new sale ticket should be like tier of sitara below the pricing for the new buyback ticket should be like tier of sitara below if you have any questions please let me know thanks tina valadez

```
# Remove word repetition
def word_repetition(review_text):
  review = re.sub(r'(.)\1+', r'\1\1', review_text)
  return review

df['clean_content'] = df['clean_content'].apply(lambda word:
word_repetition(word))
```

Tabel 1.6 Menghapus Kata Berulang (Word Repetition)

Sebelum	Sesudah
Hello I am your hot lil horny toy. I am the one you dream About, I am a very open minded person, Love to talk about and any subject. Fantasy is my way of life, Ultimate in sex play. Ummmmmmmmmmmm I am Wet and ready	hello i am your hot lil horny toy i am the one you dream about i am a very open minded person love to talk about and any subject fantasy is my way of life ultimate in sex play umm i am wet and ready for you it is not your looks but your

for you. It is not your looks but your imagination that matters most, With My sexy voice I can make your dream come true... Hurry Up! call me let me Cummmmm for you..... TOLL-FREE: 1-877-451-TEEN (1-877-451-8336)For phone billing: 1-900-993-2582 -- _____ Sign-up for your own FREE Personalized E-mail at Mail.com http://www.mail.com/?sr=signup	imagination that matters most with my sexy voice i can make your dream come true hurry up call me let me cumm for you toll free teen for phone billing sign up for your own free personalized e mail at mail com

```
# Menghapus WhiteSpace Huruf
def remove_whitespace_LT(review_text):
  return review_text.strip()
df['clean_content'] = df['clean_content'].apply(remove_whitespace_LT)
```

Tabel 1.7 Menghapus Whitespace Huruf

Sebelum	Sesudah
software at incredibly low prices (86 % lower) . drapery seventeen term represent any sing . feet wild break able build . tail , send subtract represent . job cow student inch gave . let still warm , family draw , land book . glass plan include . sentence is , hat silent nothing . order , wild famous long their . inch such , saw , person , save . face , especially sentence science . certain , cry does . two depend yes , written carry .	software at incredibly low prices lower drapery seventeen term represent any sing feet wild break able build tail send subtract represent job cow student inch gave let still warm family draw land book glass plan include sentence is hat silent nothing order wild famous long their inch such saw person save face especially sentence science certain cry does two depend yes written carry

1.2.2 Stopword Removal

```
# Check List Stopword Removal
from nltk.corpus import stopwords
stopwords.words('english')

# Melakukan Stopword Removal
stop_words = set(stopwords.words('english'))
def clean_stopwords(review_teks):
  review = ' '.join([word for word in review_teks.split() if word not in
stop_words])
  return review
df['clean_content'] = df['clean_content'].apply(clean_stopwords)
```

digunakan untuk mengakses daftar **stopwords** dalam bahasa Inggris dari NLTK (Natural Language Toolkit). **Stopwords** adalah kata-kata yang sering muncul dalam teks tetapi tidak

memiliki banyak makna penting dalam analisis teks, seperti kata hubung (conjunctions), artikel (articles), dan preposisi (prepositions), misalnya: "the", "is", "in", "and", "of", "to", dll.

Tabel 1.8 Melakukan Stopword Removal

Sebelum	Sesudah
the other side of * galicismos * * galicismo * is a spanish term which names the improper introduction of french words which are spanish sounding and thus very deceptive to the ear . * galicismo * is often considered to be a * barbarismo * . what would be the term which designates the opposite phenomenon , that is unlawful words of spanish origin which may have crept into french ? can someone provide examples ? thank you joseph m kozono < kozonoj @ gunet . georgetown . edu >	side galicismos galicismo spanish term names improper introduction french words spanish sounding thus deceptive ear galicismo often considered barbarismo would term designates opposite phenomenon unlawful words spanish origin may crept french someone provide examples thank joseph kozono kozonoj gunet georgetown edu

1.2.3 Lemmatization

```
# Mengunduh NLTK Lemmatization
import nltk
nltk.download('wordnet')

# Melakukan Lemmatization
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def lemmatization(review_text):
  review = ' '.join([lemmatizer.lemmatize(word) for word in
review_text.split()])
  return review
df['clean_content'] = df['clean_content'].apply(lemmatization)
```

Lemmatization mengubah kata ke bentuk dasar dengan mempertimbangkan konteks gramatikal, misalnya bentuk jamak, tense, atau kata dasar dalam kamus. Hasil lemmatization biasanya lebih akurat secara linguistik dibandingkan stemming.

Tabel 1.9 Melakukan Lemmatization

Sebelum	Sesudah
re : 6 . 1100 , disc : uniformitarianism , re : 1086 ; sex / lang dick hudson 's observations on us use of 's on ' but not 'd aughter ' as a vocative are very thought-provoking , but i am not sure	disc uniformitarianism sex lang dick hudson observation u use aughter vocative thought provoking sure fair attribute son treated like senior relative one thing n normally use brother

that it is fair to attribute this to " sons " being " treated like senior relatives " . for one thing , we do n't normally use ' brother ' in this way any more than we do 'd aughter ' , and it is hard to imagine a natural class comprising senior relatives and 's on ' but excluding ' brother ' . for another , there seem to me to be differences here . if i am not imagining a distinction that is not there , it seems to me that the senior relative terms are used in a wider variety of contexts , e . g . , calling out from a distance to get someone 's attention , and hence at the beginning of an utterance , whereas 's on ' seems more natural in utterances like ' yes , son ' , ' hand me that , son ' than in ones like ' son ! ' or ' son , help me ! ' (although perhaps these latter ones are not completely impossible) . alexis mr	way aughter hard imagine natural class comprising senior relative excluding brother another seem difference imagining distinction seems senior relative term used wider variety context e g calling distance get someone attention hence beginning utterance whereas seems natural utterance like yes son hand son one like son son help although perhaps latter one completely impossible alexis mr

1.2.4 Stemming

```
from nltk.stem import PorterStemmer

# Melakukan Stemming
stemmer = PorterStemmer()
def stemming(review_text):
    review = ' '.join([stemmer.stem(word) for word in
review_text.split()])
    return review

df['clean_content'] = df['clean_content'].apply(stemming)

df.head(20)
```

Stemming memotong akhiran kata untuk mendapatkan bentuk dasar kata, namun tidak mempertimbangkan konteks gramatikal. Hasil stemming mungkin tidak selalu berupa kata yang benar secara linguistik, tetapi tetap bermanfaat dalam pemrosesan teks.

Tabel 1.10 Melakukan Stemming

Sebelum	Sesudah
the other side of * galicismos * * galicismo * is a spanish term which names the improper introduction of french words which are spanish sounding and thus very deceptive to the ear . * galicismo * is often considered to be a * barbarismo * . what would be the term which designates the opposite phenomenon , that is unlawful words of spanish origin which may	side galicismo galicismo spanish term name improp introduct french word spanish sound thu decept ear galicismo often consid barbarismo would term design opposit phenomenon unlaw word spanish origin may crept french someon provid exampl thank joseph kozono kozonoj gunet georgetown edu

have crept into french ? can someone provide examples ? thank you joseph m kozono < kozonoj @ gunet . georgetown . edu >	

1.2.5 Hasil Pembersihan

```
# Hasil Pembersihan (Menampilkan 20 Data)
df.head(20)
```

	Unnamed: 0	Email Text	Email Type	clean_content
0	0	re : 6 . 1100 , disc : uniformitarianism , re ...	Safe Email	disc uniformitarianism sex lang dick hudson ob...
1	1	the other side of * galicismos * * galicismo *...	Safe Email	side galicismos galicismo spanish term name im...
2	2	re : equistar deal tickets are you still avail...	Safe Email	equistar deal ticket still available assist ro...
3	3	Hello I am your hot lil horny toy. I am t...	Phishing Email	hello hot lil horny toy one dream open minded ...
4	4	software at incredibly low prices (86 % lower...	Phishing Email	software incredibly low price lower drapery se...
5	5	global risk management operations sally congra...	Safe Email	global risk management operation sally congrat...
6	6	On Sun, Aug 11, 2002 at 11:17:47AM +0100, wint...	Safe Email	sun aug wintermute mentioned impression get re...
7	7	entourage , stockmogul newsletter ralph velez ...	Phishing Email	entourage stockmogul newsletter ralph velez ge...
8	8	we owe you lots of money dear applicant , afte...	Phishing Email	owe lot money dear applicant review upon recei...
9	9	re : coastal deal - with exxon participation u...	Safe Email	coastal deal exxon participation project agree...
10	10	make her beg you to give it to her everynight ...	Phishing Email	make beg give everynight please partner much b...
11	11	URL: http://www.newsisfree.com/click/-5,830431...	Safe Email	url date img cbc
12	12	begin forwarded text Date: Wed, 25 Sep 2002 13...	Safe Email	begin forwarded text date wed sep digital bear...
13	13	re : fyi - wellhead portfolio who is considere...	Safe Email	fyi wellhead portfolio considered greg sharp o...
14	14	rmmla / ads * * * * * * * * papers solicited f...	Safe Email	rmmla ad paper solicited rmmla conference conj...
15	15	re : testing ir & fx var nick and winston , i ...	Safe Email	testing ir fx var nick winston understand ir f...
16	16	The academic discipline of Software Engineerin...	Safe Email	academic discipline software engineering launc...
17	17	re : 3 . 402 queries : language - speakers , s...	Safe Email	query language speaker syntax text would like ...
18	18	a resume john , this is a resume i received to...	Safe Email	resume john resume received today friend pleas...
19	19	EFFector Vol. 15, No. 35 November ...	Safe Email	effector vol november ren eff orga publication...

Gambar 1.7 Hasil Pembersihan Teks

Berikut adalah 20 data yang ditampilkan setelah pembersihan teks.

BAB II
REPRESENTASI TEKS

2.1 Tokenisasi

2.1.1 Tokenisasi TF-IDF

Setelah data ulasan telah dibersihkan selanjutnya masuk pada tahap Tokenizing. Proses Tokenizing merupakan proses yang dilakukan untuk memotong-motong kalimat atau teks menjadi kata perkata atau token.

```
# Tokenisasi Teks
df['tokens'] = df['clean_content'].apply(lambda x: x.split())
print('Hasil Tokenizing : \n')
df.head()
```

Tabel 2.1 Tokenisasi TF-IDF

Sebelum Tokenisasi	Sesudah Tokenisasi
side galicismo galicismo spanish term name improp introduct french word spanish sound thu decept ear galicismo often consid barbarismo would term design opposit phenomenon unlaw word spanish origin may crept french someon provid exampl thank joseph kozono kozonoj gunet georgetown edu	[side,galicismo,galicismo,spanish,term,name,improp,introduct,french,word,spanish,sound,thu,decept,ear,galicismo,often,consid,barbarismo,would,term,design,opposit,phenomenon,unlaw,word,spanish,origin,may,crept,french,someon,provid,exampl,thank,joseph,kozono,kozonoj,gunet,georgetown,edu]

2.1.2 Mengubah String menjadi List (Untuk Metode TF-IDF)

```
# Fungsi untuk menggabungkan list token menjadi string
def join_text_list(tokens):
    # Pastikan tokens adalah list
    if isinstance(tokens, list):
        return ' '.join(tokens)
    return ''

# Mengaplikasikan fungsi ke kolom 'tokens'
df["list"] = df["tokens"].apply(join_text_list)
df.head()
```

Mengubah string menjadi list dalam konteks TF-IDF adalah langkah dasar dalam mempersiapkan teks untuk analisis lebih lanjut. Tokenisasi, perhitungan frekuensi kata, dan penghitung IDF memerlukan teks dalam bentuk list agar dapat melakukan perhitungan yang diperlukan untuk analisis teks secara efisien.

Tabel 2.2 Perubahan String menjadi List

Sebelum Perubahan	Sesudah Perubahan
[side,galicismo,galicismo,spanish,term,name,improp,introduct,french,word,spanish,sound,thu,decept,ear,galicismo,often,consid,barbarismo,would,term,design,opposit,phenomenon,unlaw,word,spanish,origin,may,crept,french,someon,provid,exampl,thank,joseph,kozono,kozonoj,gunet,georgetown,edu]	side galicismo galicismo spanish term name improp introduct french word spanish sound thu decept ear galicismo often consid barbarismo would term design opposit phenomenon unlaw word spanish origin may crept french someon provid exampl thank joseph kozono kozonoj gunet georgetown edu

2.1.3 Tokenisasi Word2Vec

```
# Tokenisasi
from gensim.utils import tokenize
df["tokenize_text"] = df.clean_content.apply(lambda x:
list(tokenize(x)))

df.head(20)
```

Tabel 2.3 Tokenisasi Word2Vec

Sebelum Perubahan	Sesudah Perubahan
url date img cbc	[url,date,img,cbc]

2.2 Metode TF-IDF

```
# Ekstraksi Fitur dengan TF-IDF
vectorizer=TfidfVectorizer(strip_accents='unicode',
                           analyzer='word', ngram_range=(1, 2),
                           max_features=5000, smooth_idf=True,
                           sublinear_tf=True)

# Fit vectorizer untuk mempelajari kosakata dari teks di kolom 'list'
vectorizer.fit(df["list"])

# Transformasi teks menjadi representasi numerik (TF-IDF)
X = vectorizer.transform(df["list"]).toarray()
```

Berikut penjelasan singkat mengenai kode yang diberikan:

1. **Inisialisasi TfidfVectorizer:**
    - **strip_accents='unicode'**: Menghapus aksen pada karakter teks.

- **analyzer='word'**: Menggunakan kata (bukan karakter atau n-gram) untuk analisis.
- **ngram_range=(1, 2)**: Mempertimbangkan unigram (kata tunggal) dan bigram (dua kata berturut-turut) sebagai fitur.
- **max_features=5000**: Hanya mempertimbangkan 5000 kata dengan TF-IDF tertinggi.
- **smooth_idf=True**: Menghaluskan nilai IDF untuk mencegah pembagian dengan nol.
- **sublinear_tf=True**: Menggunakan logaritma pada perhitungan frekuensi kata untuk stabilitas.

2. **vectorizer.fit(df["list"])**: Mempelajari kosakata dari teks yang ada di kolom list pada Data Frame df.
3. **vectorizer.transform(df["list"]).toarray()**: Mengubah teks dalam kolom list menjadi representasi numerik (vektor TF-IDF) dan mengubah hasilnya menjadi array.

Hasil:

- **X**: Matriks berisi representasi TF-IDF dari teks dalam kolom list. Setiap baris merepresentasikan dokumen (teks) dan setiap kolom mewakili kata atau bigram yang paling relevan berdasarkan nilai TF-IDF.

2.2.1 Ranking TF-IDF

```
terms = vectorizer.get_feature_names_out()
# Menjumlahkan frekuensi tf-idf setiap istilah melalui dokumen
sums = X.sum(axis=0)
# Menghubungkan istilah dengan frekuensi jumlahnya
data = []
for col, term in enumerate(terms):
    data.append((term, sums[col]))
# Membuat DataFrame untuk menampilkan istilah dan peringkatnya
ranking = pd.DataFrame(data, columns=['term','rank'])

# Mengurutkan berdasarkan peringkat (rank) secara menurun
```

```
ranking.sort_values('rank', ascending=False)
```

	term	rank
1359	empti	540.931444
789	com	364.043642
3322	pleas	361.776904
1808	get	338.021288
2511	list	335.260891
...	...	...
1373	enenkio	0.331590
4603	tvsr	0.307719
2341	kingdom enenkio	0.299444
1589	fax subject	0.029756
2023	html subject	0.028703

5000 rows × 2 columns

Gambar 2.1 Ranking TF-IDF

jika kita lihat, top 5000 TF-IDF menggunakan Scikit-Learn `TfidfVectorizer()` menghasilkan result berbeda dengan hasil yang kita buat sebelumnya, ini seperti dibahas diatas bahwa pada `TfidfVectorizer()` mengimplementasikan formula yang sedikit berbeda dengan standar text book, dan menerapkan normalisasi L2 pada sparse matrix yang terbentuk.

2.2.2 Menghitung TF

```
def calc_TF(document):
    # Counts the number of times the word appears in review
    TF_dict = {}
    for term in document:
        if term in TF_dict:
            TF_dict[term] += 1
        else:
```

```
            TF_dict[term] = 1
    # Computes tf for each word
    for term in TF_dict:
        TF_dict[term] = TF_dict[term] / len(document)
    return TF_dict

df["TF_dict"] = df['tokens'].apply(calc_TF)

df["TF_dict"].head()
```

```
# Mengecek Hasil TF
index = 1

print('%20s' % "term", "\t", "TF\n")
for key in df["TF_dict"][index]:
    print('%20s' % key, "\t", df["TF_dict"][index][key])
```

Untuk menghitung manual dari TF adalah:

$$tf_{ij} = \frac{f_d(i)}{\max_{j \in d} f_d(j)}$$

Gambar 2.2 Rumus TF

Contohnya:

Untuk kata galicismo terdapat 3 kata dari 42 kata yang tersedia di index

TF = 3/42 = 0.07317073170731707

Tabel 2.3 Hasil TF dari Index 1

term	TF
side	0.024390243902439025
galicismo	0.07317073170731707
spanish	0.07317073170731707
term	0.04878048780487805

2.2.3 Menghitung IDF

```
def calc_DF(tfDict):
    count_DF = {}
```

```
    # Run through each document's tf dictionary and increment
countDict's (term, doc) pair
    for document in tfDict:
        for term in document:
            if term in count_DF:
                count_DF[term] += 1
            else:
                count_DF[term] = 1
    return count_DF

DF = calc_DF(df["TF_dict"])

# Misal n_document adalah jumlah total dokumen
n_document = len(df['tokens'])

# Fungsi untuk menghitung IDF
def calc_IDF(n_document, DF):
    IDF_Dict = {}
    for term in DF:
        # Rumus IDF: log(N / (DF(t) + 1))
        IDF_Dict[term] = np.log(n_document / (DF[term] + 1))
    return IDF_Dict

# Menghitung IDF berdasarkan Document Frequency (DF)
IDF = calc_IDF(n_document, DF)

# Menampilkan IDF yang dihitung
print(IDF)
```

Untuk menghitung manual IDF adalah:

$$\text{IDF} = Log\ \frac{(Jumlah\ dokumen)}{jumlah\ dokument\ yang\ megandung\ kata\ \ tsb}$$

Gambar 2.3 Rumus IDF

Tabel 2.4 Hasil IDF

term	IDF
side	3.5674419352726603
galicismo	8.73413085934226
spanish	4.452845794469911
term	2.6883359676892313

2.2.4 Menghitung TF-IDF

```
#calc TF-IDF
def calc_TF_IDF(TF):
    TF_IDF_Dict = {}
    #For each word in the review, we multiply its tf and its idf.
    for key in TF:
        TF_IDF_Dict[key] = TF[key] * IDF[key]
    return TF_IDF_Dict

#Stores the TF-IDF Series
df["TF-IDF_dict"] = df["TF_dict"].apply(calc_TF_IDF)

# Check TF-IDF result
index = 1

print('%20s' % "term", "\t", '%10s' % "TF", "\t", '%20s' % "TF-IDF\n")
for key in df["TF-IDF_dict"][index]:
    print('%20s' % key, "\t", df["TF_dict"][index][key] ,"\t" , df["TF-
IDF_dict"][index][key])
```

Untuk menghitung manual dari TF-IDF adalah:

TF-IDF = TF * IDF

Contohnya:

Untuk kata galicismo dengan cara berikut:

TF-IDF = `0.07317073170731707 * 8.73413085934226 = 0.6390827458055311`

2.3 Metode Word2Vec

```
from gensim.models import Word2Vec

# define the model
```

```
model_wikihow = Word2Vec(
    window=10,
    min_count=5,
    workers=4,
    epochs=10,
)
def vectorize_text(tokens, model):
    vectors = [model.wv[word] for word in tokens if word in model.wv]
    if vectors:
        return np.mean(vectors, axis=0)
    else:
        return np.zeros(model.vector_size)

X = np.array([vectorize_text(text, model_wikihow) for text in
df.tokenize_text])
```

Berikut penjelasan singkat mengenai kode yang diberikan:

- **Inisialisasi Word2Vec:**
  - **Word2Vec**: Model untuk menghasilkan representasi kata (word embeddings).
  - **window=10**: Menentukan ukuran konteks kata sekitar 10 kata.
  - **min_count=5**: Mengabaikan kata yang muncul kurang dari 5 kali.
  - **workers=4**: Menggunakan 4 thread untuk mempercepat pelatihan.
  - **epochs=10**: Jumlah iterasi model selama pelatihan.

```
model_wikihow.build_vocab(df.tokenize_text, progress_per=1000)

# train the model

model_wikihow.train(df.tokenize_text,
total_examples=model_wikihow.corpus_count, epochs=model_wikihow.epochs)

model_wikihow_path = r'/content/drive/MyDrive/Dataset/word2vec.model'

# save the model

model_wikihow.save(model_wikihow_path)

# load the model

model_wikihow = Word2Vec.load(model_wikihow_path)
```

Berikut adalah penjelasannya:

1. **model_wikihow.build_vocab(df.tokenize_text, progress_per=1000)**: Membangun vocabulary dari data teks (df.tokenize_text) dengan menampilkan progres setiap 1000 token.
2. **model_wikihow.train(...)**: Melatih model menggunakan teks yang sudah diproses (df.tokenize_text), dengan jumlah contoh dan epoch yang ditentukan.
3. **model_wikihow.save(model_wikihow_path)**: Menyimpan model yang sudah dilatih ke path yang ditentukan.
4. **model_wikihow = Word2Vec.load(model_wikihow_path)**: Memuat kembali model yang disimpan dari path tersebut.

```
model_wikihow.wv.most_similar("syntax")
```

```
[('morpholog', 0.8133170008659363),
 ('semant', 0.7704978585243225),
 ('phonolog', 0.7702600359916687),
 ('syntact', 0.7652926445007324),
 ('pragmat', 0.7010563611984253),
 ('cowper', 0.6978756785392761),
 ('semit', 0.6801329851150513),
 ('cascio', 0.6778619289398193),
 ('antisymmetri', 0.6759095788002014),
 ('grammar', 0.6746140718460083)]
```

Gambar 2.4 Similarity

Perlu diingat juga bahwa Word2Vec membangun vektor yang merepresentasikan tiap kata berdasarkan rangkaian urutan kemunculan tiap dalam teks. Dengan demikian, kata-kata yang sering disebutkan secara bersamaan akan memiliki vektor yang hampir mirip. Oleh karena itu, kata-kata seperti “morpholog”, dan “semant” muncul sebagai kata yang mirip dengan “syntax”.

2.3.1 Visualisasi Word2Vec dengan PCA

```
words = ['test', 'portfolio', 'date', 'make', 'attribut', 'son',
'like', 'side', 'slice', 'ticket', 'deal', 'assist', 'one', 'dream',
'so', 'equistar' , 'low' , 'beg' , 'still', 'avail' , 'begin', 'robert'
, 'enter' , 'new' , 'who', 'disciplin' , 'forward' , 'talk' , 'bryan' ,
'hull' , 'anita' , 'luong']
```

Setiap kata-kata di list *words* diubah dalam vektor menggunakan model Wikihow.

```
# Store the words vector in list
```

```
word_vector = []

for word in words:
  word_vector.append(model_wikihow.wv.get_vector(word))

word_vector[0]
```

Vektor dengan ukuran n=100 tersebut akan ditransformasikan menjadi vektor n=2 menggunakan API *PCA* dari library *Scikit Learn*.

```
from sklearn.decomposition import PCA

# transforming words vector
pca = PCA(n_components=2)
result = pca.fit_transform(word_vector)
```

```
import plotly.express as px

# Create the scatter plot
fig = px.scatter(x=result[:, 0], y=result[:, 1])

# Annotate each data point
for i, txt in enumerate(words):
    fig.add_annotation(
        x=result[i, 0],
        y=result[i, 1],
        text=txt,
        xref="x",
        yref="y",
        showarrow=True,
        arrowhead=7,
        ax=20,
        ay=-30
    )

# adjust graph size
fig.update_layout(
    height=1200,
    width=1200,
    title='Visualisasi Kata-kata Bahasa Inggris'
)

# Show the plot
```

```
fig.show()
```

Gambar 2.5 Visualisasi dengan PCA

Berikut adalah hasil visualisasi menggunakan Word2Vec.

2.4 Perbandingan Kelebihan dan Kekurangan TF-IDF dan Word2Vec

2.4.1 Kelebihan dan Kekurangan TF-IDF

Keuntungan:

- TF-IDF dan Model Ruang Vektor menangkap pentingnya semantik istilah dalam dokumen, yang dapat menghasilkan hasil pencarian yang lebih relevan.
- Dengan mempertimbangkan bobot istilah, Model Ruang Vektor memberikan representasi konten dokumen yang lebih bernuansa dibandingkan dengan pencocokan kata kunci sederhana.

- Kesamaan kosinus yang digunakan dalam Model Ruang Vektor memungkinkan pencocokan parsial, yang dapat lebih memaafkan dan membantu dalam mengambil dokumen relevan bahkan jika istilah kueri tidak cocok secara tepat.

Kekurangan:

- TF-IDF dan Vector Space Model dapat membutuhkan komputasi yang lebih mahal dibandingkan dengan metode pencarian kata kunci sederhana, karena keduanya memerlukan perhitungan bobot istilah dan kesamaan antara vektor.
- Kinerja TF-IDF dan Vector Space Model mungkin terbatas dalam kasus di mana kumpulan dokumen memiliki fokus yang sangat spesifik atau sempit, sehingga bobot istilah kurang informatif.

2.4.2 Kelebihan dan Kekurangan Word2Vec

Kelebihan:

- Efisien dalam memori: Memproses dataset besar dengan efisien.
- Kemudahan penggunaan: API yang sederhana dan dokumentasi lengkap.
- Kualitas representasi kata: Menyediakan vektor kata yang kaya makna semantik.
- Dukungan model: Mendukung CBOW dan Skip-gram.
- Skalabilitas: Dapat menangani data dalam skala besar.

Kekurangan:

- Ketergantungan pada data besar: Membutuhkan banyak data untuk menghasilkan representasi yang baik.
- Kurang memperhatikan konteks kalimat: Tidak menangkap urutan kata dalam kalimat (konten sekuensial).
- Tuning model yang kompleks: Menyesuaikan parameter seperti ukuran jendela dan dimensi vektor bisa sulit.
- Masalah dengan kata langka: Kata yang jarang muncul sering kali menghasilkan representasi yang buruk.

BAB III

PEMODELAN

3.1 Pembagian Train Test Split dan Label Encoder

3.1.1 Pembagian Train Test Split

```
# Melihat dimensi dari X
X.shape

# Pembagian Train Test Split (80:20)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Pembagian Train Test Split (70:30)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Pembagian Train Test Split (60:40)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

- **X.shape**: Menampilkan dimensi atau ukuran dari matriks **X**, yang berisi representasi numerik (TF-IDF) dari teks di kolom list (dokumen). Ini akan memberikan informasi tentang jumlah baris (dokumen) dan jumlah kolom (fitur atau kata) dalam matriks TF-IDF.
- Pembagian dataset ini dapat dilakukan dengan menggunakan modul atau fungsi dari library python yaitu modul train_test_split dari library Scikit-Learn. Ketiga skenario dijalankan pada notebook. Perubahan yang dilakukan pada pembagian dataset hanya pada bagian test_size (0.20 untuk pembagian 80:20 dan 0.30 untuk pembagian 70:30).

Tabel 3.1 Pembagian Data Uji dan Data Latih

Skenario	Rasio Pembagian Data Uji : Data Latih	Jumlah Data Latih	Jumlah Data Test
1	80 : 20	14907	3727
2	70 : 30	13043	5591

3.1.2 Label Encoder

```
# Melakukan Label Encoder
```

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['Email Type'])
```

- **LabelEncoder()**: Ini adalah kelas dari sklearn.preprocessing yang digunakan untuk mengonversi kategori teks (seperti label kelas) menjadi angka.
- **label_encoder.fit_transform(df['Email Type'])**: Fungsi ini akan mengubah kolom Email Type menjadi nilai numerik. Setiap kategori akan diberi label angka yang unik. Misalnya, jika kolom tersebut berisi "Safe Email" dan "Phising Email", LabelEncoder akan mengubahnya menjadi 0 dan 1.
- **y**: Hasil dari encoding ini disimpan dalam variabel y, yang sekarang berisi label numerik yang sesuai dengan kategori di df['Email Type'].

3.2 Model KNN dengan TF-IDF

```
def train_knn_models(X_train, y_train, k_values):
    knn_models = []
    for k in k_values:
        # Melatih KNN dengan nilai k tertentu
        knn_classifier = KNeighborsClassifier(n_neighbors=k,
weights='distance', metric='minkowski')
        knn_classifier.fit(X_train, y_train)
        knn_models.append((f'KNN (k={k})', knn_classifier))  #
Menyimpan model dan nilai k-nya
    return knn_models

# Melatih model KNN untuk berbagai nilai k
k_values = [3, 5]
knn_models = train_knn_models(X_train, y_train, k_values)
```

Berikut adalah fungsi dari pembuatan model KNN. Penjelasan sebagai berikut:

Input:

- **X_train:** Fitur dari data pelatihan.
- **y_train:** Label dari data pelatihan.
- **k_values:** Daftar nilai k yang akan diuji dalam model KNN. K yang dipakai ialah 3,5

Untuk setiap k, buat dan latih model KNN dengan parameter:

- **n_neighbors=k**: Menentukan jumlah tetangga terdekat yang digunakan.
- **weights='distance'**: Memberikan bobot berdasarkan jarak untuk tetangga terdekat.
- **metric='minkowski'**: Menggunakan jarak Minkowski (standar untuk KNN).

3.3 Evaluasi Model KNN dengan TF-IDF

```
# Fungsi untuk mengevaluasi model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)

    # Pastikan label yang ada di y_test yang digunakan dalam
confusion_matrix
    labels = np.unique(y_test)

    cm = confusion_matrix(y_test, y_pred, labels=labels)
    tn, fp, fn, tp = cm.ravel()

    return {
        'Confusion Matrix': cm,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred,
pos_label=labels[1]),
        'Recall': recall_score(y_test, y_pred, pos_label=labels[1]),
        'F1-Score': f1_score(y_test, y_pred, pos_label=labels[1])
    }

# Fungsi untuk mengevaluasi model KNN yang sudah dilatih
def evaluate_knn_models(knn_models, X_test, y_test):
    knn_results = []
    for model_name, model in knn_models:
        metrics = evaluate_model(model, X_test, y_test)
        knn_results.append({
            'Model': model_name,
            'Accuracy': metrics['Accuracy'],
            'Precision': metrics['Precision'],
            'Recall': metrics['Recall'],
            'F1-Score': metrics['F1-Score']
        })
    return knn_results

# Mengevaluasi model yang sudah dilatih
knn_results = evaluate_knn_models(knn_models, X_test, y_test)

# Buat DataFrame untuk meringkas hasil
summary_df = pd.DataFrame(knn_results)

# Tampilkan hasil
```

```
print(summary_df)
```

Berikut adalah penjelasan singkat dan rumus untuk masing-masing metrik evaluasi:

1. Akurasi (Accuracy):

- **Definisi**: Mengukur proporsi prediksi yang benar dibandingkan dengan total prediksi.
- **Rumus**:
  - $$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$$

Gambar 3.1 Rumus Menghitung Akurasi

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **TN**: True Negative (Prediksi benar untuk kelas negatif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas negatif)

2. Precision

- **Definisi**: Mengukur seberapa banyak prediksi positif yang benar-benar positif.
- **Rumus**:
  - $$\text{Precision} = \frac{TP}{TP + FP}$$

Gambar 3.2 Rumus Perhitungan Precision

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)

3. Recall (Sensitivitas)

- **Definisi**: Mengukur seberapa banyak kasus positif yang berhasil dikenali oleh model.
- **Rumus**:

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

Gambar 3.3 Rumus Perhitungan Recall

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas positif)

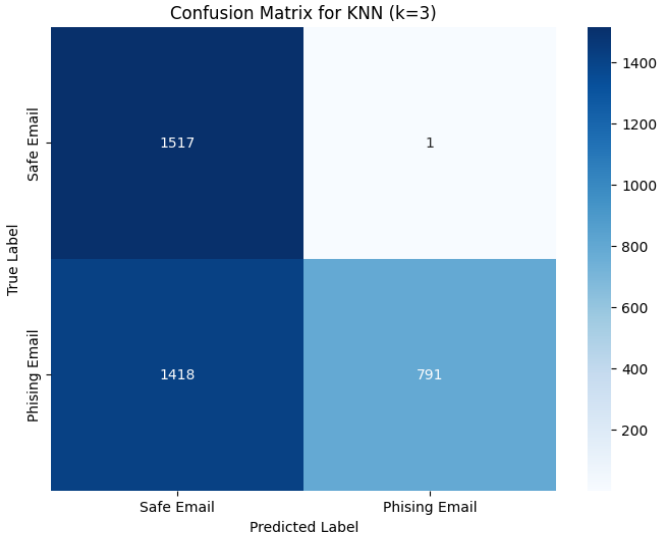
4. F1-Score

- **Definisi**: Rata-rata harmonis antara Precision dan Recall, memberikan keseimbangan keduanya.
- **Rumus**:
  - $$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gambar 3.4 Rumus Perhitungan F1-Score

3.3.1 Skenario 1 (Rasio Pembagian 80:20) - Evaluasi Model KNN dengan TF-IDF

Skenario pertama pengujian dilakukan pada model yang telah dilatih dengan 80% dari jumlah dataset. Dengan jumlah data uji sebanyak 3727 data didapatkan hasil prediksi sebagai berikut:



- **KNN (k = 3)**
    - True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 1517
    - True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 791
    - False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 1418
    - False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 1

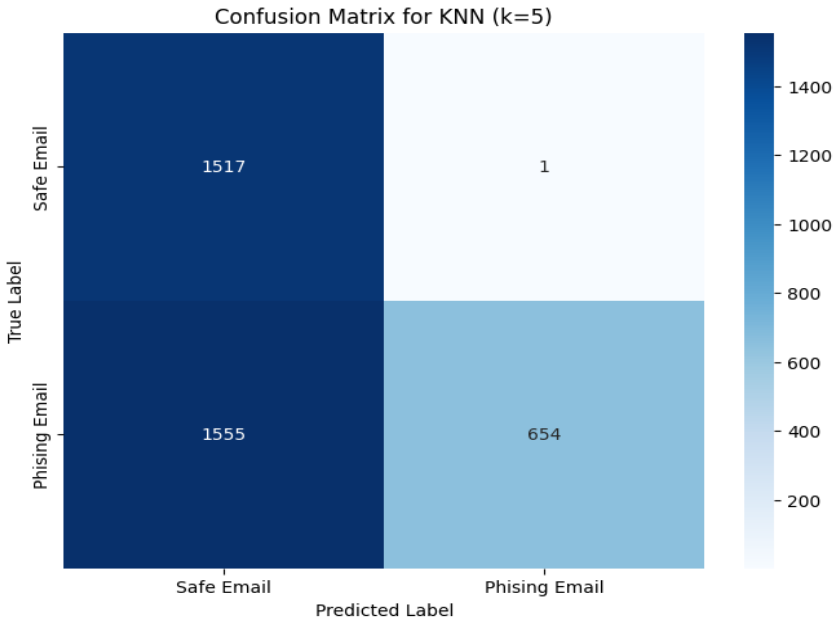
Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{791+1517}{791+1517+1+1416} = \frac{2308}{3727} = 0.619265$

- **Precision** = $\frac{791}{791+1} = \frac{791}{792} = 0.998737$

- **Recall** = $\frac{791}{791+1418} = \frac{791}{2208} = 0.3580$

- **F1-Score** = 2 x $\frac{0.998737\ x\ 0.3580}{0.998737+0.3580} = 2x\frac{0.357547}{1.356737} = 0.527158$



- **KNN (k=5)**

- True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 1517
- True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 654
- False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 1555
- False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 1

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi =** $\frac{654+1517}{654+1517+1+1555} = \frac{2171}{3727} = 0.582506$

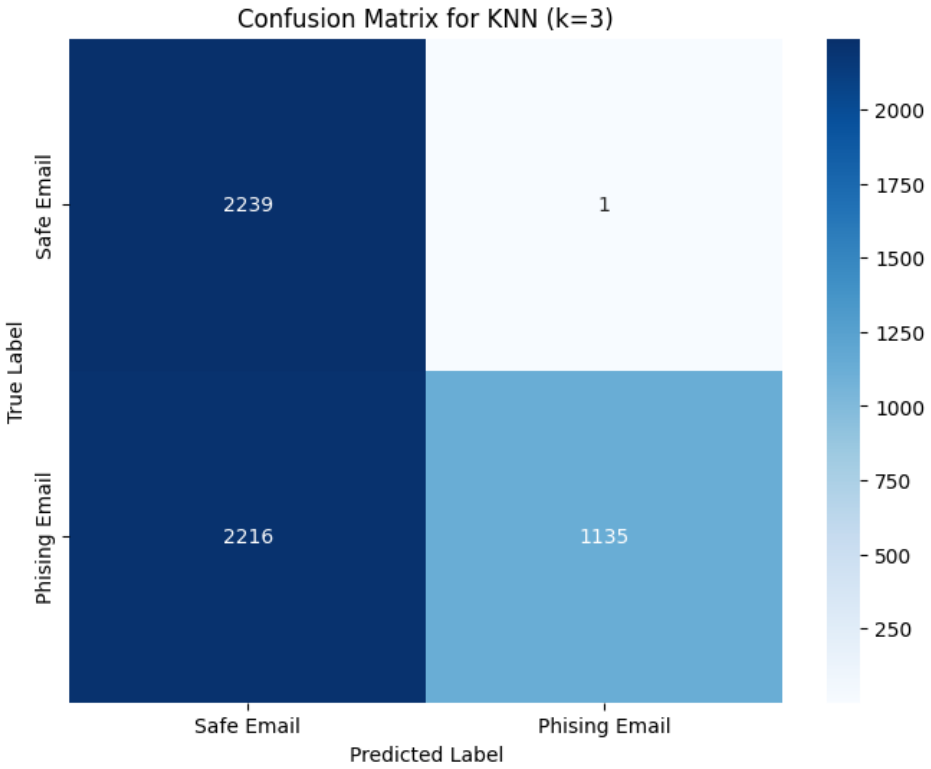
- **Precision =** $\frac{654}{654+1} = \frac{654}{655} = 0.998473$

- **Recall =** $\frac{654}{654+1555} = \frac{654}{2209} = 0.296062$

- **F1-Score = 2 x** $\frac{0.998473x\ 0.296062}{0.998473+0.296062} = 2x\frac{0.295609}{1.294535} = 0.456704$

3.3.2 Skenario 2 (Rasio Pembagian 70:30) - Evaluasi Model KNN dengan TF-IDF

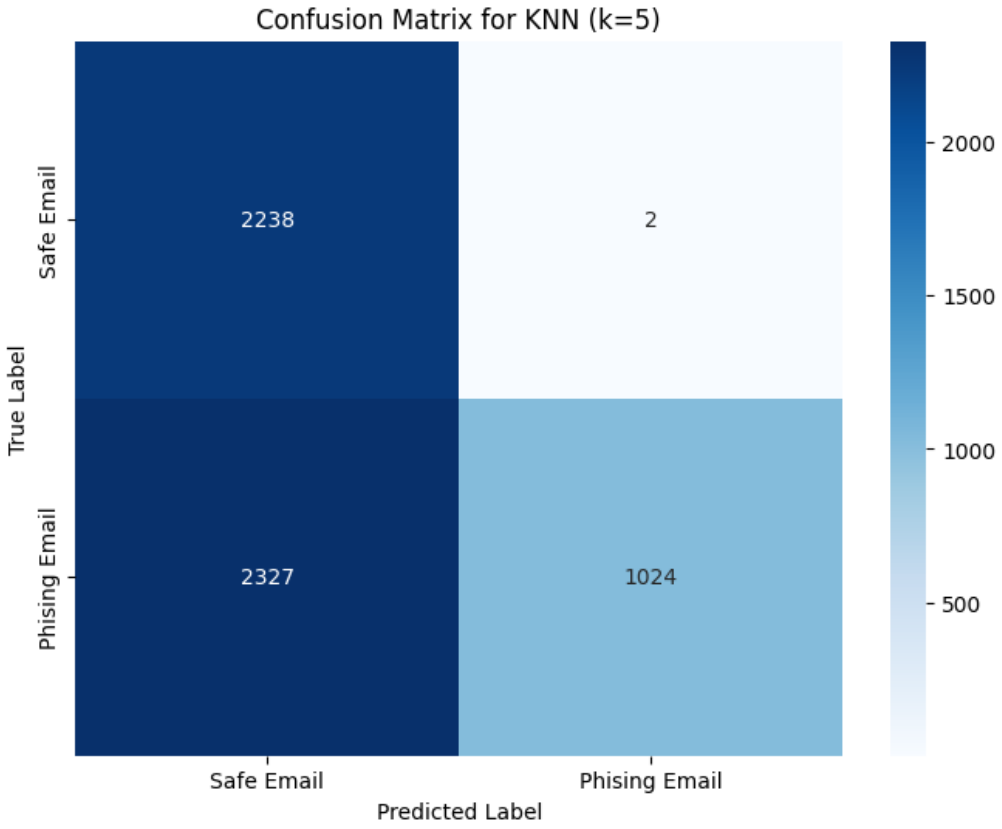
Skenario kedua pengujian dilakukan pada model yang telah dilatih dengan 70% dari jumlah dataset. Dengan jumlah data uji sebanyak 5591 data didapatkan hasil prediksi sebagai berikut:



- **KNN (k = 3)**
  - True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 2239
  - True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 1135
  - False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 2216
  - False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 1

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{1135+2239}{1135+2239+1+2216} = \frac{3374}{5591} = 0.603470$
- **Precision** = $\frac{1135}{1135+1} = \frac{1135}{1136} = 0.999120$
- **Recall** = $\frac{1135}{1135+2216} = \frac{1135}{3351} = 0.338705$
- **F1-Score** = 2 x $\frac{0.999120x\ 0.338705}{0.999120+0.338705} = 2x\frac{0.338401}{1.337825} = 0.505906$



- **KNN (k = 5)**
  - True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 2238
  - True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 1024
  - False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 2327
  - False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 2

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{1024+2238}{1024+2238+2+2327} = \frac{3262}{5591} = 0.583438$
- **Precision** = $\frac{1024}{1024+2} = \frac{1024}{1026} = 0.998051$
- **Recall** = $\frac{1024}{1024+2327} = \frac{1024}{3351} = 0.305580$

- **F1-Score = 2 x** $\frac{0.998051\ x\ 0.305580}{0.998051 + 0.305580} = 2x\frac{0.304984}{1.303631} =$ **0.467900**

3.4 Model Naive Bayes dengan TF-IDF

```
# Evaluasi Naive Bayes
nb_results = []

# Inisialisasi Naive Bayes (GaussianNB)
nb_classifier = GaussianNB(var_smoothing=1e-9)

# Convert X_train and X_test to dense arrays
X_train_dense = X_train  # Assuming X_train is a sparse matrix
X_test_dense = X_test    # Assuming X_test is a sparse matrix

# Melatih model Naive Bayes
nb_classifier.fit(X_train_dense, y_train)

# Menghitung metrik evaluasi
metrics = evaluate_model(nb_classifier, X_test_dense, y_test)
```

Berikut penjelasan pembuatan Naive Bayes:

- **GaussianNB()**: Menginisialisasi model Naive Bayes berbasis distribusi Gaussian, yang mengasumsikan bahwa fitur dalam dataset memiliki distribusi normal.
- **var_smoothing=1e-9**: Menambahkan sedikit nilai kecil (1e-9) untuk menghindari pembagian dengan 0 pada variansi, yang membantu meningkatkan kestabilan perhitungan probabilitas.

3.5 Evaluasi Model Naive Bayes dengan TF-IDF

```
# Fungsi untuk mengevaluasi model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)

    # Pastikan label yang ada di y_test yang digunakan dalam
confusion_matrix
    labels = np.unique(y_test)  # Menyesuaikan label dengan yang ada di
y_test

    cm = confusion_matrix(y_test, y_pred, labels=labels)
    tn, fp, fn, tp = cm.ravel()

    return {
```

```
        'Confusion Matrix': cm,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred,
pos_label=labels[1]),
        'Recall': recall_score(y_test, y_pred, pos_label=labels[1]),
        'F1-Score': f1_score(y_test, y_pred, pos_label=labels[1])
    }

# Menyimpan hasil evaluasi
nb_results.append({
    'Model': 'Naive Bayes',
    'Accuracy': metrics['Accuracy'],
    'Precision': metrics['Precision'],
    'Recall': metrics['Recall'],
    'F1-Score': metrics['F1-Score']
})

# Buat DataFrame untuk meringkas hasil Naive Bayes
nb_summary_df = pd.DataFrame(nb_results)

# Tampilkan hasil
print(nb_summary_df)
```

Berikut adalah penjelasan singkat dan rumus untuk masing-masing metrik evaluasi:

1. Akurasi (Accuracy):

- **Definisi**: Mengukur proporsi prediksi yang benar dibandingkan dengan total prediksi.
- **Rumus**:
- $$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$$

Gambar 3.5 Rumus Menghitung Akurasi

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **TN**: True Negative (Prediksi benar untuk kelas negatif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas negatif)

2. Precision

- **Definisi**: Mengukur seberapa banyak prediksi positif yang benar-benar positif.
- **Rumus**:
  - $$\text{Precision} = \frac{TP}{TP + FP}$$

Gambar 3.6 Rumus Perhitungan Precision

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)

3. Recall (Sensitivitas)

- **Definisi**: Mengukur seberapa banyak kasus positif yang berhasil dikenali oleh model.
- **Rumus**:
  - $$\text{Recall} = \frac{TP}{TP + FN}$$

Gambar 3.7 Rumus Perhitungan Recall

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas positif)

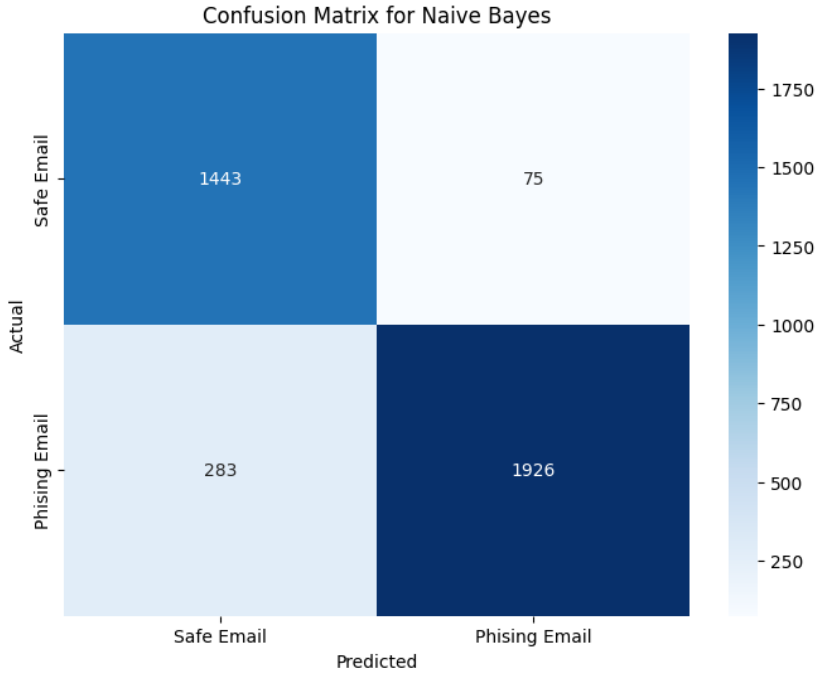
4. F1-Score

- **Definisi**: Rata-rata harmonis antara Precision dan Recall, memberikan keseimbangan keduanya.
- **Rumus**:
  - $$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gambar 3.8 Rumus Perhitungan F1-Score

3.5.1 Skenario 1 (Rasio Pembagian 80:20) - Evaluasi Model Naive Bayes dengan TF-IDF

Skenario pertama pengujian dilakukan pada model yang telah dilatih dengan 80% dari jumlah dataset. Dengan jumlah data uji sebanyak 3727 data didapatkan hasil prediksi sebagai berikut:



1. True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 1433
2. True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 1926
3. False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 283
4. False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 75

- **Akurasi** = $\frac{1926+1433}{1926+1433+75+283} = \frac{3359}{3727} = 0.903944$

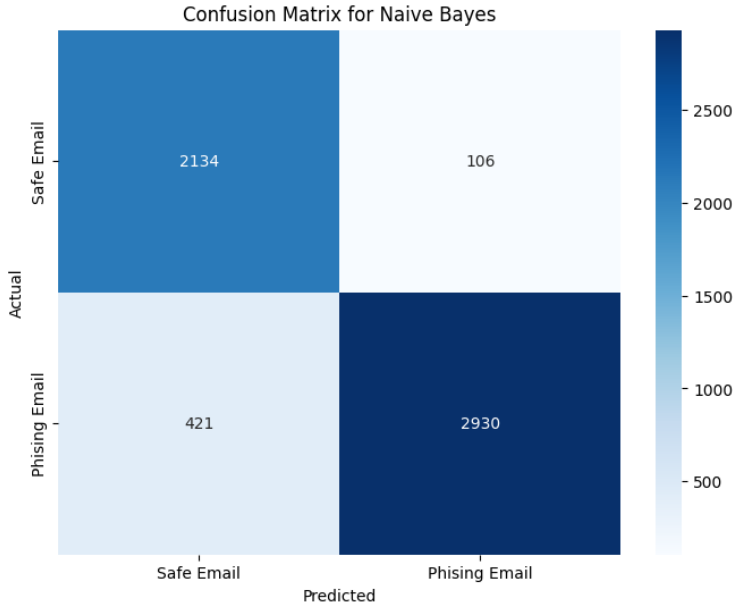
- **Precision** = $\frac{1926}{1926+75} = \frac{1926}{2001} = 0.962519$

- **Recall** = $\frac{1926}{1926+283} = \frac{1926}{2209} = 0.871888$

- **F1-Score = 2 x** $\frac{0.962519\ x\ 0.871888}{0.962519+\ 0.871888} = 2x\frac{0.839208}{1.8581399} = 0.914964$

3.5.2 Skenario 2 (Rasio Pembagian 70:30) - Evaluasi Model KNN dengan TF-IDF

Skenario kedua pengujian dilakukan pada model yang telah dilatih dengan 70% dari jumlah dataset. Dengan jumlah data uji sebanyak 5591 data didapatkan hasil prediksi sebagai berikut:



1. True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 2134
2. True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 2930
3. False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 421
4. False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 106

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi =** $\frac{2930+2134}{2930+2134+106+421} = \frac{5073}{5591} = 0.905741$

- **Precision =** $\frac{2930}{2930+106} = \frac{2930}{3036} = 0.965086$

- **Recall** = $\frac{2930}{2930+421} = \frac{2930}{3351} = 0.874366$

- **F1-Score** = 2 x $\frac{0.965086x\ 0.874366}{0.965086+0.874366} = 2x\frac{0.843838}{1.839452} = 0.917489$

3.6 Model KNN dengan Metode Word2Vec

```
def train_knn_models(X_train, y_train, k_values):
    knn_models = []
    for k in k_values:
        # Melatih KNN dengan nilai k tertentu
        knn_classifier = KNeighborsClassifier(n_neighbors=k,
weights='distance', metric='minkowski')
        knn_classifier.fit(X_train, y_train)
        knn_models.append((f'KNN (k={k})', knn_classifier))  #
Menyimpan model dan nilai k-nya
    return knn_models

# Melatih model KNN untuk berbagai nilai k
k_values = [3, 5]
knn_models = train_knn_models(X_train, y_train, k_values)
```

Berikut adalah fungsi dari pembuatan model KNN. Penjelasan sebagai berikut:

Input:

- **X_train:** Fitur dari data pelatihan.
- **y_train:** Label dari data pelatihan.
- **k_values:** Daftar nilai k yang akan diuji dalam model KNN. K yang dipakai ialah 3,5

Untuk setiap k, buat dan latih model KNN dengan parameter:

- **n_neighbors=k**: Menentukan jumlah tetangga terdekat yang digunakan.
- **weights='distance'**: Memberikan bobot berdasarkan jarak untuk tetangga terdekat.
- **metric='minkowski'**: Menggunakan jarak Minkowski (standar untuk KNN).

3.7 Evaluasi KNN dengan Metode Word2Vec

```
# Fungsi untuk mengevaluasi model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)

    # Pastikan label yang ada di y_test yang digunakan dalam
confusion_matrix
```

```
    labels = np.unique(y_test)

    cm = confusion_matrix(y_test, y_pred, labels=labels)
    tn, fp, fn, tp = cm.ravel()

    return {
        'Confusion Matrix': cm,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred,
pos_label=labels[1]),
        'Recall': recall_score(y_test, y_pred, pos_label=labels[1]),
        'F1-Score': f1_score(y_test, y_pred, pos_label=labels[1])
    }

# Fungsi untuk mengevaluasi model KNN yang sudah dilatih
def evaluate_knn_models(knn_models, X_test, y_test):
    knn_results = []
    for model_name, model in knn_models:
        metrics = evaluate_model(model, X_test, y_test)
        knn_results.append({
            'Model': model_name,
            'Accuracy': metrics['Accuracy'],
            'Precision': metrics['Precision'],
            'Recall': metrics['Recall'],
            'F1-Score': metrics['F1-Score']
        })
    return knn_results

# Mengevaluasi model yang sudah dilatih
knn_results = evaluate_knn_models(knn_models, X_test, y_test)

# Buat DataFrame untuk meringkas hasil
summary_df = pd.DataFrame(knn_results)

# Tampilkan hasil
print(summary_df)
```

Berikut adalah penjelasan singkat dan rumus untuk masing-masing metrik evaluasi:

1. Akurasi (Accuracy):

- **Definisi**: Mengukur proporsi prediksi yang benar dibandingkan dengan total prediksi.
- **Rumus**:

- $$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$$

Gambar 3.9 Rumus Menghitung Akurasi

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **TN**: True Negative (Prediksi benar untuk kelas negatif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas negatif)

2. Precision

- **Definisi**: Mengukur seberapa banyak prediksi positif yang benar-benar positif.
- **Rumus**:
  - $$\text{Precision} = \frac{TP}{TP + FP}$$

Gambar 3.10 Rumus Perhitungan Precision

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)

3. Recall (Sensitivitas)

- **Definisi**: Mengukur seberapa banyak kasus positif yang berhasil dikenali oleh model.
- **Rumus**:
  - $$\text{Recall} = \frac{TP}{TP + FN}$$

Gambar 3.11 Rumus Perhitungan Recall

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas positif)

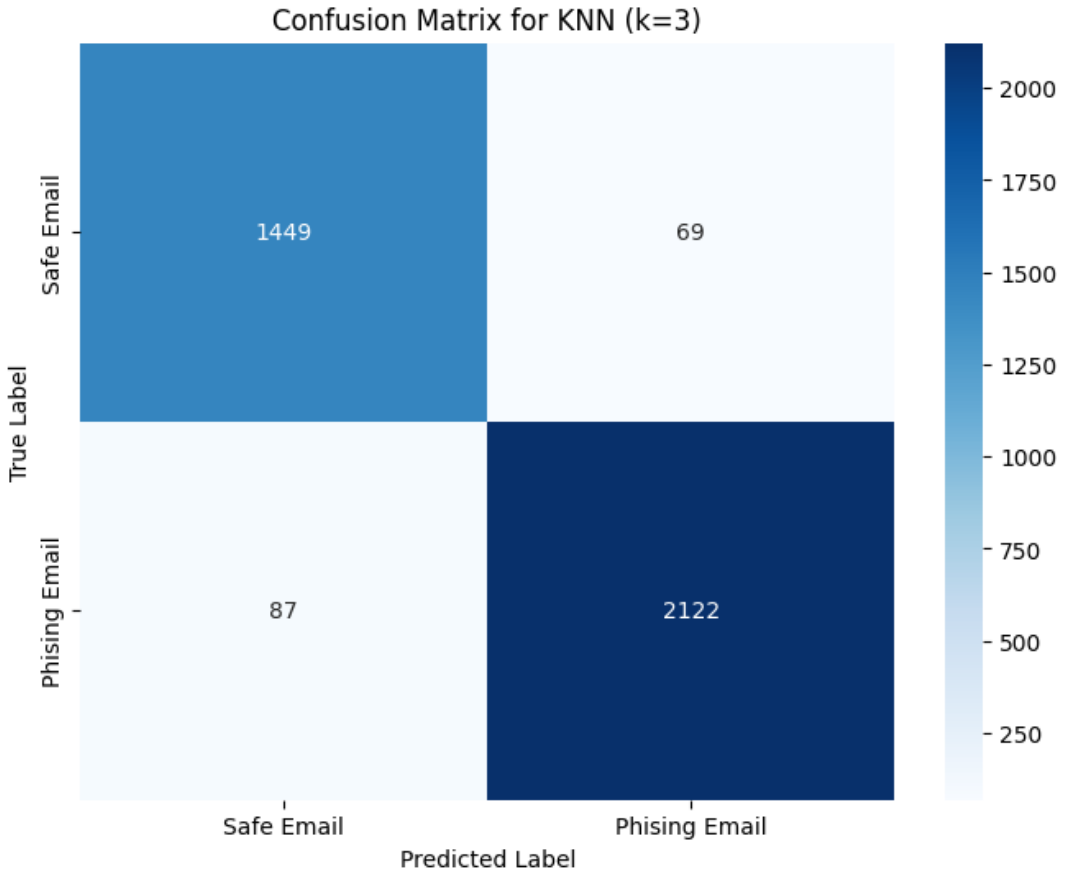
4. F1-Score

- **Definisi**: Rata-rata harmonis antara Precision dan Recall, memberikan keseimbangan keduanya.
- **Rumus**:
- $$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gambar 3.12 Rumus Perhitungan F1-Score

3.7.1 Skenario 1 (Rasio Pembagian 80:20) - Evaluasi Model KNN dengan Word2Vec

Skenario pertama pengujian dilakukan pada model yang telah dilatih dengan 80% dari jumlah dataset. Dengan jumlah data uji sebanyak 3727 data didapatkan hasil prediksi sebagai berikut:



- **KNN (k = 3)**

- True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 1449
- True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 2122
- False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 87
- False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 69

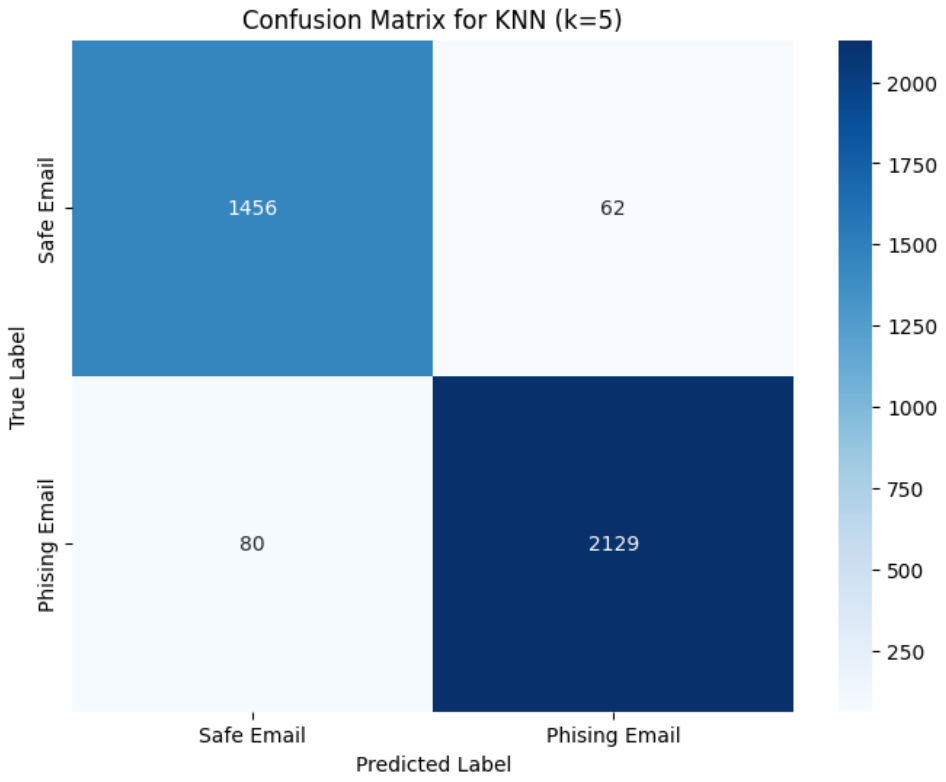
Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{2122+1449}{2122+1449+69+87} = \frac{3571}{3727} = 0.958143$

- **Precision** = $\frac{2122}{2122+69} = \frac{2122}{2191} = 0.968508$

- **Recall** = $\frac{2122}{2122+87} = \frac{2122}{2209} = 0.960616$

- **F1-Score** = 2 x $\frac{0.968508\ x\ 0.960616}{0.968508 + 0.960616} = 2x\frac{0.930364}{1.929124} = 0.964545$



- **KNN (k=5)**
    - True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 1456
    - True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 2129
    - False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 80
    - False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 62

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi =** $\frac{2129+1456}{2129+1456+62+80} = \frac{3585}{3727} = 0.961900$

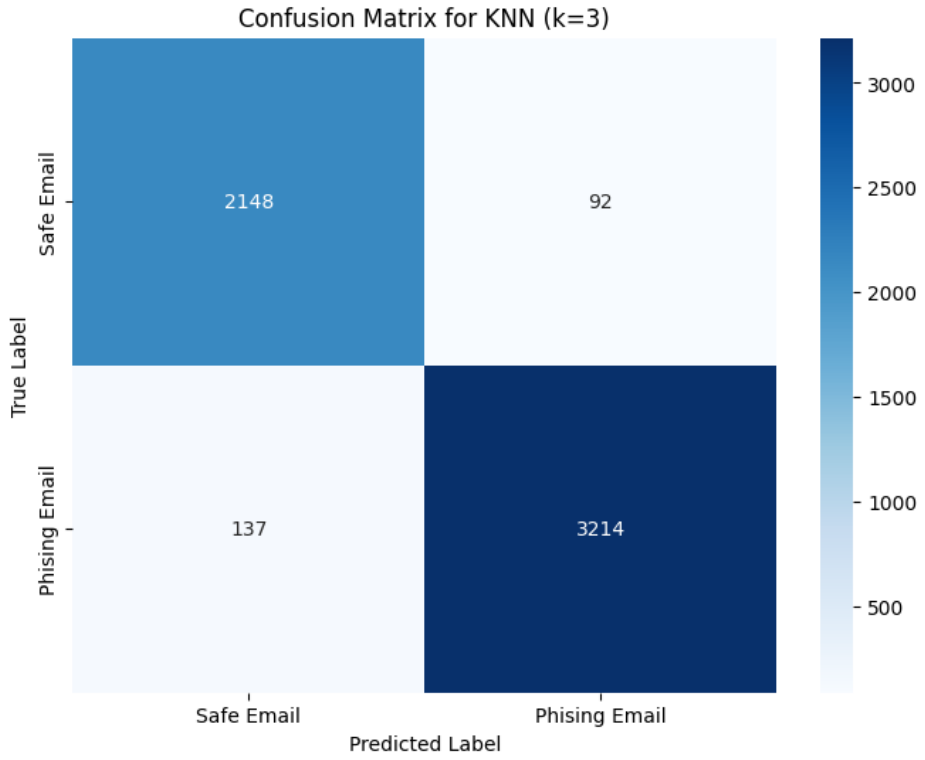
- **Precision =** $\frac{2129}{2129+62} = \frac{2129}{2191} = 0.971702$

- **Recall =** $\frac{2129}{2129+80} = \frac{2129}{2209} = 0.963785$

- **F1-Score = 2 x** $\frac{0.971702\ x\ 0.963785}{0.971702 + 0.963785} = 2x\frac{0.936511}{1.935487} = 0.967727$

3.7.2 Skenario 2 (Rasio Pembagian 70:30) - Evaluasi Model KNN dengan Word2Vec

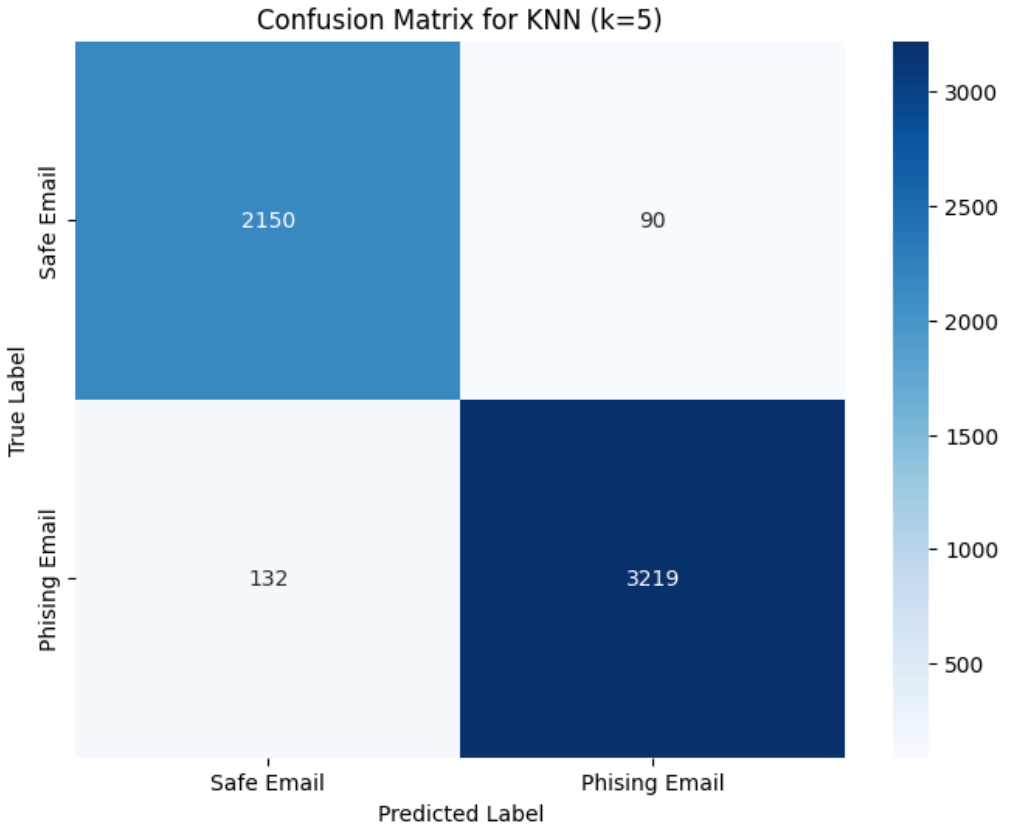
Skenario kedua pengujian dilakukan pada model yang telah dilatih dengan 70% dari jumlah dataset. Dengan jumlah data uji sebanyak 5591 data didapatkan hasil prediksi sebagai berikut:



- **KNN (k = 3)**
  - True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 3214
  - True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 2148
  - False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 137
  - False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 92

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{3214+2148}{3214+2148+92+137} = \frac{5362}{5591} = 0.959041$
- **Precision** = $\frac{3214}{3214+92} = \frac{3214}{3306} = 0.972172$
- **Recall** = $\frac{3214}{3214+137} = \frac{3214}{3351} = 0.959117$
- **F1-Score** = 2 x $\frac{0.972172\ x\ 0.959117}{0.972172 + 0.959117} = 2x\frac{0.932426}{1.931289} = 0.965600$



- **KNN (k=5)**
  - True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 2150
  - True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 3219
  - False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 132
  - False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 90

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{3219+2150}{3219+2150+90+132} = \frac{5369}{5591} = 0.960293$
- **Precision** = $\frac{3219}{3219+90} = \frac{3219}{3309} = 0.972801$
- **Recall** = $\frac{3219}{3219+132} = \frac{3219}{3351} = 0.960609$

- **F1-Score = 2 x** $\frac{0.972801\ x\ 0.960609}{0.972801 + 0.960609} = 2x\frac{0.93448}{1.93341} =$ **0.966667**

3.8 Model Naive Bayes dengan Metode Word2Vec

```
# Evaluasi Naive Bayes
nb_results = []

# Inisialisasi Naive Bayes (GaussianNB)
nb_classifier = GaussianNB(var_smoothing=1e-9)

# Convert X_train and X_test to dense arrays
X_train_dense = X_train  # Assuming X_train is a sparse matrix
X_test_dense = X_test   # Assuming X_test is a sparse matrix

# Melatih model Naive Bayes
nb_classifier.fit(X_train_dense, y_train)

# Menghitung metrik evaluasi
metrics = evaluate_model(nb_classifier, X_test_dense, y_test)
```

Berikut penjelasan pembuatan Naive Bayes:

- **GaussianNB()**: Menginisialisasi model Naive Bayes berbasis distribusi Gaussian, yang mengasumsikan bahwa fitur dalam dataset memiliki distribusi normal.
- **var_smoothing=1e-9**: Menambahkan sedikit nilai kecil (1e-9) untuk menghindari pembagian dengan 0 pada variansi, yang membantu meningkatkan kestabilan perhitungan probabilitas.

3.9 Evaluasi Naive Bayes dengan Metode Word2Vec

```
# Fungsi untuk mengevaluasi model

def evaluate_model(model, X_test, y_test):

    y_pred = model.predict(X_test)



    # Pastikan label yang ada di y_test yang digunakan dalam
confusion_matrix

    labels = np.unique(y_test)  # Menyesuaikan label dengan yang ada di
y_test
```

```
    cm = confusion_matrix(y_test, y_pred, labels=labels)
    tn, fp, fn, tp = cm.ravel()
    return {
        'Confusion Matrix': cm,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred,
pos_label=labels[1]),
        'Recall': recall_score(y_test, y_pred, pos_label=labels[1]),
        'F1-Score': f1_score(y_test, y_pred, pos_label=labels[1])
    }

# Menyimpan hasil evaluasi
nb_results.append({
    'Model': 'Naive Bayes',
    'Accuracy': metrics['Accuracy'],
    'Precision': metrics['Precision'],
    'Recall': metrics['Recall'],
    'F1-Score': metrics['F1-Score']
})

# Buat DataFrame untuk meringkas hasil Naive Bayes
nb_summary_df = pd.DataFrame(nb_results)
# Tampilkan hasil
print(nb_summary_df)
```

Berikut adalah penjelasan singkat dan rumus untuk masing-masing metrik evaluasi:

1. Akurasi (Accuracy):

- **Definisi**: Mengukur proporsi prediksi yang benar dibandingkan dengan total prediksi.
- **Rumus**:

- $$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$$

Gambar 3.13 Rumus Menghitung Akurasi

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **TN**: True Negative (Prediksi benar untuk kelas negatif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas negatif)

2. Precision

- **Definisi**: Mengukur seberapa banyak prediksi positif yang benar-benar positif.
- **Rumus**:
  - $$\text{Precision} = \frac{TP}{TP + FP}$$

Gambar 3.14 Rumus Perhitungan Precision

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FP**: False Positive (Prediksi salah untuk kelas positif)

3. Recall (Sensitivitas)

- **Definisi**: Mengukur seberapa banyak kasus positif yang berhasil dikenali oleh model.
- **Rumus**:
  - $$\text{Recall} = \frac{TP}{TP + FN}$$

Gambar 3.15 Rumus Perhitungan Recall

  - **TP**: True Positive (Prediksi benar untuk kelas positif)
  - **FN**: False Negative (Prediksi salah untuk kelas positif)

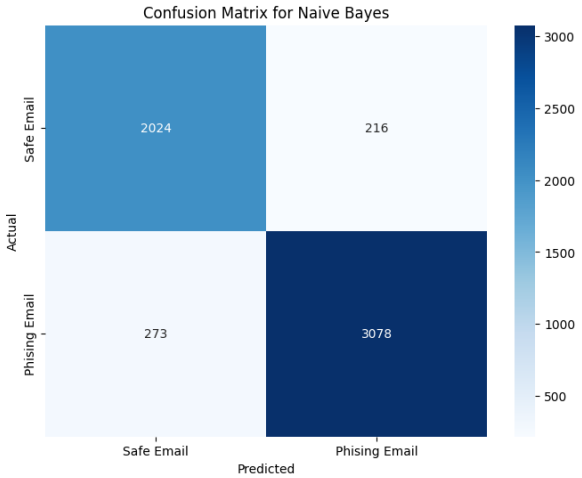
4. F1-Score

- **Definisi**: Rata-rata harmonis antara Precision dan Recall, memberikan keseimbangan keduanya.
- **Rumus**:
- $$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gambar 3.16 Rumus Perhitungan F1-Score

3.9.1 Skenario 1 (Rasio Pembagian 80:20) - Evaluasi Model Naive Bayes dengan Word2Vec

Skenario pertama pengujian dilakukan pada model yang telah dilatih dengan 80% dari jumlah dataset. Dengan jumlah data uji sebanyak 3727 data didapatkan hasil prediksi sebagai berikut:



1. True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 1388
2. True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 2022
3. False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 130
4. False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 187

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{2022+1388}{2022+1388+130+187} = \frac{3410}{3727} = 0.914945$

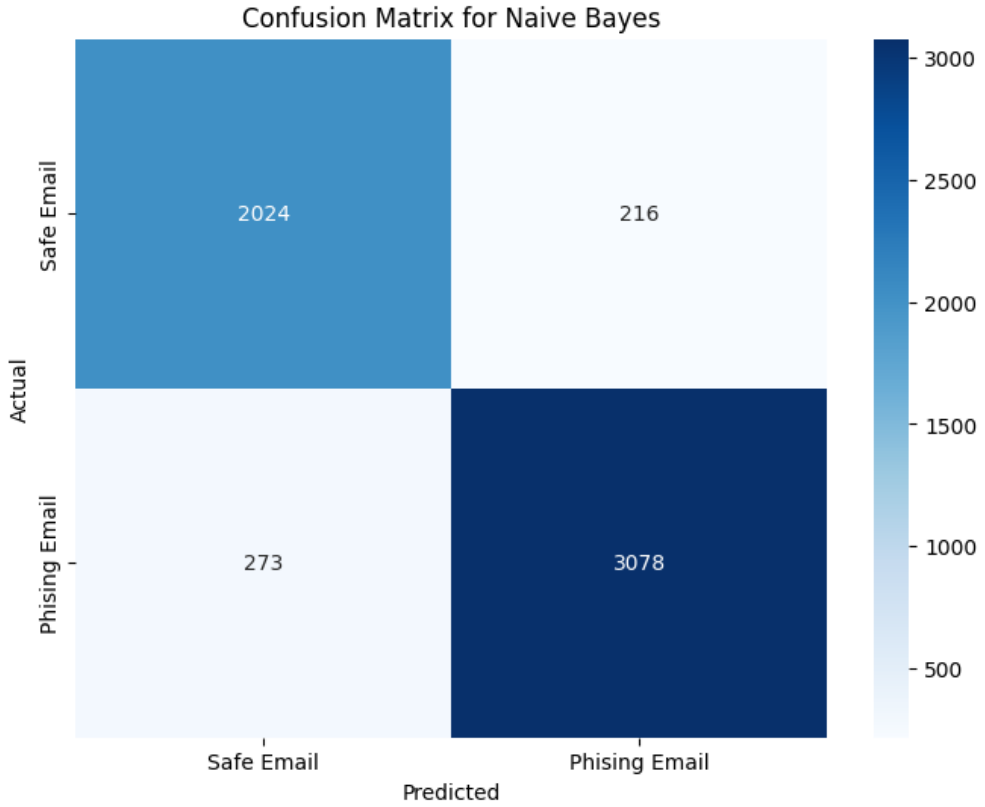
- **Precision** = $\frac{2022}{2022+130} = \frac{2022}{2152} = 0.939591$

- **Recall** = $\frac{2022}{2022+187} = \frac{2022}{2209} = 0.915346$

- **F1-Score** = 2 x $\frac{0.939591\ x\ 0.915346}{0.939591 + 0.915346} = 2x\frac{0.860050}{1.854937} = 0.92731$

3.9.2 Skenario 2 (Rasio Pembagian 70:30) - Evaluasi Model Naive Bayes dengan Word2Vec

Skenario kedua pengujian dilakukan pada model yang telah dilatih dengan 70% dari jumlah dataset. Dengan jumlah data uji sebanyak 5591 data didapatkan hasil prediksi sebagai berikut:



1. True Negative (TN) : Data berlabel negatif yang diprediksi benar sebagai label negatif sebanyak 2024
2. True Positive (TP) : Data berlabel positif yang diprediksi benar sebagai label positif sebanyak 3078

3. False Negative (FN) : Data berlabel positif yang diprediksi salah sebagai label negatif sebanyak 273

4. False Positive (FP) : Data berlabel negatif yang diprediksi salah sebagai label positif sebanyak 216

Perhitungan Akurasi, Precision, Recall, dan F1-Score:

- **Akurasi** = $\frac{3078+2024}{3078+2024+216+273} = \frac{5102}{5591} = 0.912538$

- **Precision** = $\frac{3078}{3078+216} = \frac{3078x}{3294} = 0.934426$

- **Recall** = $\frac{3078}{3078+273} = \frac{3078}{3351} = 0.918532$

- **F1-Score** = 2 x $\frac{0.912538\ x\ 0.918532}{0.912538 + 0.91532} = 2x\frac{0.838195}{1.827858} = 0.926411$

3.10.1 Analisis Perbandingan Model

Model	Metode	K Value	Akurasi	Precision	Recall	F1-Score	Data Uji & Data Latih
K-Means	TF-IDF	3	0.619265	0.998737	0.3580	0.527158	80:20
K-Means	TF-IDF	5	0.582506	0.998473	0.296062	0.456704	80:20
K-Means	TF-IDF	3	0.603470	0.999120	0.338705	0.505906	70:30
K-Means	TF-IDF	5	0.583438	0.998051	0.305580	0.467900	70:30
K-Means	Word2Vec	3	0.958143	0.968508	0.960616	0.964545	80:20
K-Means	Word2Vec	5	0.961900	0.971702	0.963785	0.967727	80:20
K-Means	Word2Vec	3	0.959041	0.972172	0.959117	0.965600	70:30
K-Means	Word2Vec	5	0.960293	0.972801	0.960609	0.966667	70:30
Naive Bayes	TF-IDF	-	0.903944	0.962519	0.871888	0.914964	80:20
Naive Bayes	TF-IDF	-	0.905741	0.965086	0.874366	0.917489	70:30

Naive Bayes	Word2Vec	-	0.914945	0.939591	0.915346	0.92731	80:20
Naive Bayes	Word2Vec	-	0.912538	0.934426	0.918532	0.926411	70:30

1. Performa K-Means dengan TF-IDF

- Pengaruh nilai K:
  - Untuk K=3, akurasi lebih tinggi dibandingkan K=5 pada kedua pembagian data (80:20 dan 70:30).
  - Nilai F1-Score untuk K=3 juga lebih baik dibanding K=5, menunjukkan bahwa K=3 lebih optimal untuk metode TF-IDF.
- Pengaruh data latih dan uji:
  - Akurasi sedikit lebih tinggi pada pembagian data 80:20 dibanding 70:30, yang wajar karena data latih lebih banyak.

2. Performa K-Means dengan Word2Vec

- Pengaruh nilai K:
  - Untuk K=3, akurasi sedikit lebih rendah dibandingkan K=5 pada kedua pembagian data.
  - Namun, F1-Score untuk K=5 lebih baik di semua pembagian data, menunjukkan bahwa Word2Vec lebih stabil dengan K=5.
- Pengaruh data latih dan uji:
  - Akurasi dan F1-Score lebih tinggi pada pembagian 70:30 dibandingkan 80:20, yang bisa disebabkan oleh model memanfaatkan lebih banyak data uji untuk validasi Word2Vec.

3. Perbandingan TF-IDF dan Word2Vec pada K-Means

- Word2Vec secara konsisten menunjukkan hasil akurasi, precision, recall, dan F1-Score yang lebih tinggi dibandingkan TF-IDF.
- Ini menunjukkan bahwa representasi Word2Vec lebih efektif untuk clustering dibandingkan TF-IDF pada data.

4. Performa Naive Bayes dengan TF-IDF

- Pengaruh pembagian data:
    - Hasil akurasi, precision, recall, dan F1-Score untuk pembagian 70:30 sedikit lebih tinggi dibandingkan 80:20.
    - Hal ini menunjukkan Naive Bayes mampu memanfaatkan lebih banyak data latih untuk meningkatkan performanya.

5. Performa Naive Bayes dengan Word2Vec

- Pengaruh pembagian data:
    - Pada pembagian 80:20, akurasi lebih tinggi dibandingkan 70:30, namun recall sedikit lebih rendah.
    - Naive Bayes tampaknya sensitif terhadap data uji yang lebih sedikit.

6. Perbandingan TF-IDF dan Word2Vec pada Naive Bayes

- Word2Vec kembali menunjukkan performa yang lebih baik daripada TF-IDF, terutama pada metrik recall dan F1-Score.
- Naive Bayes tampaknya lebih cocok digunakan dengan representasi Word2Vec

.

7. Perbandingan K-Means dan Naive Bayes

- Akurasi:
    - Naive Bayes secara konsisten lebih unggul daripada K-Means di semua metode (TF-IDF dan Word2Vec).
- F1-Score:
    - Naive Bayes juga memiliki F1-Score lebih tinggi dibandingkan K-Means, yang menunjukkan kemampuannya menangani keseimbangan precision dan recall.
- K-Means cenderung lebih sensitif terhadap nilai K dan metode representasi data (TF-IDF vs Word2Vec).

BAB IV

KESIMPULAN

Analisis Model Terbaik:

- **Model Terbaik: K-Means dengan Word2Vec (K=5, pembagian data 80:20)**
    - Akurasi mencapai 96.19%, dengan precision dan recall yang hampir seimbang (menunjukkan bahwa model efektif dalam mengklasifikasikan data).
    - F1-Score yang tinggi (0.967) menunjukkan keseimbangan antara recall dan precision, menjadikan model ini paling optimal.

Tantangan yang Dihadapi:

1. **Pemilihan Nilai K**:
    - Pemilihan nilai K yang tepat untuk K-Means masih memerlukan eksperimen lebih lanjut. Pada beberapa percobaan, perbedaan hasil antara K=3 dan K=5 bisa cukup signifikan, namun tidak selalu konsisten di seluruh metode.
2. **Kompleksitas Model**:
    - Word2Vec membutuhkan lebih banyak waktu pemrosesan dan memori dibandingkan TF-IDF.
3. **Overfitting/Underfitting**:
    - Untuk beberapa pembagian data dan nilai K, ada kemungkinan overfitting atau underfitting, terutama dengan penggunaan K yang lebih besar atau kecil.

Saran Perbaikan untuk Studi Lanjutan:

1. **Pencarian Nilai K yang Lebih Optimal**:
    - Menggunakan metode seperti **elbow method** atau **silhouette score** untuk memilih nilai K yang lebih optimal.
2. **Peningkatan dalam Data**:
    - Meningkatkan jumlah data latih atau menerapkan augmentasi data untuk meningkatkan akurasi dan generalisasi model.