# v01

February 11, 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
data_path = "../data/dataset.xlsx"
data = pd.read_excel(data_path)
```

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

[ ]:

| | Tweets | clean_tweets | Social | Historical | Dehumanization | Accusation | Attack | Loyalty | Threat | HS | Non HS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Sama banget Komentar ini dgn para pendukung se… | komentar dukung setia joko widodo presiden ri … | 1 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 1 |
| 1 | Ha ha ha sigundul penguasa ancol karena selama… | sigundul kuasa ancol jilat gabenar buka jeroan… | 1 | 0 | 0 | 1 | 0.0 | 0 | 0 | 1 | 0 |
| 2 | Kasihan Ibu ini jadi korban akibat dicuci otak… | kasihan korban akibat cuci otak kadrun tanggun… | 0 | 0 | 1 | 1 | 0.0 | 0 | 0 | 1 | 0 |
| 3 | Ha ha hakadrun pada stressssss mengenai beber… | hakadrun stres pimpin negara inggeris turun an… | 1 | 0 | 1 | 0 | 0.0 | 0 | 0 | 0 | 1 |
| 4 | Kok sewot dgn Pidato Sambutan Bpk Joko Widodo … | sewot pidato sambut joko widodo presiden ri hu… | 0 | 0 | 0 | 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.describe()
```

[ ]:
```
            Social   Historical  Dehumanization   Accusation      Attack  \
count  1500.000000  1500.000000     1500.000000  1500.000000  1500.00000
mean      0.392667     0.018667        0.563333     0.274667     0.01400
std       0.488507     0.135390        0.496138     0.446495     0.11753
min       0.000000     0.000000        0.000000     0.000000     0.00000
25%       0.000000     0.000000        0.000000     0.000000     0.00000
50%       0.000000     0.000000        1.000000     0.000000     0.00000
75%       1.000000     0.000000        1.000000     1.000000     0.00000
max       1.000000     1.000000        1.000000     1.000000     1.00000

           Loyalty       Threat           HS       Non HS  Anticipation  \
count  1500.000000  1500.000000  1500.000000  1500.000000   1500.000000
mean      0.017333     0.040000     0.588000     0.412000      0.194000
std       0.130554     0.196025     0.492359     0.492359      0.395561
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  1500.000000  1500.000000  1499.000000  1500.000000  1500.00000
mean      0.110667     0.090000     0.249500     0.266667     0.01400
std       0.313824     0.286277     0.432868     0.442364     0.11753
min       0.000000     0.000000     0.000000     0.000000     0.00000
25%       0.000000     0.000000     0.000000     0.000000     0.00000
50%       0.000000     0.000000     0.000000     0.000000     0.00000
75%       0.000000     0.000000     0.000000     1.000000     0.00000
max       1.000000     1.000000     1.000000     1.000000     1.00000

           Sadness     Surprise
count  1500.000000  1500.000000
mean      0.032000     0.040000
std       0.176059     0.196025
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 = data.dropna()
data.describe()
```

[ ]:
```
            Social   Historical  Dehumanization   Accusation       Attack  \
count  1499.000000  1499.000000     1499.000000  1499.000000  1499.000000
mean      0.392262     0.018679        0.563709     0.274850     0.014009
std       0.488417     0.135434        0.496090     0.446588     0.117568
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  1499.000000  1499.000000  1499.000000  1499.000000   1499.000000
mean      0.017345     0.040027     0.588392     0.411608      0.194129
std       0.130596     0.196087     0.492289     0.492289      0.395661
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  1499.000000  1499.000000  1499.000000  1499.000000  1499.000000
mean      0.110740     0.090060     0.249500     0.266845     0.014009
std       0.313915     0.286363     0.432868     0.442458     0.117568
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     1.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

           Sadness     Surprise
count  1499.000000  1499.000000
mean      0.032021     0.039360
std       0.176116     0.194514
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.loc[:, ["Non HS", "HS"]].values
```

[ ]:
```
array([[1, 0],
       [0, 1],
       [0, 1],
       …,
       [1, 0],
       [0, 1],
       [0, 1]])
```

[ ]:
```
# 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,
```

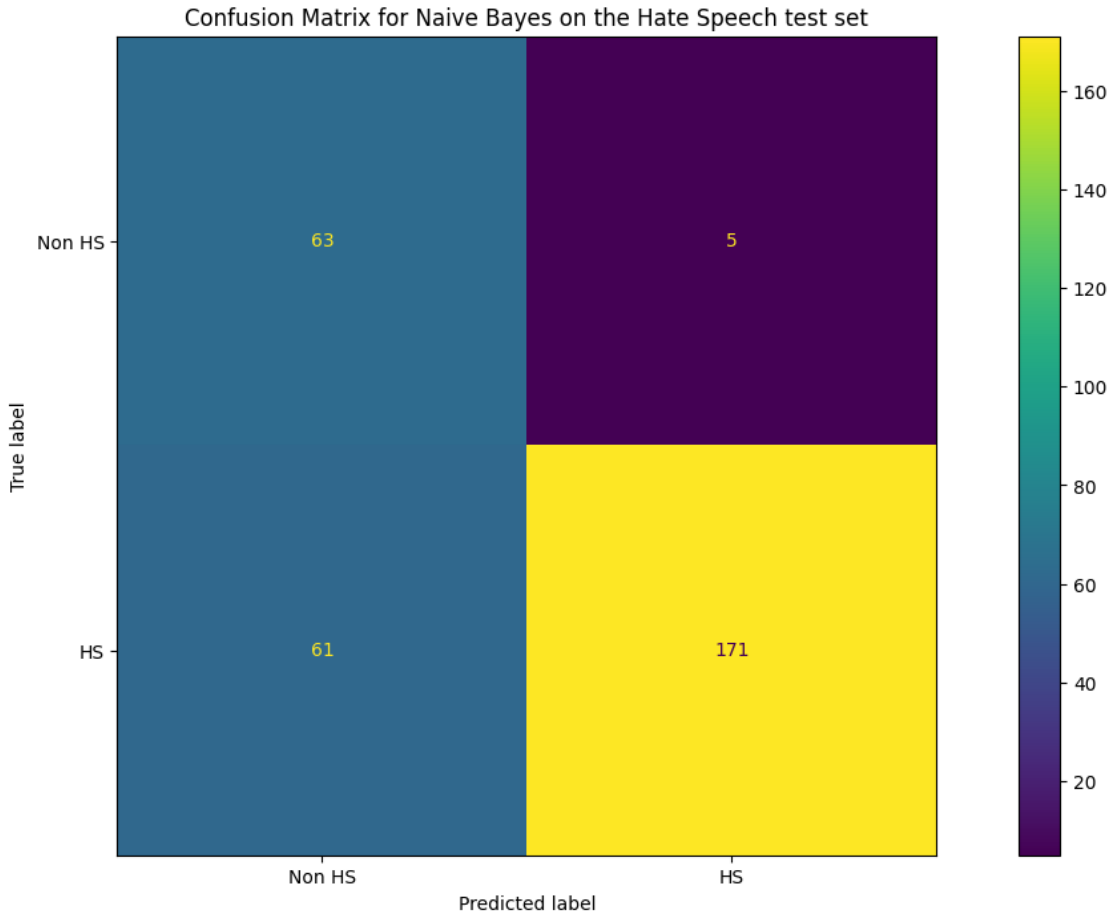
```
        "y0_train": y0_train,
        "y0_test": y0_test,
        "y1_train": y1_train,
        "y1_test": y1_test,
        "y0_names": y0_names,
        "y1_names": y1_names,
        "feature_names": feature_names,
    }

    return tfidf, ret
```

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

[ ]:
```
# 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: 0.78
```

[ ]:
```
# 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")
```



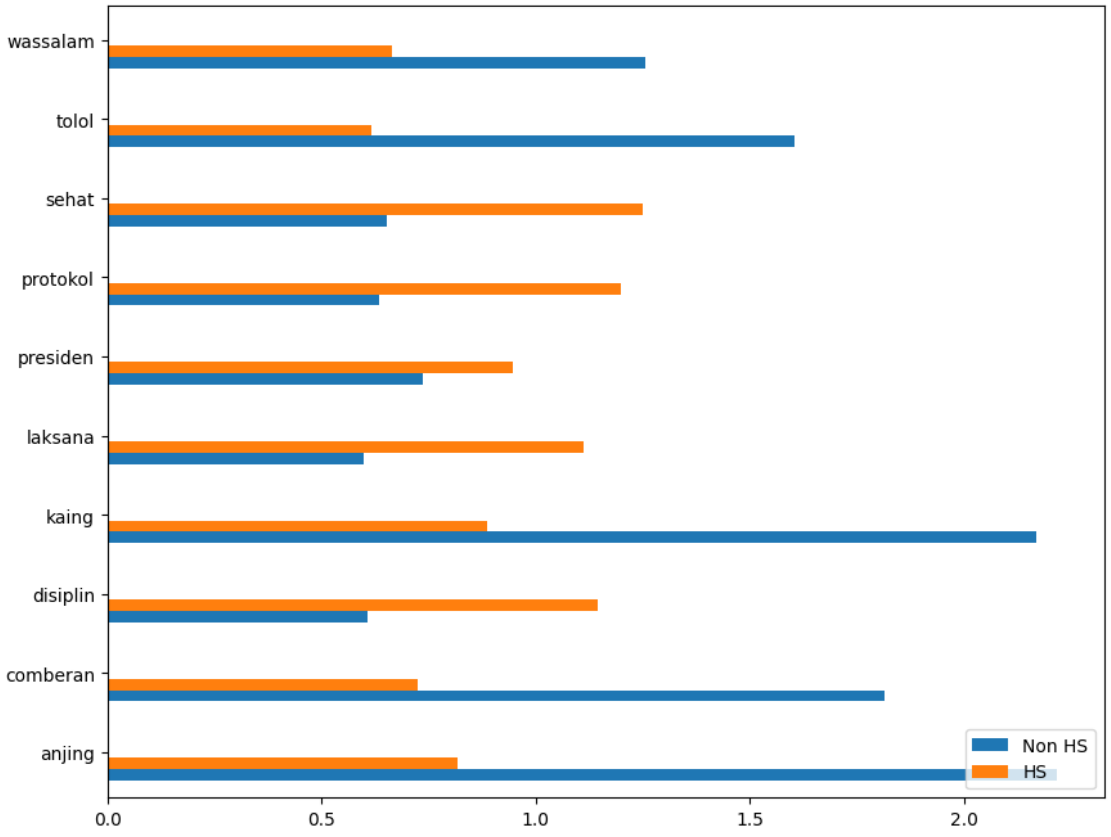

[ ]:
```python
# Helper function to plot feature effects
def plot_feature_effects(clf, X_train, target_names, feature_names):
    # learned coefficients weights
    average_feature_effects = (
        clf.feature_log_prob_**2 * np.asarray(X_train.mean(axis=0)).ravel()
    )

    for i, label in enumerate(target_names):
        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=(10, 8))
    for i, label in enumerate(target_names):
        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:
     Non HS        HS
0    anjing     sehat
1     kaing  protokol
2  comberan  disiplin
3     tolol   laksana
4  wassalam  presiden
```

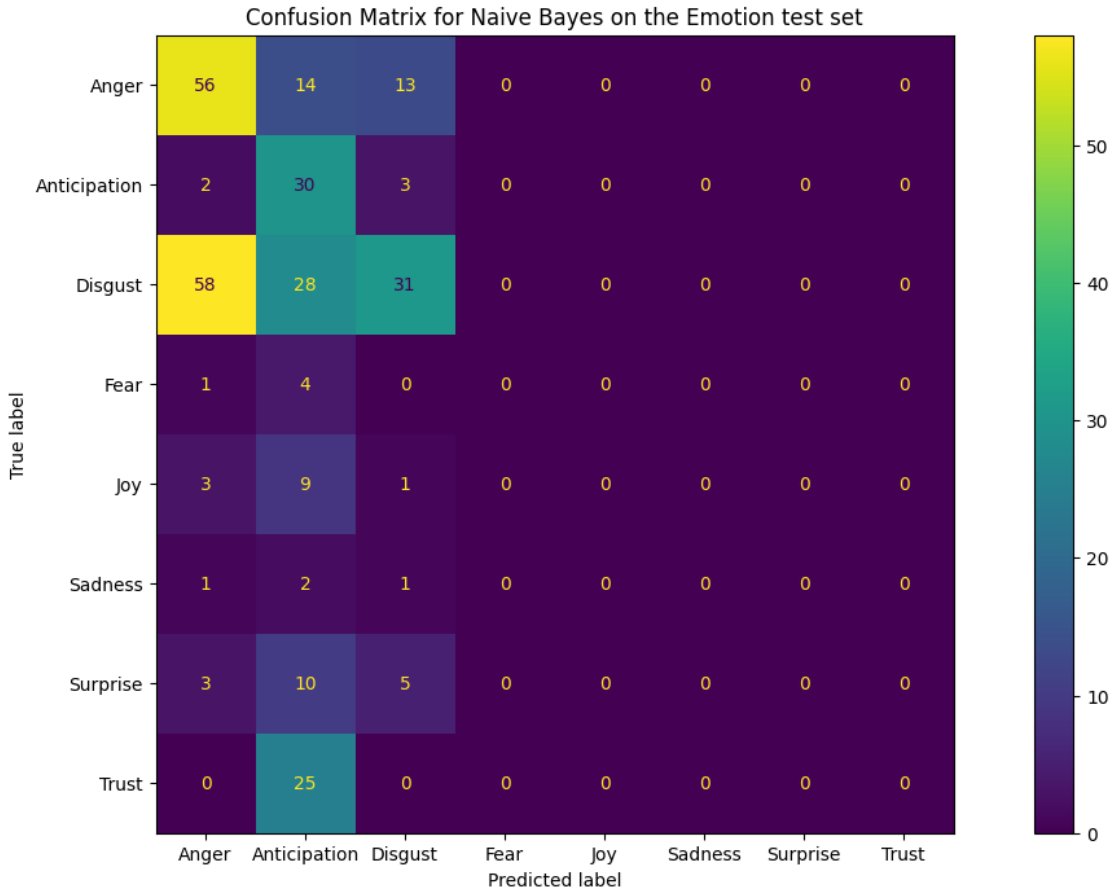
[ ]:
```
<Axes: >
```




[ ]:
```
# 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: 0.39
```

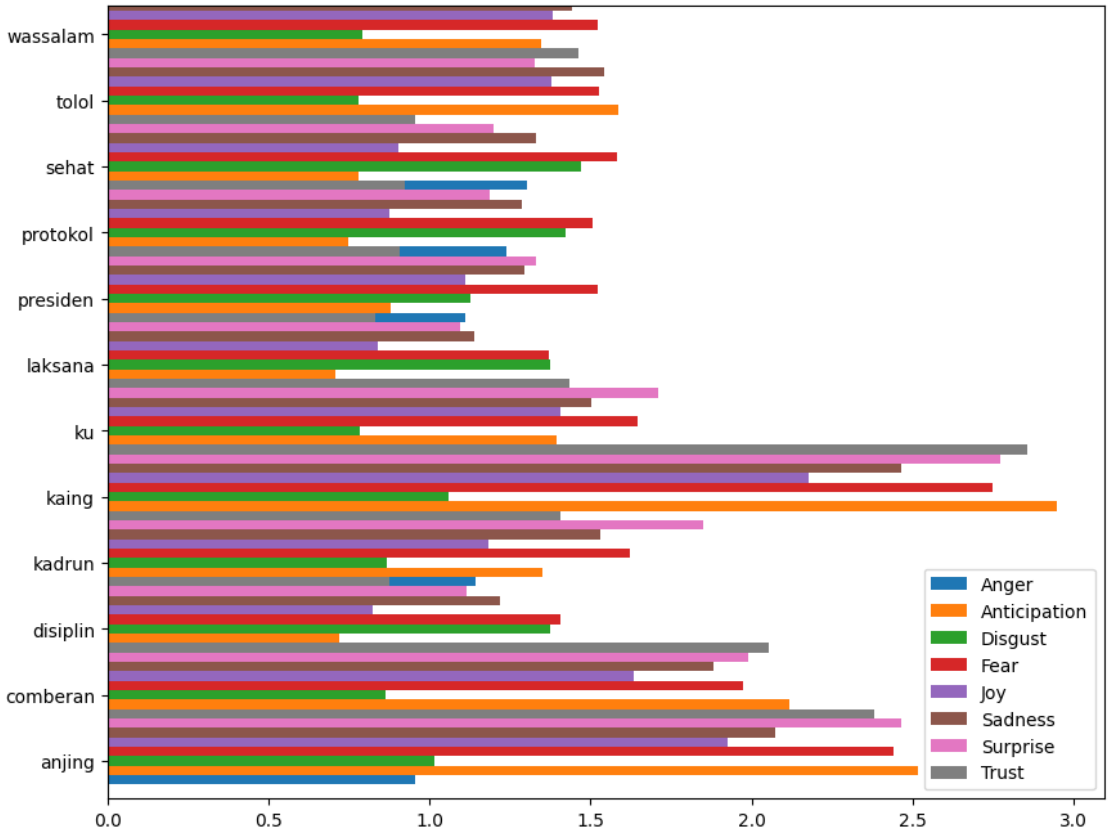
[ ]:
```
# Plot the Confusion Matrix in test data
fig, ax = plt.subplots(figsize=(16, 8))
ConfusionMatrixDisplay.from_predictions(data_dict["y1_test"], pred, ax=ax)
ax.xaxis.set_ticklabels(data_dict["y1_names"])
ax.yaxis.set_ticklabels(data_dict["y1_names"])
_ = ax.set_title(f"Confusion Matrix for Naive 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:
      Anger Anticipation   Disgust      Fear       Joy   Sadness  Surprise  \
0     sehat        kaing     sehat     kaing     kaing     kaing     kaing
1  protokol       anjing  protokol    anjing    anjing    anjing    anjing
2  disiplin     comberan   laksana  comberan  comberan  comberan  comberan
3    kadrun        tolol  disiplin        ku        ku     tolol    kadrun
4   laksana           ku  presiden    kadrun  wassalam    kadrun        ku

      Trust
0     kaing
1    anjing
2  comberan
3    tolol
4        ku
```

[ ]: <Axes: >


wassalam
tolol
sehat
protokol
presiden
laksana
ku
kaing
kadrun
disiplin
comberan
anjing
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Anger
Anticipation
Disgust
Fear
Joy
Sadness
Surprise
Trust