

v02

March 20, 2024

```
[ ]: # Import packages
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.metrics import ConfusionMatrixDisplay

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[ ]: # Load Dataset
data0_path = "../data/dataset_rais.xlsx"
data1_path = "../data/dataset.xlsx"
data0 = pd.read_excel(data0_path)
data1 = pd.read_excel(data1_path)
```

```
[ ]: # Preview dataset
data0.head()
```

```
[ ]:                                     Tweets \
0  Sama banget Komentar ini dgn para pendukung se...
1  Ha ha ha sigundul penguasa ancol karena selama...
2  Kasihan Ibu ini jadi korban akibat dicuci otak...
3  Ha ha hakadrun pada stresssssss mengenai beber...
4  Kok sewot dgn Pidato Sambutan Bpk Joko Widodo ...

                                     clean_tweets  Social  Historical \
0  komentar dukung setia joko widodo presiden ri ...    1.0        0.0
1  sigundul kuasa ancol jilat gabenar buka jeroan...    1.0        0.0
2  kasihan korban akibat cuci otak kadrun tanggun...    0.0        0.0
3  hakadrun stres pimpin negara inggeris turun an...    1.0        0.0
4  sewot pidato sambut joko widodo presiden ri hu...    0.0        0.0

    Dehumanization  Accusation  Attack  Loyalty  Threat  HS  Non HS  \
0                0.0          0.0    0.0    0.0    0.0  0.0    1.0
1                0.0          1.0    0.0    0.0    0.0  1.0    0.0
```

2	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	1.0
4	1.0	0.0	0.0	0.0	0.0	0.0	1.0

	Anticipation	Trust	Joy	Anger	Disgust	Fear	Sadness	Surprise
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
[ ]: data1.head()
```

```
[ ]:                                Tweets \
```

0	Sama banget Komentar ini dgn para pendukung se...
1	Ha ha ha sigundul penguasa ancol karena selama...
2	Kasihani Ibu ini jadi korban akibat dicuci otak...
3	Ha ha hakadrn pada stressssssss mengenai beber...
4	Kok sewot dgn Pidato Sambutan Bpk Joko Widodo ...

	clean_tweets	Social	Historical	\
0	komentar dukung setia joko widodo presiden ri ...	1	0	
1	sigundul kuasa ancol jilat gabenar buka jeroan...	1	0	
2	kasihan korban akibat cuci otak kadrn tanggun...	0	0	
3	hakadrn stres pimpin negara inggeris turun an...	1	0	
4	sewot pidato sambut joko widodo presiden ri hu...	0	0	

	Dehumanization	Accusation	Attack	Loyalty	Threat	HS	Non HS	\
0	0	0	0.0	0	0	0	1	
1	0	1	0.0	0	0	1	0	
2	1	1	0.0	0	0	1	0	
3	1	0	0.0	0	0	0	1	
4	1	0	0.0	0	0	0	1	

	Anticipation	Trust	Joy	Anger	Disgust	Fear	Sadness	Surprise
0	0	1	0	0.0	0	0	0	0
1	0	0	0	0.0	1	0	0	0
2	0	0	0	0.0	0	0	1	0
3	0	1	0	0.0	0	0	0	0
4	1	0	0	0.0	0	0	0	0

```
[ ]: data = pd.concat([data0, data1])
data.describe()
```

```
[ ]:                                Social    Historical    Dehumanization    Accusation    Attack \
count    3001.000000    3001.000000    3001.000000    3001.000000    3001.000000
mean      0.392536      0.018660      0.563146      0.274575      0.013995
```

std	0.488396	0.135345	0.496079	0.446374	0.117491
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	0.000000	0.000000
75%	1.000000	0.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Loyalty	Threat	HS	Non HS	Anticipation \
count	3001.000000	3001.000000	3367.000000	3606.000000	3077.000000
mean	0.017328	0.039987	0.632611	0.510815	0.213845
std	0.130510	0.195961	0.482165	0.499952	0.410085
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	1.000000	0.000000
75%	0.000000	0.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Trust	Joy	Anger	Disgust	Fear \
count	3198.000000	3152.000000	3117.000000	3107.000000	3079.000000
mean	0.165729	0.133566	0.277831	0.291600	0.038974
std	0.371895	0.340239	0.448002	0.454572	0.193564
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Sadness	Surprise
count	3103.000000	3146.000000
mean	0.063809	0.084234
std	0.244452	0.277782
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1.000000	1.000000

```
[ ]: data.fillna(0, inplace=True)
data.dropna(inplace=True)
data.describe()
```

	Social	Historical	Dehumanization	Accusation	Attack \
count	3972.000000	3972.000000	3972.000000	3972.000000	3972.000000
mean	0.296576	0.014099	0.425478	0.207452	0.010574
std	0.456805	0.117913	0.494478	0.405533	0.102298
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000

50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	0.000000	1.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Loyalty	Threat	HS	Non HS	Anticipation \
count	3972.000000	3972.000000	3972.000000	3972.000000	3972.000000
mean	0.013092	0.030211	0.536254	0.463746	0.165660
std	0.113682	0.171190	0.498747	0.498747	0.371822
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	0.000000	0.000000
75%	0.000000	0.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Trust	Joy	Anger	Disgust	Fear \
count	3972.000000	3972.000000	3972.000000	3972.000000	3972.000000
mean	0.133434	0.105992	0.218026	0.228097	0.030211
std	0.340086	0.307866	0.412957	0.419658	0.171190
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Sadness	Surprise
count	3972.000000	3972.000000
mean	0.049849	0.066717
std	0.217660	0.249563
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1.000000	1.000000

```
[ ]: #Helper function to plot the dataset distribution
def plot_data(data: pd.DataFrame):
    plt.figure(figsize=(12, 12))
    y0_names = ["Non HS", "HS"]
    y1_names = [
        "Anger",
        "Anticipation",
        "Disgust",
        "Fear",
        "Joy",
        "Sadness",
        "Surprise",
        "Trust",
    ]
```

```

]

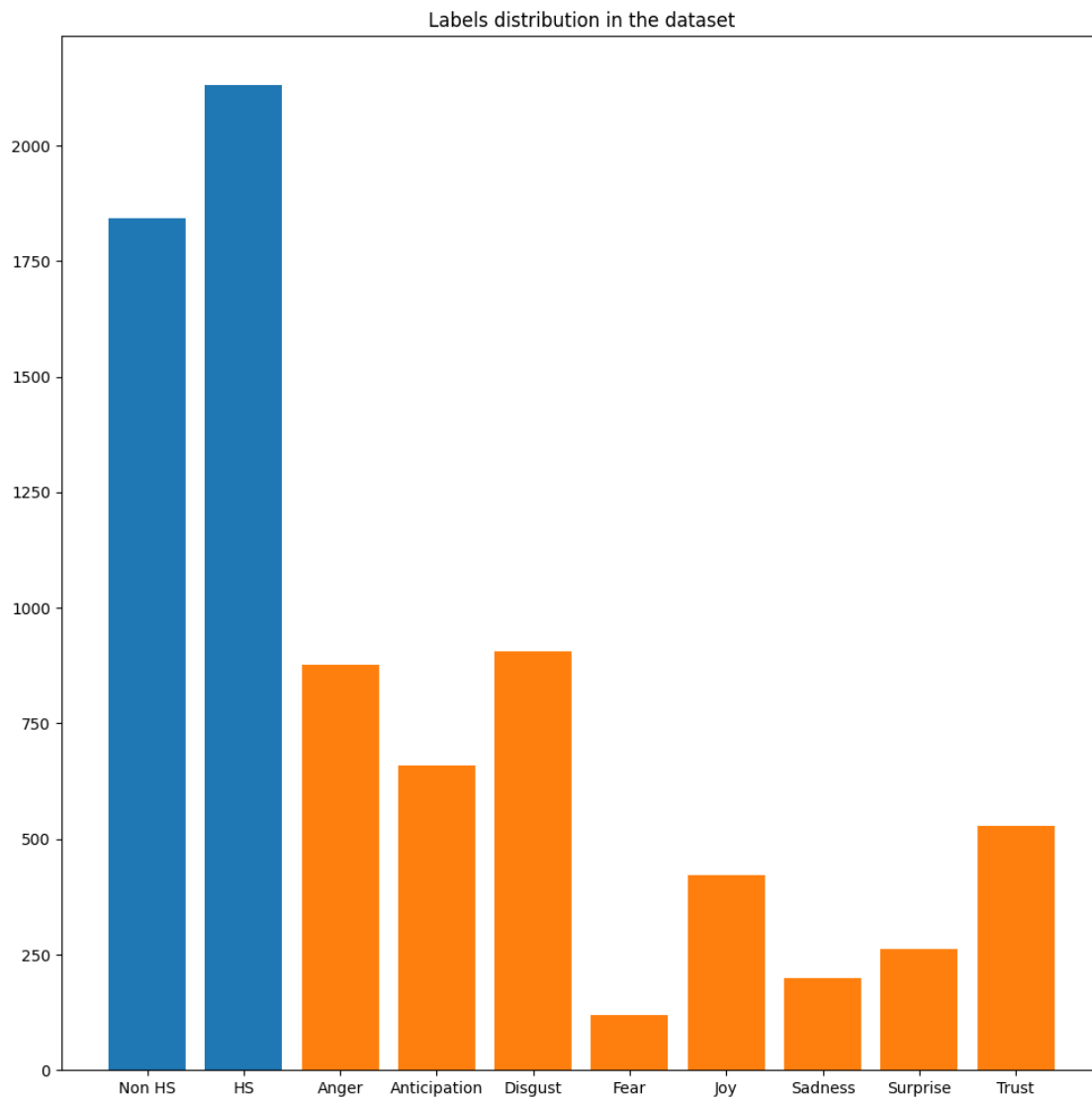
y0 = data.loc[:, y0_names].values.argmax(1) # HS
y1 = data.loc[:, y1_names].values.argmax(1) # Emotion

uniq0 = np.unique(y0, return_counts=True)
uniq1 = np.unique(y1, return_counts=True)

total0 = {y0_names[x]: y for (x, y) in zip(uniq0[0], uniq0[1])}
total1 = {y1_names[x]: y for (x, y) in zip(uniq1[0], uniq1[1])}

plt.bar(total0.keys(), total0.values())
plt.bar(total1.keys(), total1.values())
plt.title('Labels distribution in the dataset')
plot_data(data)

```



```

[ ]: # Helper function to pre-process dataset
def preprocess_data(data: pd.DataFrame):
    # Get only the needed datas
    y0_names = ["Non HS", "HS"]
    y1_names = [
        "Anger",
        "Anticipation",
        "Disgust",
        "Fear",
        "Joy",
        "Sadness",
        "Surprise",
        "Trust",
    ]

    X = data["clean_tweets"].values # Input
    y0 = data.loc[:, y0_names].values.argmax(1) # HS
    y1 = data.loc[:, y1_names].values.argmax(1) # Emotion

    # Split for train and test
    ratio = int(0.8 * len(X))
    split_data = lambda x: [x[:ratio], x[ratio:]]

    x_train, x_test = split_data(X)
    y0_train, y0_test = split_data(y0)
    y1_train, y1_test = split_data(y1)

    # TF-IDF for Feature Extraction
    tfidf = TfidfVectorizer()
    x_train = tfidf.fit_transform(x_train)
    x_test = tfidf.transform(x_test)

    feature_names = tfidf.get_feature_names_out()

    ret = {
        "x_train": x_train,
        "x_test": x_test,
        "y0_train": y0_train,
        "y0_test": y0_test,
        "y1_train": y1_train,
        "y1_test": y1_test,
        "y0_names": y0_names,
        "y1_names": y1_names,
        "feature_names": feature_names,
    }

```

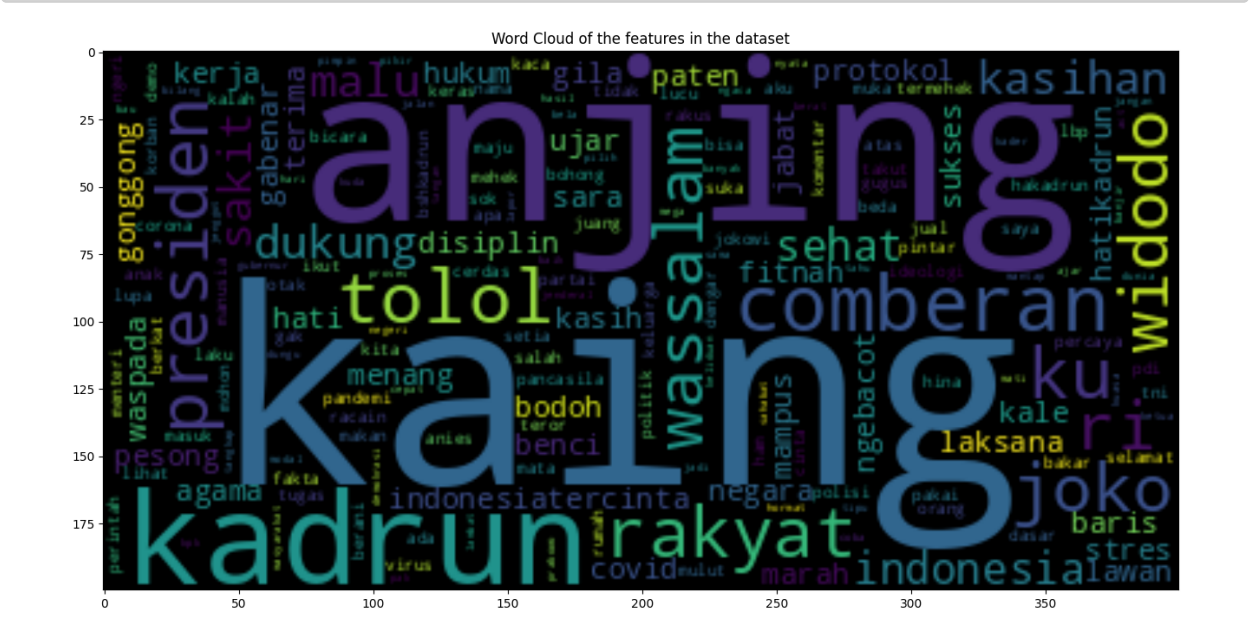
```
return tfidf, ret
```

```
# PreProcess the data
tf_idf, data_dict = preprocess_data(data)
```

```
denselist = data_dict['x_train'].todense().tolist()
df_ = pd.DataFrame(denselist, columns=tf_idf.get_feature_names_out())
word_count = dict(df_.sum(axis=0))
word_count = {x: int(y*3000) for (x, y) in word_count.items()}
```

```
from wordcloud import WordCloud

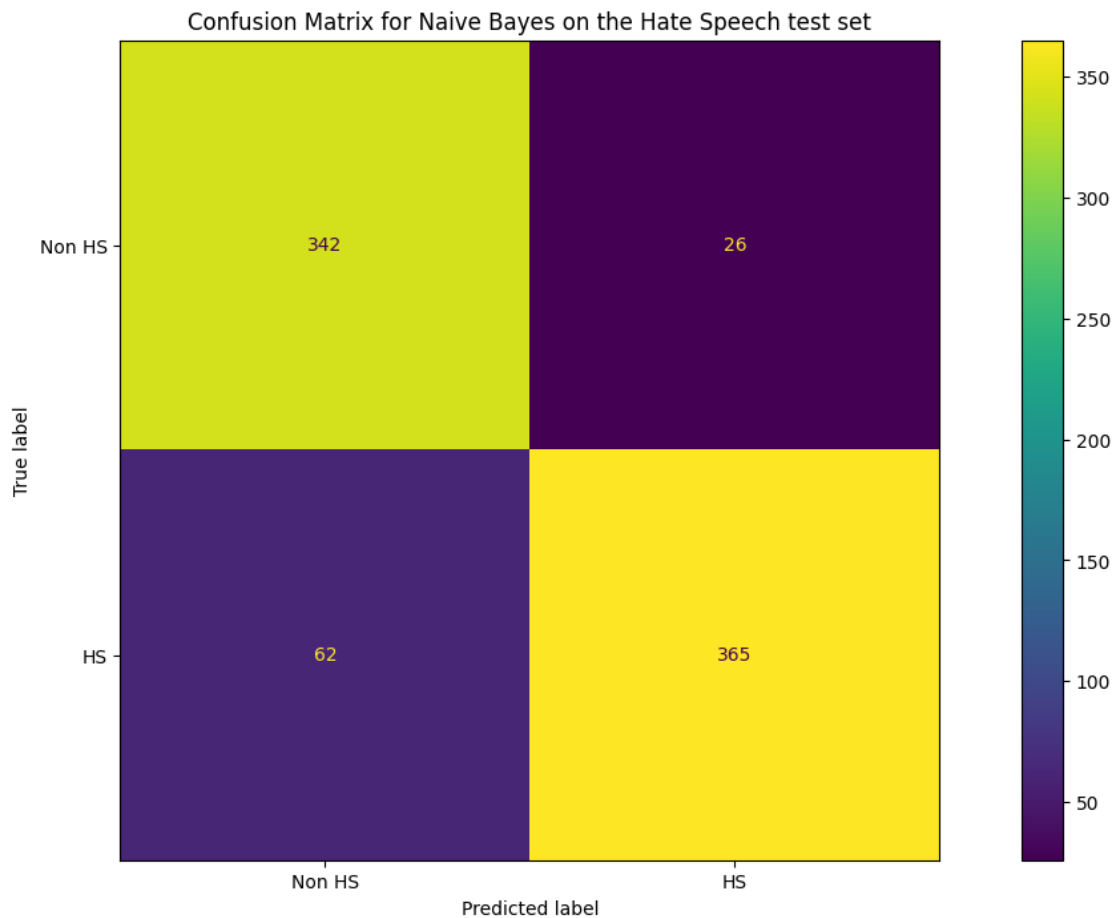
wc = WordCloud().generate_from_frequencies(word_count)
plt.figure(figsize=(16, 16))
plt.title('Word Cloud of the features in the dataset')
plt.imshow(wc, interpolation='bilinear')
plt.show()
```



```
# Hate Speech Classification
hs_clf = MultinomialNB()
hs_clf.fit(data_dict["x_train"], data_dict["y0_train"])
pred = hs_clf.predict(data_dict["x_test"])
print(
    f'Mean Accuracy for Hate Speech Classification: {hs_clf.
    ↪score(data_dict["x_test"], data_dict["y0_test"])}'
)
```

Mean Accuracy for Hate Speech Classification: 0.889308176100629

```
[ ]: # Plot the Confusion Matrix in test data
fig, ax = plt.subplots(figsize=(16, 8))
ConfusionMatrixDisplay.from_predictions(data_dict["y0_test"], pred, ax=ax)
ax.xaxis.set_ticklabels(data_dict["y0_names"])
ax.yaxis.set_ticklabels(data_dict["y0_names"])
_ = ax.set_title(f"Confusion Matrix for Naive Bayes on the Hate Speech test_
↪set")
```



```
[ ]: # Helper function to plot feature effects
def plot_feature_effects(clf, X_train, target_names, feature_names):
    # learned coefficients weights
    average_feature_effects = (
        clf.feature_log_prob_**2 * np.asarray(X_train.mean(axis=0)).ravel()
    )

    for i, label in enumerate(target_names[:-1]):
```



```

top5 = np.argsort(average_feature_effects[i])[-5:] [::-1]
if i == 0:
    top = pd.DataFrame(feature_names[top5], columns=[label])
    top_indices = top5
else:
    top[label] = feature_names[top5]
    top_indices = np.concatenate((top_indices, top5), axis=None)
top_indices = np.unique(top_indices)
predictive_words = feature_names[top_indices]

# plot feature effects
bar_size = 0.25
padding = 0.75
y_locs = np.arange(len(top_indices)) * (4 * bar_size + padding)

fig, ax = plt.subplots(figsize=(12, 12))
for i, label in enumerate(target_names[:, -1]):
    ax.barh(
        y_locs + (i - 2) * bar_size,
        average_feature_effects[i, top_indices],
        height=bar_size,
        label=label,
    )
ax.set(
    yticks=y_locs,
    yticklabels=predictive_words,
    ylim=[
        0 - 4 * bar_size,
        len(top_indices) * (4 * bar_size + padding) - 4 * bar_size,
    ],
)
ax.legend(loc="lower right")

print("top 5 keywords per class:")
print(top)

return ax

```

```

[ ]: plot_feature_effects(
    hs_clf, data_dict["x_train"], data_dict["y0_names"],
    ↪data_dict["feature_names"]
)

```

top 5 keywords per class:

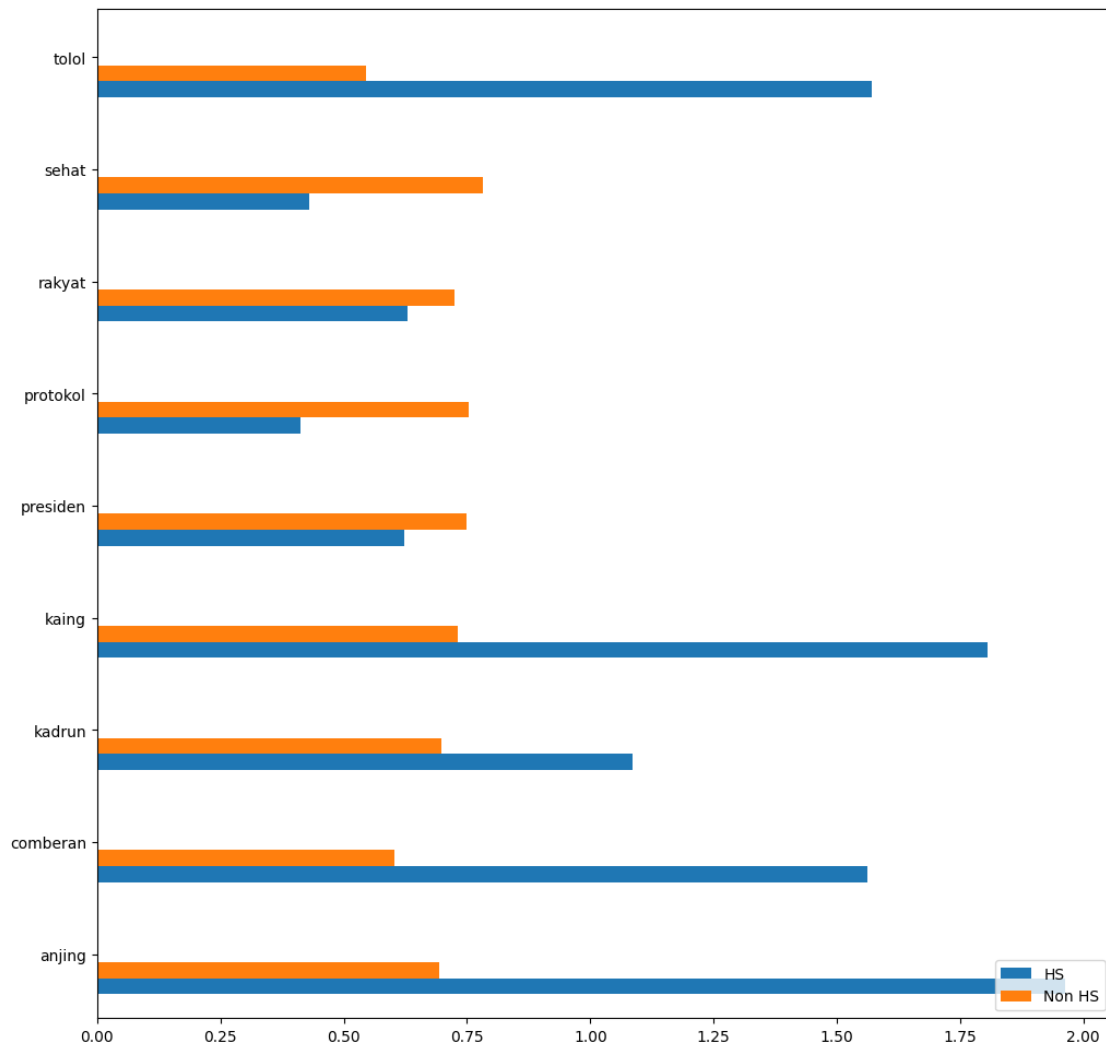
	HS	Non HS
0	anjing	sehat
1	kaing	protokol

```

2     tolol presiden
3 comberan   kaing
4   kadrunk  rakyat

```

```
[ ]: <Axes: >
```



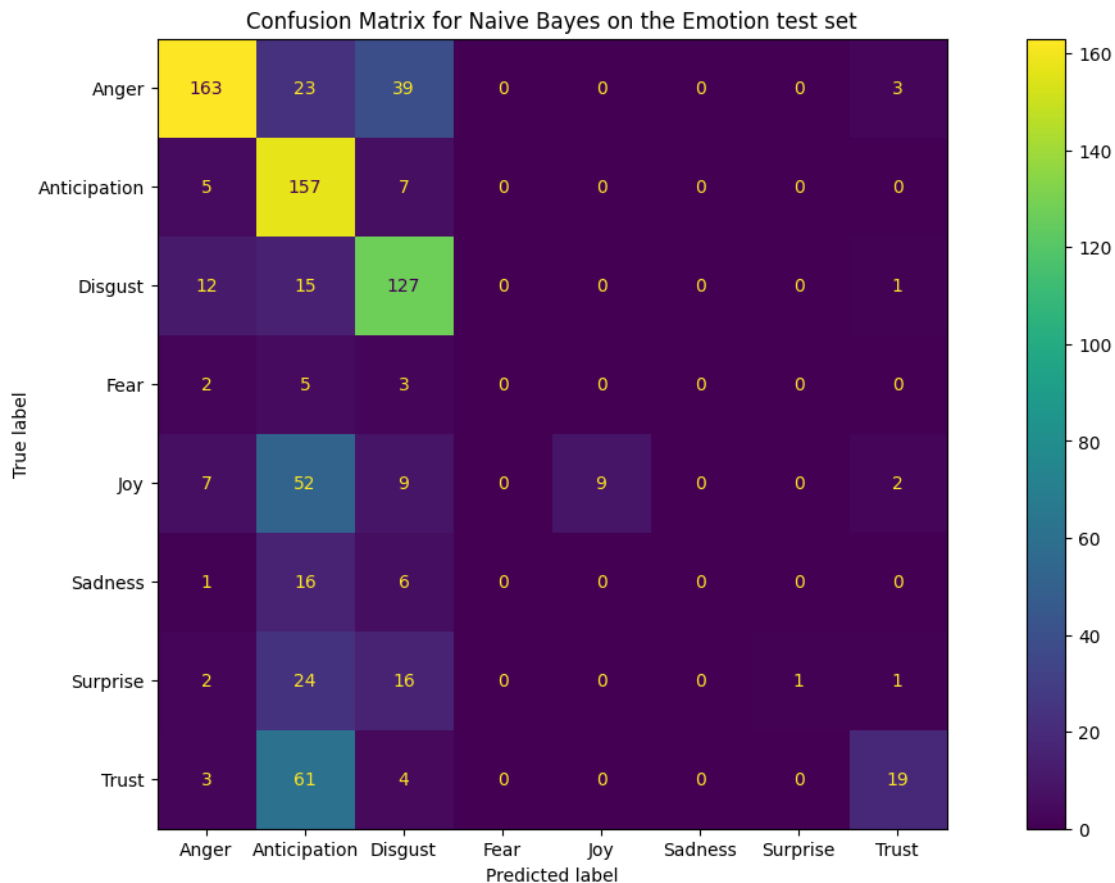
```

[ ]: # Emotion Classification
em_clf = MultinomialNB()
em_clf.fit(data_dict["x_train"], data_dict["y1_train"])
pred = em_clf.predict(data_dict["x_test"])
print(
    f'Mean Accuracy for Emotion Classification: {em_clf.
    ↪score(data_dict["x_test"], data_dict["y1_test"])}'
)

```

Mean Accuracy for Emotion Classification: 0.5987421383647799

```
[ ]: # Plot the Confusion Matrix in test data
fig, ax = plt.subplots(figsize=(16, 8))
ConfusionMatrixDisplay.from_predictions(data_dict["y1_test"], pred, ax=ax)
ax.xaxis.set_ticklabels(data_dict["y1_names"])
ax.yaxis.set_ticklabels(data_dict["y1_names"])
_ = ax.set_title(f"Confusion Matrix for Naive Bayes on the Emotion test set")
```



```
[ ]: plot_feature_effects(
    em_clf, data_dict["x_train"], data_dict["y1_names"],
    data_dict["feature_names"]
)
```

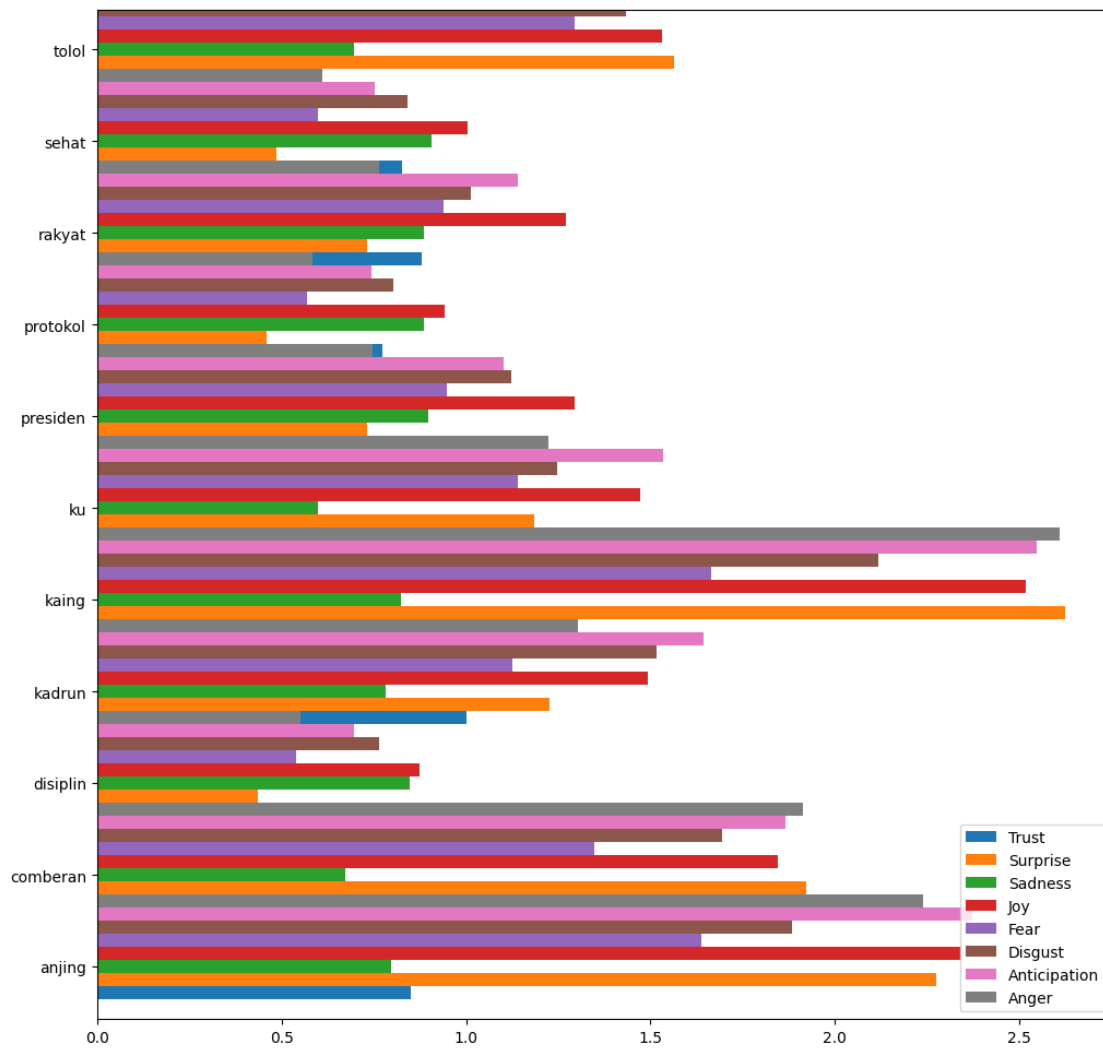
top 5 keywords per class:

	Trust	Surprise	Sadness	Joy	Fear	Disgust	Anticipation
0	kadrun	kaing	sehat	kaing	kaing	kaing	kaing
1	presiden	anjing	presiden	anjing	anjing	anjing	anjing
2	kaing	comberan	protokol	comberan	comberan	comberan	comberan
3	rakyat	tolol	rakyat	tolol	tolol	kadrun	kadrun

4 anjing kadrun disiplin kadrun ku tolol ku

Anger
0 kaing
1 anjing
2 comberan
3 tolol
4 kadrun

[]: <Axes: >



1 Prediction

```
[ ]: #Helper function to run prediction
def predict(txt_input: str,
            clf_type: str):

    #Input Text
    pre_processed_text = tf_idf.transform([txt_input])

    if clf_type not in ['hs', 'emotion']:
        raise f'clf_type should be either hs or emotion, got {clf_type}'

    if clf_type == 'hs':
        return data_dict['y0_names'][hs_clf.predict(pre_processed_text)[0]]
    else:
        return data_dict['y1_names'][em_clf.predict(pre_processed_text)[0]]

[ ]: text = 'negara gagal mengentaskan kemiskinan lembaga pendidikan semena mena_
↳ mencabut hak peserta didik memperoleh pendidikan bukan cuma sekali dua_
↳ terjadi cuma tidak tersorot berita'

#Start Predict
predict(text, 'hs')

[ ]: 'HS'
```