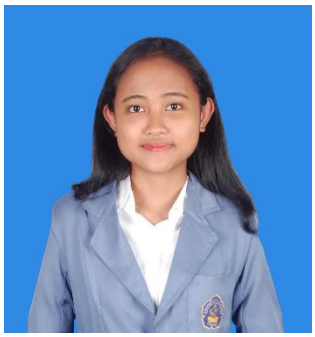
LAMPIRAN

Lampiran 1. Riwayat Hidup

RIWAYAT HIDUP



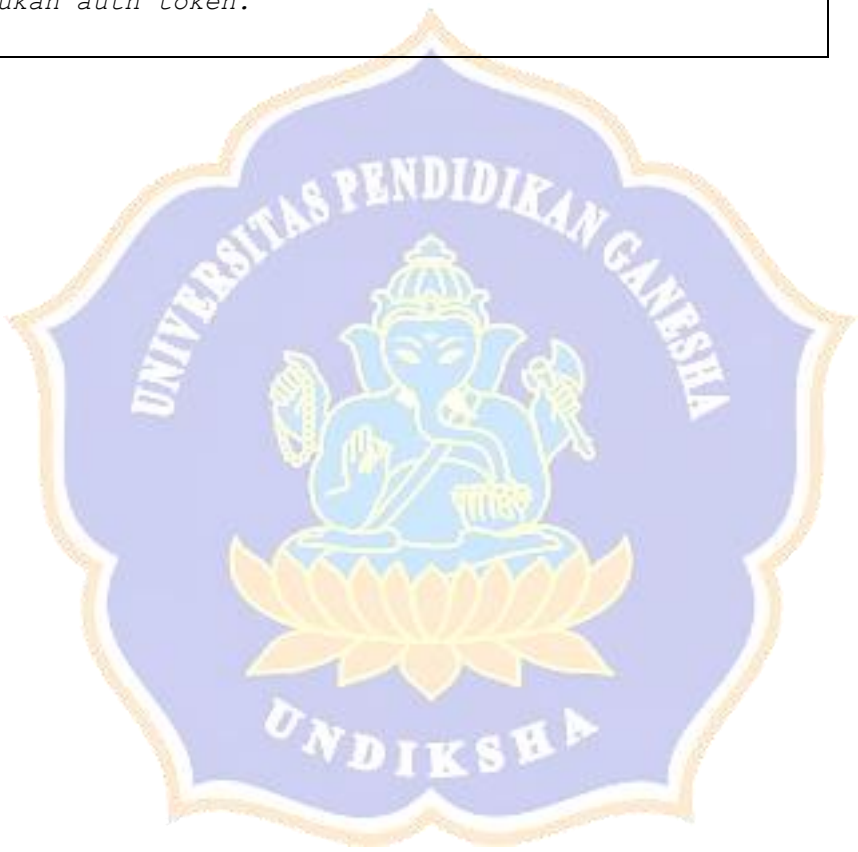
Ni Putu Ana Rainita lahir di Batuan Sukawati pada tanggal 20 November 2003. Penulis lahir dari pasangan suami istri Bapak I Ketut Wirna dan Ibu Ni Made Sudiani sebagai anak pertama dari dua bersaudara. Kini penulis beralamat di Banjar Puaya, Desa Batuan, Kecamatan Sukawati, Kabupaten Gianyar, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD ANA Putra dan lulus pada tahun 2015. Kemudian penulis melanjutkan pendidikan di SMP Negeri 1 Sukawati dan lulus tahun 2018. Pada tahun 2021, penulis lulus dari SMA Negeri 1 Sukawati jurusan MIA (Matematika dan Ilmu Pengetahuan Alam) dan melanjutkan studi (S1) di Universitas Pendidikan Ganesha dengan Program Studi Sistem Informasi, Jurusan Teknik Informatika. Seluruh perjalanan pendidikan ini semoga dapat menjadi pijakan yang kuat untuk mencapai kesuksesan di masa depan serta memberikan manfaat bagi banyak orang.

Lampiran 2. Kode *Crawling* Data

```
!pip install pandas          #install pustaka pandas
!sudo apt-get install nodejs -y   #install node.js
filename = 'kipk.csv'   #nama file untuk menyimpan data
tweet hasil crawling

search_keyword   =   'KIP-K   lang:id   until:2024-09-30
since:2020-02-20' #keyword crawling
limit = 2000 #banyak data yang ingin diambil

!npx   -y   tweet-harvest@2.6.1   -o   "{filename}"   -s
"{search_keyword}" --tab "LATEST" -l {limit} --token
{twitter_auth_token}
#mengeksekusi   atau   Menjalankan   tweet   harvest   dan
memasukan auth token.
```



Lampiran 3. Kode *Pre-Processing* Data

1. *Case Folding*

```
df['full_text'] = df['full_text'].str.lower()
df.head(5)
```

2. *Cleaning*

```
import re

def clean_text(tweet):
    # 1. Hapus username (@username)
    tweet = re.sub(r'@[^\s]+', '', tweet)

    # 2. Hapus URL (tautan/link)
    tweet = re.sub(r'https?://\S+|www\.\S+', '', tweet)

    # 3. Hapus HTML tag
    tweet = re.sub(r'<.*?>', '', tweet)

    # 4. Hapus hashtag (#tagar)
    tweet = re.sub(r'#\w+', '', tweet)

    # 5. Hapus RT (retweet tag)
    tweet = re.sub(r'RT\s', '', tweet)

    # 6. Hapus angka
    tweet = re.sub(r'\d+', '', tweet)

    # 7. Hapus simbol
    tweet = re.sub(r'[^a-zA-Z0-9\s]', '', tweet)

    # 8. Hapus karakter tunggal
    tweet = re.sub(r"\b[a-zA-Z]\b", '', tweet)

    # 9. Hapus karakter berulang (>2 kali menjadi 1)
    tweet = re.sub(r'(.)\1{2,}', r'\1', tweet)

    # 10. Hapus emoji
     tweet = re.sub(r"[^\w\s]", "", tweet)


    # 11. Hapus spasi berlebihan (lebih dari satu spasi)
    tweet = re.sub(r'\s+', ' ', tweet).strip()

    return tweet

# 12. Terapkan pembersihan pada kolom 'full_text'
df['cleaning']= df['full_text'].apply(clean_text)

# 13. Tampilkan 8 baris pertama hasil pembersihan
df.head(8)
```

Lampiran 4. Kode *Pre-Processing* Data

1. *Tokenizing*

```
# Fungsi untuk tokenisasi
def tokenize(text):
    text = re.split(r'\W+', text)
    return text

# Menerapkan fungsi tokenization pada kolom 'cleaning'
dan menambahkan hasil ke kolom 'Tokenization'
df['tokenize']    =    df['cleaning'].apply(lambda    x:
tokenize(x.lower()))
df.head(10)
```

2. *Normalize*

```
import pandas as pd
import re

# Path file kamus slang
file_path
='/content/drive/MyDrive/skripsi/kbba_backup.txt'

# Membaca kamus slang dari file
kbba_dictionary = pd.read_csv(file_path, delimiter='\t',
names=['slang', 'formal'], header=None, encoding='utf-
8')

# Membuat kamus slang_dict
slang_dict = dict(zip(kbba_dictionary['slang'],
kbba_dictionary['formal']))

# Fungsi untuk mengganti slang word
def convert_slangword(text):
    # Mengganti kata slang dengan kata formal
    normalized_words = [slang_dict[word] if word in
slang_dict else word for word in text.split()]
    return ' '.join(normalized_words)  # Mengembalikan
dalam bentuk string

# Mengaplikasikan fungsi convert slangword pada kolom
tokenization
df['slang_word'] = df['tokenize'].apply(lambda x: '
'.join(x) if isinstance(x, list) else
x).apply(convert_slangword)
df[['slang_word']]
df.head(5)
```

Lampiran 5. Kode *Pre-Processing* Data

1. *Stopword Removal*

```
import re
import nltk

# Definisikan daftar stopword
stopword = nltk.corpus.stopwords.words('indonesian')

# Kata yang ingin dikecualikan dari stopword (misalnya
"tidak")
exclude_words = ['tidak']

# Buat pattern regex untuk stopword, kecuali kata 'tidak'
stopword_pattern = r'\b(?:{})\b'.format('|'.join([word
for word in stopword if word not in exclude_words]))

# Fungsi untuk menghapus stopword
def remove_stopwords(text):
    text = re.sub(stopword_pattern, '', text)  # Menghapus
stopword dengan regex
    tokens = re.split('\W+', text)  # Memisahkan teks
menjadi token
    return tokens

# Menerapkan fungsi remove_stopwords pada kolom
'slang_word'
df['Stop_Removal'] = df['slang_word'].apply(lambda x:
remove_stopwords(x))

# Melihat 10 baris pertama setelah stopword removal
df.head(10)
```

Lampiran 6. Kode *Pre-Processing* Data

1. Stemming

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# Membuat objek StemmerFactory dan instansi stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Fungsi untuk menerapkan stemming pada setiap kata dalam
teks
def apply_stemming(text):
    # Jika text berupa list, gabungkan menjadi string
terlebih dahulu
    if isinstance(text, list):
        text = ' '.join(text)
    words = text.split()  # Memisahkan teks menjadi kata-
kata
    stemmed_words = [stemmer.stem(word) for word in
words]  # Menerapkan stemming pada setiap kata
    stemmed_text = ' '.join(stemmed_words)  #
Menggabungkan kata yang telah di-stem kembali menjadi teks
    return stemmed_text
# Menerapkan fungsi apply_stemming pada kolom
'Stop_Removal'
df['Tweet_Stemmed'] = df['Stop_Removal'].apply(lambda x:
apply_stemming(x))
# Melihat 10 baris pertama setelah stemming
df.head(10)
```

Lampiran 7. Kode TF-IDF

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import
TfidfTransformer

# Membaca data dari file CSV
df =
pd.read_csv('/content/drive/MyDrive/skripsi/cobatfidf.csv')
# Inisialisasi TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer()
# Menghitung vektor kata menggunakan TF-IDF Vectorizer
word_vector =
tfidf_vectorizer.fit_transform(df['Tweet_Stemmed'])
# Mendapatkan daftar fitur (kata-kata)
features = tfidf_vectorizer.get_feature_names_out()

# Membuat DataFrame untuk Term Frequency (TF)
tf_matrix = pd.DataFrame(word_vector.toarray(),
columns=features)

# Inisialisasi TF-IDF Transformer
tfidf_transformer = TfidfTransformer(norm=None)

# Menghitung IDF
X = tfidf_transformer.fit_transform(word_vector)
idf = pd.DataFrame({'Term': features, 'IDF':
tfidf_transformer.idf_})
# Membuat DataFrame untuk TF-IDF
tfidf_df = pd.DataFrame(X.toarray(), columns=features)


# Mencetak hasil TF-IDF
print("\nTF-IDF Matrix:")
print(tfidf_df)

# Menampilkan DataFrame TF
print("\nTF Matrix:")
print(tf_matrix)

# Menampilkan DataFrame IDF
print("\nIDF Values:")
print(idf)
```

Lampiran 8. Kode Implementasi *Naive Bayes*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedKFold,
learning_curve
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report
from wordcloud import WordCloud
from collections import Counter

# Membaca data TF-IDF yang sudah diproses
df pd.read_csv('/content/drive/MyDrive/skripsi/tfidf_matrix
(1).csv')

# Memisahkan fitur dan label
X = df.drop(columns=['label'])
y = df['label']

# Inisialisasi array untuk menyimpan metrik per fold
all_y_true = []
all_y_pred = []
fold_accuracy = []


# Melakukan StratifiedKFold untuk 10-fold cross-validation
kf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=0)

# Inisialisasi model Naive Bayes Multinomial
nb_model = MultinomialNB(alpha=1.0, fit_prior=True)

# K-Fold Cross-Validation dengan MultinomialNB
for fold_idx, (train_idx, test_idx) in enumerate(kf.split(X,
y), 1):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

# Melatih model
nb_model.fit(X_train, y_train)
y_pred = nb_model.predict(X_test)

# Evaluasi model
test_accuracy = accuracy_score(y_test, y_pred)
fold_accuracy.append(test_accuracy)

all_y_true.extend(y_test)
all_y_pred.extend(y_pred)

class_rep = classification_report(y_test, y_pred,
output_dict=True)
```

Lampiran 9. Kode Implementasi *Support Vector Machine*

1. Implementasi *Support Vector Machine*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedKFold,
learning_curve
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report
from wordcloud import WordCloud
# Membaca data TF-IDF yang sudah diproses
df =
pd.read_csv('/content/drive/MyDrive/skripsi/tfidf_matrix
(1).csv')

# Memisahkan fitur dan label
X = df.drop(columns=['label'])  # Fitur TF-IDF
y = df['label']                 # Label sentimen

# Inisialisasi array untuk menyimpan metrik per fold
fold_metrics = []
all_y_true = []
all_y_pred = []

# Melakukan StratifiedKFold untuk 10-fold cross-validation
kf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=0)

# K-Fold Cross-Validation dengan SVM
for fold_idx, (train_idx, test_idx) in enumerate(kf.split(X,
y), 1):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    # Inisialisasi model SVM dengan kernel RBF
    svm_model = SVC(kernel='rbf', C=10, gamma=0.01)

    # Melatih model
    svm_model.fit(X_train, y_train)
    y_pred = svm_model.predict(X_test)

    # Evaluasi model
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)
    class_rep = classification_report(y_test, y_pred,
output_dict=True)

    # Output hasil per fold
    print(f"Fold {fold_idx} - Akurasi: {accuracy:.2f}")

    # Simpan label aktual dan prediksi untuk seluruh fold
    all_y_true.extend(y_test)
    all_y_pred.extend(y_pred)

    # Simpan metrik per fold
```

```
    fold_metrics.append({
        'conf_matrix': conf_matrix,
        'class_rep': class_rep,
        'accuracy': accuracy,
    })

# Confusion matrix untuk seluruh data (gabungan semua fold)
labels_df = np.unique(all_y_true)  # Pastikan label sudah
unik dan terurut
conf_matrix_total = confusion_matrix(all_y_true, all_y_pred,
labels=labels_df)

# Menampilkan Confusion Matrix untuk seluruh fold
print("\nConfusion Matrix untuk Seluruh Fold:")
print(conf_matrix_total)

# Heatmap dari Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix_total,     annot=True,     fmt='d',
cmap='YlGnBu',
            xticklabels=labels_df, yticklabels=labels_df)
plt.title('Heatmap Confusion Matrix - SVM (All Folds)')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()

# Menampilkan akurasi rata-rata
avg_accuracy = sum(f['accuracy'] for f in fold_metrics) /
len(fold_metrics)
print(f"\nRata-rata     Akurasi     dari     10     Fold:
{avg_accuracy:.2f}")

# Menghitung dan menampilkan classification report gabungan
final_classification_report                                =
classification_report(all_y_true, all_y_pred)
print("\nClassification Report untuk Seluruh Fold:")
print(final_classification_report)
```

2. Pengujian Parameter C dan Gamma

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, precision_score,
recall_score, f1_score

# Menentukan rentang parameter
param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'gamma': [0.01, 0.1, 1, 10],
    'kernel': ['rbf']
}

# Inisialisasi model SVM
svm = SVC()
```

```
# Scorer untuk GridSearchCV
scorers = {
    'accuracy': 'accuracy',
    'precision': make_scorer(precision_score,
average='macro'),
    'recall': make_scorer(recall_score, average='macro'),
    'f1': make_scorer(f1_score, average='macro')
}

# Grid Search dengan 10-Fold CV untuk setiap metrik
grid_search_acc = GridSearchCV(svm, param_grid, cv=10,
scoring=scorers, refit='accuracy', n_jobs=-1)
grid_search_acc.fit(X, y)

# Mengambil hasil Grid Search
results = grid_search_acc.cv_results_

# Membentuk array hasil Grid Search untuk setiap metrik
C_values = param_grid['C']
gamma_values = param_grid['gamma']
num_C = len(C_values)
num_gamma = len(gamma_values)

# Fungsi untuk mengubah hasil GridSearchCV ke dalam array
heatmap
def extract_scores(metric_name):
    return
np.array(results[f'mean_test_{metric_name}']).reshape(num_C,
num_gamma)

accuracy_scores = extract_scores('accuracy')
precision_scores = extract_scores('precision')
recall_scores = extract_scores('recall')
f1_scores = extract_scores('f1')

# Membuat heatmap untuk semua metrik
fig, axes = plt.subplots(2, 2, figsize=(12, 8))

heatmaps = [
    (accuracy_scores, "Akurasi"),
    (precision_scores, "Presisi"),
    (recall_scores, "Recall"),
    (f1_scores, "F1-Score"),
]

for ax, (data, title) in zip(axes.flat, heatmaps):
    sns.heatmap(data, annot=True, fmt=".3f", cmap="viridis",
                xticklabels=gamma_values,
yticklabels=C_values, ax=ax)
    ax.set_xlabel("Gamma")
    ax.set_ylabel("C")
    ax.set_title(f"Heatmap {title} untuk Grid Search (SVM)")

plt.tight_layout()
plt.show()
```

Lampiran 10. Kode Implementasi SMOTE

```
# Mendefinisikan variabel target dan fitur teks
y = df['label']
X_text = df['Tweet_Stemmed']

# Inisialisasi TF-IDF Vectorizer & Transformasi TF-IDF
tfidf_vectorizer = TfidfVectorizer()
word_vector = tfidf_vectorizer.fit_transform(X_text)
features = tfidf_vectorizer.get_feature_names_out()

tfidf_transformer = TfidfTransformer(norm=None)
X_tfidf = tfidf_transformer.fit_transform(word_vector)

# Inisialisasi dan Penerapan SMOTE
smote = SMOTE(random_state=0)
X_resampled, y_resampled = smote.fit_resample(X_tfidf, y)

# Konversi data hasil resampling ke DataFrame
resampled_df     =     pd.DataFrame(X_resampled.toarray(),
columns=features)
resampled_df['sentimen'] = y_resampled
```

Lampiran 11. Surat Permohonan Pelabelan Data

KEMENTERIAN PENDIDIKAN, SAINS DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja Bali
Laman: http://ftk.undiksha.ac.id

Singaraja, 07 Januari 2025

Nomor : 46/UN48.11.1/KM/2025
Perihal : Surat Permohonan Data

Yth. Kepala SMA Negeri 1 Sukawati
Di tempat

Dengan hormat, ehubungan dengan proses penyelesaian Tugas Akhir / Skripsi, maka melalui surat ini kami mohon Bapak/ Ibu berkenan memberikan data yang dibutuhkan. Adapun mahasiswa yang akan melakukan pengambilan data seperti tersebut di bawah ini :

Nama	: Ni Putu Ana Rainita
Nim	: 2115091075
Prodi/Jurusan	: Sistem Informasi / Teknik Informatika
Data yang dibutuhkan	: Terkait data pelabelan Data Skripsi
Judul	: Komparasi Algoritma Naïve Bayes dan Support Vector Machine (SVM) Pada Analisis Sentimen Program Kartu Indonesia Pintar Kuliah

Demikian kami sampaikan. Atas perhatian dan kerjasamanya, diucapkan terima kasih.

a.n Dekan
Wakil Dekan Bidang Akademik

Made Windu Antara Kesiman
NIP. 198211112008121001

KEMENTERIAN PENDIDIKAN, SAINS DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
JURUSAN TEKNIK INFORMATIKA
Jalan Udayana Singaraja-Bali Kode Pos 81116
Tlp. (0362) 22570 Fax. (0362) 25735
Laman: www.undiksha.ac.id

Nomor : 05/UN48.11.5/KM/2025 Singaraja, 07 Januari 2025
Perihal : Surat Permohonan Pengambilan Data
Lampiran : -

Yth. Dekan FTK
Universitas Pendidikan Ganesha
Di tempat

Dengan hormat,
Sehubungan dengan proses penyelesaian Tugas Akhir / Skripsi yang dilaksanakan oleh saudara mahasiswa:

Nama	: Ni Putu Ana Rainita
Nim	: 2115091075
Prodi/Jurusan	: Sistem Informasi / Teknik Informatika
Instansi yg ditujui	: SMA Negeri 1 Sukawati
Jabatan yg dituju	: Kepala SMA Negeri 1 Sukawati
Data yang dibutuhkan	: Terkait data pelabelan Data Skripsi
Judul	: Komparasi Algoritma Naïve Bayes dan Support Vector Machine (SVM) Pada Analisis Sentimen Program Kartu Indonesia Pintar Kuliah

Bersama ini kami mohonkan kepada Bapak untuk berkenan memfasilitasi kebutuhan data untuk Tugas Akhir / Skripsi mahasiswa yang bersangkutan.
Demikian kami sampaikan. Atas perhatian dan kerjasama Bapak, kami ucapkan terima kasih.

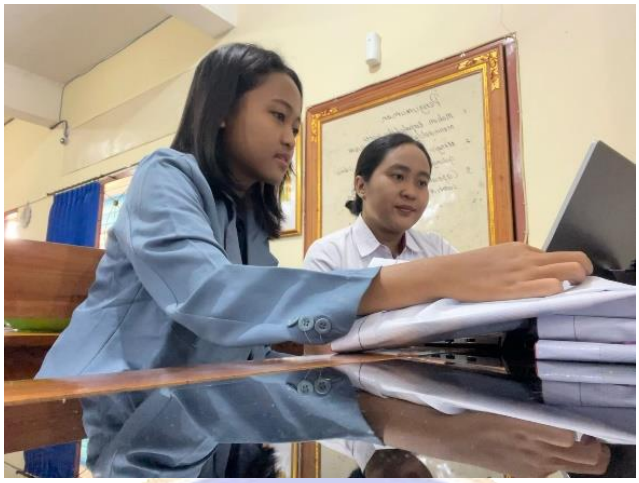
Ketua Jurusan Teknik Informatika,

Putu Hendra Suputra
NIP. 198212222006041001

Balai Sertifikasi Elektronik

Catatan :
- UU ITE No. 11 Tahun 2008 Pasal 5 ayat 1"Informasi Elektronik dan/atau Dokumen Elektornik dan/atau hasil cetaknya merupakan alat bukti hukum yang sah"
- Dokumen ini tertanda ditandatangani secara elektronik menggunakan sertifikat elektronik yang diterbitkan

Lampiran 12. Dokumentasi Pelabelan Bersama ibu Ida Ayu Candra Dewi, S.Pd., M.Pd.



Lampiran 13. Dokumentasi Pelabelan Bersama ibu Ni Putu Yuna Martika, S.Pd.