

## APPENDIX

### SUPPORT VECTOR MACHINE ONLY

```
1. from sklearn.feature_extraction.text import TfidfVectorizer
2. import pandas as pd
3. import numpy as np
4. from nltk.stem import PorterStemmer
5. df = pd.read_csv('data/tripadvisor_hotel_reviews.csv')
6. import nltk
7. nltk.download('stopwords')
8. from nltk.corpus import stopwords
9. stopwords_list = set(stopwords.words("english"))
10. punctuations = ""!"()-![]{};:;+',\,<>./?@#$$%^&*~^"
11. ps = PorterStemmer()
12. #PREPROCESSING
13. def reviewParse(review):
14.     splitReview = review.split() #Split review menjadi kata - kata
15.     parsedReview = " ".join([word.translate(str.maketrans(' ', '',
16.     punctuations)) + " " for word in splitReview])
17.     return parsedReview
18. def clean_review(review):
19.     kata_bersih = []
20.     splitReview = review.split() #Split review menjadi kata - kata
21.     for w in splitReview:
22.         if w.isalpha() and w not in stopwords_list:
23.             word = ps.stem(w) #merubah kata menjadi kata dasar
24.             kata_bersih.append(word.lower())
25.     clean_review = " ".join(kata_bersih)
26.     return clean_review
27. df["Review"] = df["Review"].apply(reviewParse).apply(clean_review)
28. print(df.head())
29. df['Rating']=df['Rating'].astype(int)
30. conditions = [
31.     df.Rating >= 4,
32.     df.Rating == 3,
33.     df.Rating <= 2,
34. ]
35. values = ['2', '1', '0']
36. df['label'] = np.select(conditions, values)
37. print("TFIDF Vectorizer.....")
38. vectorizer= TfidfVectorizer()
39. X = vectorizer.fit_transform(df['Review'])
40. y=df['label']
41. from sklearn.model_selection import train_test_split
42. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
43.     0.20,random_state=2)
44. print("Train:",X_train.shape,y_train.shape,"Test:
45.     ",(X_test.shape,y_test.shape))
46. from sklearn.svm import LinearSVC
47. clf = LinearSVC(random_state=0)
48. clf.fit(X_train,y_train)
49. y_pred=clf.predict(X_test)
50. from sklearn.metrics import classification_report, confusion_matrix
51. from sklearn.metrics import accuracy_score
```

```

49. print("Confusion Matrix SVM")
50. print(confusion_matrix(y_test,y_pred))
51. print(classification_report(y_test,y_pred))
52. acu_svm=accuracy_score(y_test,y_pred)
53. print('AKURASI SVM: %.3f' % acu_svm)

```

## FEATURE SELECTION CHI - SQUARE

```

1. from sklearn.feature_extraction.text import TfidfVectorizer
2. import pandas as pd
3. import numpy as np
4. from sklearn.metrics import classification_report, confusion_matrix
5. from nltk.stem import PorterStemmer
6. df = pd.read_csv('data/tripadvisor_hotel_reviews.csv')
7. import nltk
8. nltk.download('stopwords')
9. from nltk.corpus import stopwords
10. stopwords_list = set(stopwords.words("english"))
11. punctuations = """"!()-![]{};:+'"\,<>./?@#$$%^&*~^_""
12. ps = PorterStemmer()
13.
14. #PREPROCESSING
15. def reviewParse(review):
16.     splitReview = review.split()
17.     parsedReview = " ".join([word.translate(str.maketrans('', '',
punctuations)) + " " for word in splitReview]) #Keluarin punctiation dari
data
18.     return parsedReview
19.
20. def clean_review(review):
21.     kata_bersih = []
22.     splitReview = review.split()
23.     for w in splitReview:
24.         if w.isalpha() and w not in stopwords_list:
25.             word = ps.stem(w) #merubah kata menjadi kata dasar
26.             kata_bersih.append(word.lower())
27.     clean_review = " ".join(kata_bersih)
28.     return clean_review
29.
30. df["Review"] = df["Review"].apply(reviewParse).apply(clean_review)
31. print(df.head())
32.
33. df['Rating']=df['Rating'].astype(int)
34. conditions = [
35.     df.Rating >= 4,
36.     df.Rating == 3,
37.     df.Rating <= 2,
38. ]
39. values = ['2', '1', '0']
40.
41. df['label'] = np.select(conditions, values)
42.
43. #Ekstraksi Fitur
44. print(df["Review"])
45. print("TFIDF Vectorizer.....")
46. vectorizer= TfidfVectorizer()
47. X = vectorizer.fit_transform(df['Review'])

```

```

48. y=df['label']
49.
50. #seleksi fitur
51. from sklearn.feature_selection import SelectKBest
52. from sklearn.feature_selection import chi2
53. chi2_features = SelectKBest(chi2, k = 5000)
54. X_kbest_features = chi2_features.fit_transform(X, y)
55.
56. #print("CHI SQUARE RESULT")
57. #print(X_kbest_features)
58.
59. from sklearn.model_selection import train_test_split
60. X_train, X_test, y_train, y_test = train_test_split(X_kbest_features, y,
    test_size = 0.60,random_state=3)
61. print("Train:",X_train.shape,y_train.shape,"Test:
    ",(X_test.shape,y_test.shape))
62.
63. from sklearn.svm import LinearSVC
64. clf = LinearSVC(random_state=0)
65. clf.fit(X_train,y_train)
66. y_pred=clf.predict(X_test)
67.
68. print("Confusion Matrix SVM")
69. print(confusion_matrix(y_test,y_pred))
70. print(classification_report(y_test,y_pred))
71. from sklearn.metrics import accuracy_score
72. acu_svm=accuracy_score(y_test,y_pred)
73. print('AKURASI SVM: %.3f' % acu_svm)

```

## FEATURE SELECTION INFORMATION GAIN

```

1. from sklearn.feature_extraction.text import TfidfVectorizer
2. import pandas as pd
3. import numpy as np
4. from sklearn.metrics import classification_report, confusion_matrix
5. from nltk.stem import PorterStemmer
6. df = pd.read_csv('data/tripadvisor_hotel_reviews.csv')
7. import nltk
8. nltk.download('stopwords')
9. from nltk.corpus import stopwords
10. stopwords_list = set(stopwords.words("english"))
11. punctuations = """! () -! [] {} ; : , + ' " \ , < > . / ? @ # $ % ^ & * _ ~ Â """"
12. ps = PorterStemmer()
13.
14. #PREPROCESSING
15. def reviewParse(review):
16.     splitReview = review.split()
17.     parsedReview = " ".join([word.translate(str.maketrans('', '',
    punctuations)) + " " for word in splitReview
18.     return parsedReview
19.
20. def clean_review(review):
21.     kata_bersih = []
22.     splitReview = review.split()
23.     for w in splitReview:
24.         if w.isalpha() and w not in stopwords_list: #Cek apakah kata
    adalah alpabet dan tidak masiuk dalam stopword

```

```

25.         word = ps.stem(w) #merubah kata menjadi kata dasar
26.         kata_bersih.append(word.lower())
27.     clean_review = " ".join(kata_bersih)
28.     return clean_review
29.
30. df["Review"] = df["Review"].apply(reviewParse).apply(clean_review)
31. print(df.head())
32.
33. df['Rating']=df['Rating'].astype(int)
34. conditions = [
35.     df.Rating >= 4,
36.     df.Rating == 3,
37.     df.Rating <= 2,
38. ]
39. values = ['2', '1', '0']
40.
41. df['label'] = np.select(conditions, values)
42.
43. #TF-IDF
44. print("TFIDF Vectorizer.....")
45. vectorizer= TfidfVectorizer()
46. X = vectorizer.fit_transform(df['Review'])
47. #dfX=pd.DataFrame(X)
48. y=df['label']
49.
50. from sklearn.feature_selection import mutual_info_classif as MIC
51. mi_score = MIC(X,y)
52. print("INFORMATION GAIN")
53. print(mi_score)
54. mi_score_selected_index = np.where(mi_score >-0.4)[0]
55. Xinfo = X[:,mi_score_selected_index]
56. print("HASIL INFORMATION GAIN")
57. print(Xinfo)
58.
59. from sklearn.model_selection import train_test_split
60. X_train, X_test, y_train, y_test = train_test_split(Xinfo, y, test_size
= 0.20,random_state=8)
61. print("Train:",X_train.shape,y_train.shape,"Test:
", (X_test.shape,y_test.shape))
62.
63.
64. from sklearn.svm import LinearSVC
65. clf = LinearSVC(random_state=0)
66. clf.fit(X_train,y_train)
67. y_pred=clf.predict(X_test)
68.
69. print("Confusion Matrix SVM")
70. print(confusion_matrix(y_test,y_pred))
71. print(classification_report(y_test,y_pred))
72. from sklearn.metrics import accuracy_score
73. 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 than 10 words)