# APPENDIX

## SUPPORT VECTOR MACHINE ONLY

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import numpy as np
from nltk.stem import PorterStemmer
df = pd.read_csv('data/tripadvisor_hotel_reviews.csv')
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords_list = set(stopwords.words("english"))
punctuations = """!()-![]{};:,+'"\,<>./?@#$%^&*_~Â"""
ps = PorterStemmer()
#PREPROCESSING
def reviewParse(review):
    splitReview = review.split() #Split review menjadi kata - kata
    parsedReview  =  "  ".join([word.translate(str.maketrans('',  '', punctuations)) + " " for word in splitReview])
    return parsedReview
 def clean_review(review):
    kata_bersih = []
    splitReview = review.split() #Split review menjadi kata - kata
    for w in splitReview:
        if w.isalpha() and w not in stopwords_list:
            word = ps.stem(w) #merubah kata menjadi kata dasar
            kata_bersih.append(word.lower())
    clean_review = " ".join(kata_bersih)
    return clean_review
df["Review"] = df["Review"].apply(reviewParse).apply(clean_review)
print(df.head())
df['Rating']=df['Rating'].astype(int)
conditions = [
    df.Rating >= 4,
    df.Rating == 3,
    df.Rating <= 2,
]
values = ['2', '1', '0']
df['label'] = np.select(conditions, values)
print("TFIDF Vectorizer……")
vectorizer= TfidfVectorizer()
X = vectorizer.fit_transform(df['Review'])
y=df['label']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,random_state=2)
print("Train:",X_train.shape,y_train.shape,"Test: ",(X_test.shape,y_test.shape))
from sklearn.svm import LinearSVC
clf = LinearSVC(random_state=0)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
```

```
print("Confusion Matrix SVM")
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
acu_svm=accuracy_score(y_test,y_pred)
print('AKURASI SVM: %.3f' % acu_svm)
```

## FEATURE SELECTION CHI - SQUARE

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
from nltk.stem import PorterStemmer
df = pd.read_csv('data/tripadvisor_hotel_reviews.csv')
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords_list = set(stopwords.words("english"))
punctuations = """!()-![]{};:,+'"\,<>./?@#$%^&*_~Â"""
ps = PorterStemmer()

#PREPROCESSING
def reviewParse(review):
    splitReview = review.split()
    parsedReview  =  "  ".join([word.translate(str.maketrans('',  '',
punctuations)) + " " for word in splitReview]) #Keluarin punctiation dari
data
    return parsedReview

def clean_review(review):
    kata_bersih = []
    splitReview = review.split()
    for w in splitReview:
        if w.isalpha() and w not in stopwords_list:
            word = ps.stem(w) #merubah kata menjadi kata dasar
            kata_bersih.append(word.lower())
    clean_review = " ".join(kata_bersih)
    return clean_review

df["Review"] = df["Review"].apply(reviewParse).apply(clean_review)
print(df.head())

df['Rating']=df['Rating'].astype(int)
conditions = [
    df.Rating >= 4,
    df.Rating == 3,
    df.Rating <= 2,
]
values = ['2', '1', '0']

df['label'] = np.select(conditions, values)

#Ekstraksi Fitur
print(df["Review"])
print("TFIDF Vectorizer……")
vectorizer= TfidfVectorizer()
X = vectorizer.fit_transform(df['Review'])
```

```
y=df['label']

#seleksi fitur
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
chi2_features = SelectKBest(chi2, k = 5000)
X_kbest_features = chi2_features.fit_transform(X, y)

#print("CHI SQUARE RESULT")
#print(X_kbest_features)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_kbest_features, y, test_size = 0.60,random_state=3)
print("Train:",X_train.shape,y_train.shape,"Test: ",(X_test.shape,y_test.shape))

from sklearn.svm import LinearSVC
clf = LinearSVC(random_state=0)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)

print("Confusion Matrix SVM")
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
from sklearn.metrics import accuracy_score
acu_svm=accuracy_score(y_test,y_pred)
print('AKURASI SVM: %.3f' % acu_svm)
```

## FEATURE SELECTION INFORMATION GAIN

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
from nltk.stem import PorterStemmer
df = pd.read_csv('data/tripadvisor_hotel_reviews.csv')
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords_list = set(stopwords.words("english"))
punctuations = """!()-![]{};:,+'"\,<>./?@#$%^&*_~Â"""
ps = PorterStemmer()

#PREPROCESSING
def reviewParse(review):
    splitReview = review.split()
    parsedReview = " ".join([word.translate(str.maketrans('', '', punctuations)) + " " for word in splitReview
    return parsedReview

def clean_review(review):
    kata_bersih = []
    splitReview = review.split()
    for w in splitReview:
        if w.isalpha() and w not in stopwords_list: #Cek apakah kata adalah alpabet dan tidak masiuk dalam stopword
```

```
            word = ps.stem(w) #merubah kata menjadi kata dasar
            kata_bersih.append(word.lower())
    clean_review = " ".join(kata_bersih)
    return clean_review

df["Review"] = df["Review"].apply(reviewParse).apply(clean_review)
print(df.head())

df['Rating']=df['Rating'].astype(int)
conditions = [
    df.Rating >= 4,
    df.Rating == 3,
    df.Rating <= 2,
]
values = ['2', '1', '0']

df['label'] = np.select(conditions, values)

#TF-IDF
print("TFIDF Vectorizer……")
vectorizer= TfidfVectorizer()
X = vectorizer.fit_transform(df['Review'])
#dfX=pd.DataFrame(X)
y=df['label']

from sklearn.feature_selection import mutual_info_classif as MIC
mi_score = MIC(X,y)
print("INFORMATION GAIN")
print(mi_score)
mi_score_selected_index = np.where(mi_score >-0.4)[0]
Xinfo = X[:,mi_score_selected_index]
print("HASIL INFORMATION GAIN")
print(Xinfo)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xinfo, y, test_size
  = 0.20,random_state=8)
print("Train:",X_train.shape,y_train.shape,"Test:
  ",(X_test.shape,y_test.shape))


from sklearn.svm import LinearSVC
clf = LinearSVC(random_state=0)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)

print("Confusion Matrix SVM")
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
from sklearn.metrics import accuracy_score
acu_svm=accuracy_score(y_test,y_pred)
print('AKURASI SVM: %.3f' % acu_svm)
```

PAPER NAME
18.K1.0027_Nathanael Karunia Wibowo

AUTHOR
Nathanael Karunia Wibowo

WORD COUNT
10656 Words

CHARACTER COUNT
53439 Characters

PAGE COUNT
19 Pages

FILE SIZE
53.2KB

SUBMISSION DATE
May 9, 2022 9:42 AM GMT+7

REPORT DATE
May 9, 2022 9:43 AM GMT+7

## 7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 4% Internet database
- Crossref database
- 3% Submitted Works database
- 4% Publications database
- Crossref Posted Content database

## Excluded from Similarity Report

- Bibliographic material
- Cited material
- Quoted material
- Small Matches (Less then 10 words)