# LAMPIRAN

**Lampiran 1** Terjemah Hadits Nabi

Terjemahan HR. at-Thabrani : “Sebaik-baiknya manusia adalah yang paling bermanfaat bagi sesamanya.”

**Lampiran 2** *Source Code* Analisis Sentimen

```
# -*- coding: utf-8 -*-
"""Edit of sentiment_analysis_project_clg.ipynb
Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/17bUdx6oR7s6bZhjtRYM4eH9R594Lm2OB

Source Code:
https://github.com/Arpan-dev/Sentiment-Analysis/ also reference:
https://medium.com/@arpann1997/twitter-sentiment-analysis-using-supervised-learning-and-nltk-b7c12c4fa281
and https://www.youtube.com/watch?v=3DJ3mClKNWA

# Reviews Sentiment Analysis
## Importing Libraries"""

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import random
from sklearn import preprocessing
import nltk
```

```
from sklearn.utils import resample
nltk.download("puntk")
import pickle
from nltk.corpus import stopwords
import string
from nltk.stem import PorterStemmer
from nltk import word_tokenize
from wordcloud import WordCloud
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score,precision_score,multilabel_confusion_matrix,recall_score

"""## Importing the Dataset"""
df= pd.read_csv("RG_3000.csv")
df.head(10)

"""## Checking for Shape of Data"""
df.shape

"""## Checking for Null Values"""
df.isna().sum()       #checking null values

"""**We don't have any null value in text and one null value in the file, so we don't have to drop them**
```

```
## Droping the Null Values"""
df.dropna(inplace = True)
df.isna().sum()
df.duplicated().sum()

"""## Creating Pie Chart for Sentiment Column."""
explode = (0.1, 0,0)
fig = plt.figure(figsize =(12, 10))
plt.pie(df["Sentiment"].value_counts(),labels=["Neutral","Positive","Negative"],a
utopct="%1.1f%%",shadow=True,explode=explode)
plt.title('Distribution of sentiments',
          loc='left',
          fontsize=42,
          color="green")
plt.show()

"""## Downsampling Neutral Class"""
neutral_df = df[df["Sentiment"] == "Neutral"]
print(neutral_df.shape)

negative_df = df[df["Sentiment"] == "Negative"]
print(negative_df.shape)

positive_df = df[df["Sentiment"] == "Positive"]
print(positive_df.shape)

neutral_downsample = resample(neutral_df,
             replace=True,
             n_samples=len(negative_df),
             random_state=42)
print(neutral_downsample.shape)
```

```
positive_downsample = resample(positive_df,
            replace=True,
            n_samples=len(negative_df),
            random_state=42)
print(positive_downsample.shape)

df=pd.concat([neutral_downsample,positive_downsample,negative_df],ignore_in
dex=True)

df["Sentiment"].unique()

"""## Creating Pie Chart for Sentiment Column after Downsampling the Neutral
class"""
explode = (0.1, 0,0)
fig = plt.figure(figsize =(12, 10))
plt.pie(df["Sentiment"].value_counts(),labels=["Neutral","Positive","Negative"],a
utopct="%1.1f%%",shadow=True,explode=explode)
plt.title('Distribution of sentiments',
          loc='left',
          fontsize=42,
          color="green")
plt.show()

df.shape

df["Sentiment"].value_counts()

"""## Applying Label Encoder on Sentiment Column"""
label_encoder = preprocessing.LabelEncoder()

df['Sentiment']= label_encoder.fit_transform(df['Sentiment'])
```

```
df['Sentiment'].unique()

df.iloc[0]

df.iloc[1558]

#With iloc() function, we can retrieve a particular value belonging to a row and column using the index values assigned to it.
#Remember, iloc() function accepts only integer type values as the index values for the values to be accessed and displayed.

df.iloc[778]

"""- **1 = Neutral**
- **0 = Negative**
- **2 = Positive**

## Droping 'Star' since it is not required"""
df.drop(columns = ['Star'], inplace =True)

df.head(15)

"""## Visualizing Total count of Positive and Negative Sentiments"""
sns.set(style="whitegrid", context="talk")
plt.style.use("dark_background")
plt.figure(figsize=(6,6))
sns.countplot(x="Sentiment", data=df,palette=["yellow","green"])

"""## Checking for Duplicate Rows"""
df.duplicated().sum()
```

```
"""# Removing Url in dataset"""
def url_rmv(df):
    temp = []
    for i in df:
      #print(f"Data = {i} \n\n")

a=re.sub(r'"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]
+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{
};:'".,<>?«»“”‘’]))'", " ", i)
      temp.append(a)
    return temp

url_rmvd_text = url_rmv(df["Reviews"])
"""## Appending Url removed text in a new Col."""
df["url_rmvd_text"]= url_rmvd_text
df.head()

"""## Loading the English Stopwords"""
import nltk
from nltk.corpus import stopwords
nltk.download("stopwords")
sw=set(stopwords.words("english"))

"""## Creating a new list of Stopwords that doesn`t include "no","nor","not" """
neg_sw=["no","nor","not"]
pos_sw = [i for i in sw if i not in neg_sw]

"""## Loading the punctuations"""
import string
string.punctuation
```

```
"""## Loading PorterStemmer for Stemming"""
from nltk.stem import PorterStemmer

ps=PorterStemmer()

"""## Loading Word Tokenization"""
from nltk import word_tokenize

"""## Function for transforming text
- Lower case conversion
- Word Tokenization
- Alphanumeric Checking
- Stopword Removal
- Perform Stemming"""

def transform_text(text):
    text=text.lower()
    text=nltk.word_tokenize(text)

    L=[]
    for i in text:
        if i.isalnum():
            L.append(i)

    text=L[:]          #copying list
    L.clear()

    for i in text:
        if i not in pos_sw and i not in string.punctuation:
            L.append(i)
```

```
    text=L[:]
    L.clear()

    for i in text:
        L.append(ps.stem(i))

    return " ".join(L)

import nltk
nltk.download('punkt')
# Checking the performance of the Function
transform_text("i, am not! sohini, http: www.policybazzar.com")

"""## Applying the "transform_text" function on the df["url_rmvd_text"]"""
df["trans_text"]=df["url_rmvd_text"].apply(transform_text)

df["trans_text"].to_csv("preprocessing_final")

df.head(25)

"""## Frequency of words after Transformation"""
df["num_words_trans_text"]=df["trans_text"].apply(lambda
x:len(nltk.word_tokenize(x)))
df
df["num_words_trans_text"].mean()

"""## Frequency of words before Transformation"""
df["num_words_orignal_text"]=df["Reviews"].apply(lambda
x:len(nltk.word_tokenize(x)))
df["num_words_orignal_text"].mean()
```

```
"""**Here we can observe that before transformation Frequency of words was 171.165 but after transformation Frequency of words is 70.18. So our transformation of text is successful**
## Comparing the changes before and after Transformation"""
random.seed(10)
index = random.sample(range(0, 2748), 5)
comp = df[["num_words_trans_text","num_words_orignal_text"]].iloc[index]
comp["index"]=index
comp

plt.figure(figsize=(8,6))
sns.set_style("darkgrid")
plt.ylabel("num_words")
sns.lineplot(data=comp, x="index", y="num_words_trans_text",color='r')
sns.lineplot(data=comp, x="index", y="num_words_orignal_text",color='b')
plt.legend(labels=['num_words_trans_text', 'num_words_orignal_text'],fontsize = 'small',loc = 2)

"""**Here also we can observe that our transformation of text is successful**"""
from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,colormap='RdYlGn')

"""## Creating Word Cloud for Neutral Sentiment (1)"""
tweet_wc=wc.generate(df[(df["Sentiment"]==1)]["trans_text"].str.cat(sep=" "))
plt.figure(figsize=(18,8))
plt.imshow(tweet_wc)

"""## Creating Word Cloud for Negative Sentiment (0)"""
tweet_wc=wc.generate(df[(df["Sentiment"]==0)]["trans_text"].str.cat(sep=" "))
plt.figure(figsize=(18,8))
plt.imshow(tweet_wc)
```

```
"""## Creating Word Cloud for Positive Sentiment (2)"""
tweet_wc=wc.generate(df[(df["Sentiment"]==2)]["trans_text"].str.cat(sep=" "))
plt.figure(figsize=(18,8))
plt.imshow(tweet_wc)

"""## To count most no. of words in Neutral Sentiment(1)"""
tweet_corpus_neutral=[]
for msg in df[(df["Sentiment"]==1)]["trans_text"].tolist():
    for word in msg.split():
        tweet_corpus_neutral.append(word)

from collections import Counter
pd.DataFrame(Counter(tweet_corpus_neutral).most_common(20),columns=["Wor
ds","Count"])

plt.figure(figsize=(18,8))
sns.barplot(pd.DataFrame(Counter(tweet_corpus_neutral).most_common(20))[0],
pd.DataFrame(Counter(tweet_corpus_neutral).most_common(20))[1])
plt.xlabel("words")
plt.ylabel("word frequency")
plt.xticks(rotation="vertical")
plt.show()

"""## To count most no. of words in Positive Sentiment(2)"""
tweet_corpus_positive=[]        #to count most no of words
for msg in df[(df["Sentiment"]==2)]["trans_text"].tolist():
    for word in msg.split():
        tweet_corpus_positive.append(word)

pd.DataFrame(Counter(tweet_corpus_positive).most_common(20),columns=["W
ords","Count"])
```

```
plt.figure(figsize=(18,8))
sns.barplot(pd.DataFrame(Counter(tweet_corpus_positive).most_common(20))[0]
,pd.DataFrame(Counter(tweet_corpus_positive).most_common(20))[1])
plt.xlabel("words")
plt.ylabel("word frequency")
plt.xticks(rotation="vertical")
plt.show()

"""## To count most no. of words in Negative Sentiment(0)"""
tweet_corpus_negative=[]        #to count most no of words
for msg in df[(df["Sentiment"]== 0)]["trans_text"].tolist():
    for word in msg.split():
        tweet_corpus_negative.append(word)

pd.DataFrame(Counter(tweet_corpus_negative).most_common(20),columns=["W
ords","Count"])
plt.figure(figsize=(18,8))
sns.barplot(pd.DataFrame(Counter(tweet_corpus_negative).most_common(20))[0
],pd.DataFrame(Counter(tweet_corpus_negative).most_common(20))[1])
plt.xlabel("words")
plt.ylabel("word frequency")
plt.xticks(rotation="vertical")
plt.show()

"""## Creating Bag-of-Words"""
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=8000,ngram_range=(1,2))

X = cv.fit_transform(df["trans_text"]).toarray()

X
```

```
X.shape

y = df["Sentiment"]

y

"""## Performing Train Test Split"""
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

X_train

X_test

X_train.shape

X_test.shape

"""## Creating Model for MultinomialNB, BernoulliNB, Random forest, Logistic regression, Decision tree, and Support Vector Machine"""
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score,precision_score,multilabel_confusion_matrix,recall_score

for c in [0.01, 0.05, 0.25, 0.5, 0.75, 1]:
  svm = LinearSVC(C=c)
```

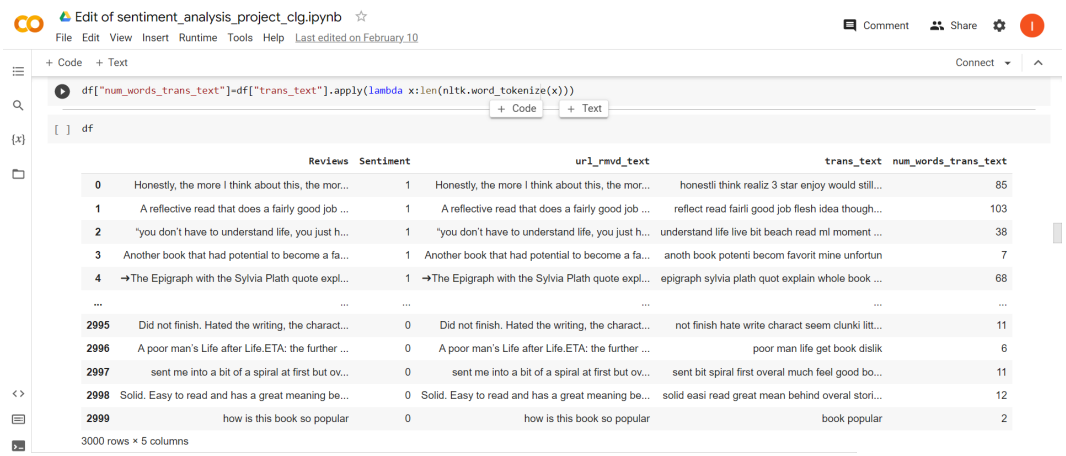
```
 svm.fit(X_train, y_train)
      print('Akurasi untuk c = %s : %s' %(c, accuracy_score(y_test,
svm.predict(X_test))))

svm = LinearSVC(C = 0.05)
svm.fit(X_train, y_train)

print('Accuracy score model final: %s ' %accuracy_score(y_test,
svm.predict(X_test)))

mnb=MultinomialNB()
bnb=BernoulliNB()
lr=LogisticRegression(max_iter=300,multi_class="multinomial")
rfc=RandomForestClassifier(n_estimators=50,random_state=2,max_depth=25)
dtc=DecisionTreeClassifier(max_depth=30)
svm=SupportVectorMachine = LinearSVC

def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_test)
    accuracy=accuracy_score(y_test,y_pred)
    precision=precision_score(y_test,y_pred,average="micro")

    return accuracy,precision

clfs = {
    'Multinomial Naive Bayes': mnb,
    'Bernoulli Naive Bayes': bnb,
    'Logistic Regression': lr,
    'Random Forest Clasifier': rfc,
    'Decision Tree Clasifier': dtc,
```

```
    'Support Vector Machine' : svm,
}

accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

                    current_accuracy,current_precision = train_classifier(clf,
X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)

performance=pd.DataFrame({"Algorithm":clfs.keys(),"Accuracy":accuracy_score
s,"Precision":precision_scores}).sort_values("Precision",ascending=False,ignore_
index=True)

performance
"""**Logistic Regression is giving the best accuracy and precison score among
the rest of five**
## Saving the Logistic Regression model to the local system"""
import pickle
filename="final_model.pickle"
pickle.dump(lr,open(filename,"wb"))

filename_cv="final_cv.pickle"
```

```
pickle.dump(cv,open(filename_cv,"wb"))

"""## Bulk Prediction
### Prediction of x_test data using the saved model"""
loaded_model=pickle.load(open(filename,"rb"))
y_pred=loaded_model.predict(X_test)
y_pred

comp = {"y_test(actual)":y_test ,"y_pred":y_pred}
comp_df = pd.DataFrame.from_dict(comp)
comp_df

"""## Blind Data"""
tweet = [input("Enter a Review: ")]

tweet = pd.DataFrame(tweet,columns = ["Reviews"])
tweet

url_rmvd_text = url_rmv(tweet["Reviews"])
url_rmvd_text

tweet["url_rmvd_text"]= url_rmvd_text
#tweet

tweet["trans_text"]=tweet["url_rmvd_text"].apply(transform_text)
#tweet

cv = pickle.load(open('final_cv.pickle','rb'))

X = cv.transform(tweet["trans_text"])
```

```
cv = pickle.load(open('final_model.pickle','rb'))
pred=loaded_model.predict(X)[0]

if pred == 0:
        print("Negative Sentiment")
elif pred == 1:
        print("Neutral Sentiment")
else:
  print("Positive Sentiment")
```

**Lampiran 3** Pratinjau Jumlah Kata pada Hasil Transformasi Teks



Edit of sentiment_analysis_project_clg.ipynb

File Edit View Insert Runtime Tools Help Last edited on February 10

Comment Share

+ Code + Text Connect

```
df["num_words_trans_text"]=df["trans_text"].apply(lambda x:len(nltk.word_tokenize(x)))
```

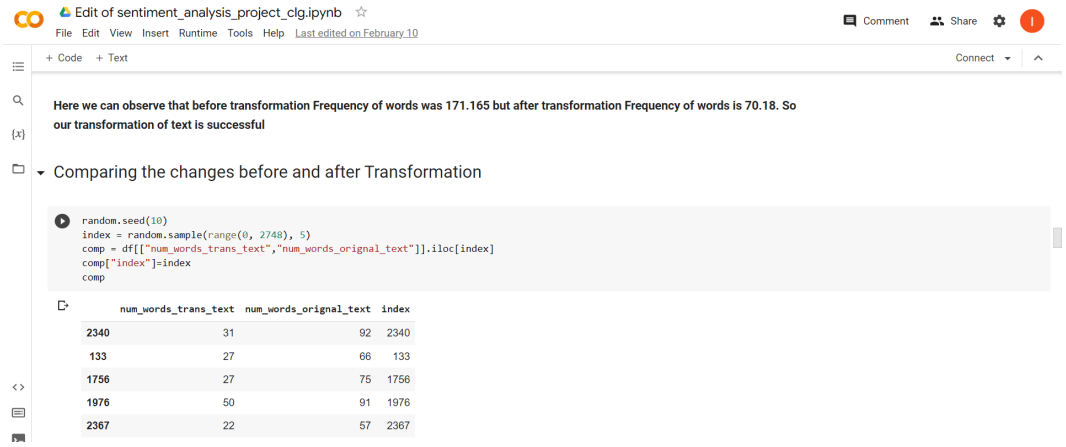
+ Code + Text

```
[ ] df
```

| | Reviews | Sentiment | url_rmvd_text | trans_text | num_words_trans_text |
|---|---|---|---|---|---|
| 0 | Honestly, the more I think about this, the mor... | 1 | Honestly, the more I think about this, the mor... | honestli think realiz 3 star enjoy would still... | 85 |
| 1 | A reflective read that does a fairly good job ... | 1 | A reflective read that does a fairly good job ... | reflect read fairli good job flesh idea though... | 103 |
| 2 | "you don't have to understand life, you just h... | 1 | "you don't have to understand life, you just h... | understand life live bit beach read ml moment ... | 38 |
| 3 | Another book that had potential to become a fa... | 1 | Another book that had potential to become a fa... | anoth book potenti becom favorit mine unfortun | 7 |
| 4 | →The Epigraph with the Sylvia Plath quote expl... | 1 | →The Epigraph with the Sylvia Plath quote expl... | epigraph sylvia plath quot explain whole book ... | 68 |
| ... | ... | ... | ... | ... | ... |
| 2995 | Did not finish. Hated the writing, the charact... | 0 | Did not finish. Hated the writing, the charact... | not finish hate write charact seem clunki litt... | 11 |
| 2996 | A poor man's Life after Life.ETA: the further ... | 0 | A poor man's Life after Life.ETA: the further ... | poor man life get book dislik | 6 |
| 2997 | sent me into a bit of a spiral at first but ov... | 0 | sent me into a bit of a spiral at first but ov... | sent bit spiral first overal much feel good bo... | 11 |
| 2998 | Solid. Easy to read and has a great meaning be... | 0 | Solid. Easy to read and has a great meaning be... | solid easi read great mean behind overal stori... | 12 |
| 2999 | how is this book so popular | 0 | how is this book so popular | book popular | 2 |

3000 rows × 5 columns

**Lampiran 4** Perbandingan Jumlah Kata Sebelum dan Sesudah Transformasi Teks



Edit of sentiment_analysis_project_clg.ipynb

File Edit View Insert Runtime Tools Help Last edited on February 10

Comment Share

+ Code + Text Connect

**Here we can observe that before transformation Frequency of words was 171.165 but after transformation Frequency of words is 70.18. So our transformation of text is successful**

Comparing the changes before and after Transformation

```
random.seed(10)
index = random.sample(range(0, 2748), 5)
comp = df[["num_words_trans_text","num_words_orignal_text"]].iloc[index]
comp["index"]=index
comp
```

| | num_words_trans_text | num_words_orignal_text | index |
|---|---|---|---|
| 2340 | 31 | 92 | 2340 |
| 133 | 27 | 66 | 133 |
| 1756 | 27 | 75 | 1756 |
| 1976 | 50 | 91 | 1976 |
| 2367 | 22 | 57 | 2367 |

**Lampiran 5** Menghitung Jumlah Token Kata

| | Netral | | Positif | | Negatif | |
|---|---|---|---|---|---|---|
| **No.** | **Kata** | **Jumlah** | **Kata** | **Jumlah** | **Kata** | **Jumlah** |
| **1** | book | 2044 | book | 2505 | book | 2366 |
| **2** | life | 1288 | life | 2299 | life | 1314 |
| **3** | like | 970 | live | 1336 | not | 1198 |
| **4** | live | 900 | nora | 1110 | like | 1084 |
| **5** | read | 843 | read | 1032 | live | 832 |
| **6** | not | 796 | librari | 914 | read | 801 |
| **7** | nora | 750 | one | 844 | nora | 781 |
| **8** | stori | 524 | not | 773 | one | 620 |
| **9** | librari | 516 | differ | 716 | would | 552 |
| **10** | one | 515 | like | 711 | end | 498 |
| **11** | end | 505 | regret | 633 | depress | 483 |
| **12** | would | 498 | love | 624 | charact | 470 |
| **13** | realli | 496 | could | 606 | get | 466 |
| **14** | think | 465 | would | 601 | realli | 464 |
| **15** | feel | 449 | stori | 568 | think | 448 |
| **16** | love | 442 | midnight | 568 | librari | 440 |
| **17** | differ | 426 | end | 502 | could | 438 |
| **18** | could | 391 | think | 498 | feel | 430 |
| **19** | regret | 358 | haig | 478 | stori | 408 |
| **20** | time | 352 | realli | 478 | time | 387 |

## DAFTAR RIWAYAT HIDUP PENELITI

**1. Nama Lengkap** : Isnani Hayati
**2. NIM** : 190101120246
**3. Tempat, Tanggal Lahir** : Banua Jingah, 29 Agustus 2001
**4. Kebangsaan** : Indonesia
**4. Alamat** : Hulu Sungai Tengah, Kalimantan Selatan
**5. Alamat email** : 190101120246@uin-antasari.ac.id
**6. Pelatihan Terkait** :

a. Data Analyst Student Generasi Gigih Batch 2.0, Yayasan Anak Bangsa Bisa
b. Jurnalisme dan Visualisasi Data, Fresh Graduate Academy Kominfo
c. Progate (SQL dan Python)
d. Dicoding (Data 101, Programming Logic 101, dan Dasar Visualisasi Data)
e. RevoU (Intro to Data Analytics)
f. kecerdasandigital.id CfDS UGM (Data Science for Social Good)

**7. Pengalaman Magang & Kerja:**

a. Pustakawan, SMPN 1 Gambut
b. Pustakawan, Universitas Nahdlatul Ulama Kalimantan Selatan
c. Pejuang Muda (Program MBKM), Kementerian Sosial Republik Indonesia
d. Divisi Ruang Terbuka, Pustakalana Children's Library (*remote internship*)

**8. Pengalaman Organisasi** :

a. Koordinator Verifikasi, Komisi Pemilihan Umum Mahasiswa FTK
b. Himpunan Mahasiswa Jurusan Ilmu Perpustakaan dan Informasi Islam
c. Linguistics and Debating Society Educia

**9. Prestasi Akademik** :

a. Juara I Karya Tulis Ilmiah lomba menulis dan presentasi esai berjudul, "Mahasiswa Masa Kini" yang diselenggarakan UKM KM-FEBI 2019.

Banjarmasin, 17 Februari 2023

Peneliti