Nicholas Juan Kalvin P._162012133068_TEXT_PROCESSING

November 2, 2022

1 Tugas Praktikum Minggu 8

1.0.1 Nicholas Juan Kalvin P. | 162012133068

2 Text Classification

2.1 Tugas Praktikum:

Download data teks dari halaman berikut: https://raw.githubusercontent.com/ruzcmc/ClickbaitIndo-textclassifier/master/all_agree.csv

Lakukan classification menggunakan metode MultinomialNB, RandomForest, dan metode klasifikasi pilihan kalian (terserah) dengan menggunakan fitur TF-IDF

Buat perbandingan performa ketiga metode supervised learning tersebut berdasarkan performanya (poin plus: lakukan cross validation)

2.2 Requirements

[ ]:
```
# Module imports
from Sastrawi.StopWordRemover.StopWordRemoverFactory import␣
 ↪StopWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from pyarrow import csv
import requests
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBRFClassifier
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
```

Pada praktikum kali ini, digunakan 4 library penting untuk melakukan mayoritas dari praktikum.

cv2 atau pip:popencv-python

adalah library yang sangat populer untuk visualisasi, dari visualisasi menggunakan kamera hingga visualisasi gambar digital. Kali ini cv2 digunakan untuk membedah gambar ke beberapa channel untuk tujuan perhitungan.

sklearn adalah library utama dari praktikum ini, sklearn melayani berbagai fungsi statistika hingga pemodelan. Dimasukkan fungsi PCA, StandardScaler, Kmeans dan make_blobs yang bertujuan untuk analisis, prosesing hingga pemodelan data

matplotlib adalah library populer untuk melakukan visualisasi data menjadi grafik

2.3 Data Loading

2.3.1 Loading Step 1: Get from GitHub

[ ]:
```
data_link = 'https://raw.githubusercontent.com/ruzcmc/
 ↪ClickbaitIndo-textclassifier/master/all_agree.csv'
r = requests.get(data_link)
content = r.content.decode('utf-8')
with open("all_agree.csv", "a") as f:
    f.write(content)
    f.close()
```

2.3.2 Loading Step 2: Load as DataFrame

[ ]:
```
read_options = csv.ReadOptions(use_threads=True, encoding='unicode_escape')
padf = csv.read_csv('all_agree.csv', read_options=read_options)
df = padf.to_pandas()
```

2.4 Data Cleaning

[ ]:
```
print(df.columns)
print(df.describe())
print(df.isna())
df = df.drop(columns='label')
df.head(5)
```

```
Index(['title', 'label', 'label_score'], dtype='object')
                                                    title          label  \
count                                              189507         189507
unique                                               8603              3
top     Diawasi di Hong Kong, Aktivis Joshua Wong Terb…  non-clickbait
freq                                                   66         116534
```

```
       label_score
count       189507
unique           3
top              0
freq        116534
        title  label  label_score
0       False  False        False
1       False  False        False
2       False  False        False
3       False  False        False
4       False  False        False
…       …    …           …
189502  False  False        False
189503  False  False        False
189504  False  False        False
189505  False  False        False
189506  False  False        False

[189507 rows x 3 columns]
```

[ ]:
```
                                               title label_score
0  Masuk Radar Pilwalkot Medan, Menantu Jokowi Be…           0
1  Malaysia Sudutkan RI: Isu Kabut Asap hingga In…           0
2  Viral! Driver Ojol di Bekasi Antar Pesanan Mak…           1
3  Kemensos Salurkan Rp 7,3 M bagi Korban Kerusuh…           0
4  MPR: Amandemen UUD 1945 Tak Akan Melebar ke Ma…           0
```

[ ]:
```
df.duplicated().sum()
```

[ ]:
```
180903
```

Dari fungsi diatas, diketahui banyak data terduplikasi yang akan lebih baik jika dihilangkan

[ ]:
```
print(df.duplicated().value_counts())
df = df.drop_duplicates()
df.duplicated().sum()
df['label_score'].value_counts()
df.loc[df['label_score']=='label_score']
df.drop([8613], inplace=True)
```

```
True     180903
False      8604
dtype: int64
```

Codeblock diatas bertugas untuk menghilangkan duplikat, dan juga menghapus salah satu baris dalam dataset yang mengandung nilai-nilai kolom dan bukan nilai observasi sesungguhnya.

2.5 Text Cleaning

2.5.1 Text Cleaning: Cleaner Functions

[ ]:
```
import re, string

def bersih_text(text):
    text = str(text)
    text = text.lower()
    text = re.sub("@[A-Za-z0-9_]+", "", text)  # Remove Twitter Mentions
    text = re.sub("#\w+", "", text)
    text = re.sub("\[.*?\]", "", text)
    text = re.sub("https?://\S+|www\.\S+", "", text)
    text = re.sub("<.*?>+", "", text)
    text = re.sub("[%s]" % re.escape(string.punctuation), "", text)
    text = re.sub('\n', '', text)
    text = re.sub("\w*\d\w*", "", text)
    text = re.sub("\d+", "", text)
    text = re.sub("\s+", " ", text).strip()
    text = " ".join(text.split())
    return text

df['title'] = [bersih_text(t) for t in df['title']]
df
```

Menggunakan Regex, dapat dipilih banyak set karakter yang tidak diinginkan untuk dilakukan penghapusan. Fungsi diatas adalah fungsi yang menyimpan banyak ekspresi regex yang bertugas untuk membersihkan semua karakter yang jarang dibutuhkan untuk klasifikasi teks

2.6 Text Preprocessing

2.6.1 Text Processing step 1: Remove Stopwords

[ ]:
```
sw_factory = StopWordRemoverFactory()
stopword = sw_factory.create_stop_word_remover()

df['title'] = [stopword.remove(kalimat) for i, kalimat in enumerate␣
 ↪(df['title'])]
df['title']
```

[ ]:
```
0       radar pilwalkot medan menantu jokowi bertemu d…
1         malaysia sudutkan ri isu kabut asap invasi babi
2       viral driver ojol bekasi pesanan makanan pakai…
3       kemensos salurkan rp korban kerusuhan sosial p…
4                      mpr amandemen uud melebar manamana
                             …
8608    twice rilis teaser mv feel special jelang come…
8609    asap karhutla riau merambah nias bmkg imbau wa…
8610    tolak ruu pertanahan ribuan petani gelar aksi …
```

```
8611    niat momongan program hamil fedi nuril pengin aja
8612         txt comeback soobin akui gatal pamer spoiler
Name: title, Length: 8603, dtype: object
```

Stopwords biasanya dihilangkan pada saat pengklasifikasian teks, karena dianggap tidak membantu dalam pemodelan. Dengan jumlah yang banyak juga, stopwords lebih merugikan daripada insignifikan.

[ ]:
```
st_factory = StemmerFactory()
stemmer = st_factory.create_stemmer()

df['title'] = [stemmer.stem(kalimat) for i, kalimat in enumerate (df['title'])]
df['title']
```

[ ]:
```
0       radar pilwalkot medan menantu jokowi temu dpw …
1            malaysia sudut ri isu kabut asap invasi babi
2        viral driver ojol bekas pesan makan pakai sepeda
3             kemensos salur rp korban rusuh sosial papua
4                        mpr amandemen uud lebar manamana
                              …
8608    twice rilis teaser mv feel special jelang come…
8609    asap karhutla riau rambah nias bmkg imbau waspada
8610    tolak ruu tanah ribu tani gelar aksi istanadpr…
8611      niat momong program hamil fedi nuril pengin aja
8612          txt comeback soobin aku gatal pamer spoiler
Name: title, Length: 8603, dtype: object
```

Stemming adalah proses dimana suatu kata dikembalikan menjadi suku katanya. Proses ini membantu dalam pemodelan karena selain mengurangi jumlah kata yang diproses, juga membantu dalam proses-proses pemodelan lainnya seperti vectorizing

2.6.2 Text Processing Step 2: Count Vectorizing

[ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,␣
↪TfidfTransformer
countvectorizer = CountVectorizer(min_df=5, max_df=0.95)
countvec_title = countvectorizer.fit_transform(df['title'])
```

Vectorizing adalah salah satu cara yang efektif dalam pembuatan model klasifikasi teks, dengan mentransformasi informasi teks menjadi berbentuk kuantiti. Countvector melakukan pemrosesan teks persis seperti berikut, dengan melakukan kuantisasi frekuensi kata yang sudah di normalisasi.

[ ]:
```
countvec = dict(zip(countvectorizer.get_feature_names_out(), countvec_title.
↪toarray()[0]))
df_cv = pd.DataFrame.from_dict(countvec, columns=['frequency'], orient='index')
df_cv.sort_values(by=['frequency'], ascending = False)
```

[ ]:
```
           frequency
jokowi             1
nasdem             1
sumut              1
menantu            1
temu               1
…                  …
innova             0
inovasi            0
insentif           0
inspirasi          0
zona               0

[2727 rows x 1 columns]
```

2.6.3 Text Processing Step 3: TFID Transformation

[ ]:
```
tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True,␣
 ↪sublinear_tf=True)
X = tfidf_transformer.fit_transform(countvec_title).toarray()
y = df['label_score']

tfidf_df = pd.DataFrame(tfidf_transformer.idf_, index=countvectorizer.
 ↪get_feature_names(),columns=["idf_weights"])
tfidf_df.sort_values(by=['idf_weights'])
```

```
c:\Users\nicho\PycharmProjects\DataMining2\.dm2\lib\site-
packages\sklearn\utils\deprecation.py:87: FutureWarning: Function
get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will
be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

[ ]:
```
           idf_weights
kpk           3.989245
indonesia     4.119811
habibie       4.320190
polisi        4.499301
jokowi        4.534530
…                …
monica        8.268223
monsta        8.268223
move          8.268223
mutilasi      8.268223
zona          8.268223

[2727 rows x 1 columns]
```

Hampir terbalik dari countvectorizer, TFIDF memberikan bobot kepada setiap kata dimana kata

yang paling sering muncul di banyak dokumen memiliki bobot terkecil. Dengan vektorisasi ini, dapat ditentukan kata mana yang berbobot dan kata mana yang tidak signifikan

[ ]:
```
print("Size of X:", X.shape)
print("Size of y:", y.shape)
```

```
Size of X: (8603, 2727)
Size of y: (8603,)
```

2.7 Machine Learning

2.7.1 Machine Learning Step 1: Balancing

[ ]:
```
smote = SMOTE()
ros = RandomOverSampler()
X_bal, y_bal = ros.fit_resample(X, y)
```

Pada saat data cleaning, diketahui bahwa data tidaklah seimbang antara labelnya. Oleh karena itu, dilakukan random over sampling agar kedua variabel target memiliki jumlah yang sama

[ ]:
```
print("Size of X after balancing", X_bal.shape)
print("Size of y after balancing:", y_bal.shape)
```

```
Size of X after balancing (10578, 2727)
Size of y after balancing: (10578,)
```

2.7.2 Machine Learning Step 2: Splitting

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size=0.
↪33, random_state=42)
```

Splitting dilakukan selayaknya pemodelan machine learning lain, dengan dibaginya dataset dapat dibuat data evaluasi dan data training

[ ]:
```
print("Size of train X:", X_train.shape)
print("Size of train y:", y_train.shape)
print("Size of test X:", X_test.shape)
print("Size of test y:", y_test.shape)
```

```
Size of train X: (7087, 2727)
Size of train y: (7087,)
Size of test X: (3491, 2727)
Size of test y: (3491,)
```

2.7.3 Machine Learning step 3: Modelling & Hyperparameter Tuning

[ ]:
```
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBRFClassifier
```

[ ]:
```
random_forest_param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']}

multinomialnb_param_grid = {
    'alpha': (1, 0.1, 0.01, 0.001, 0.0001, 0.00001)}

sgdclassifier_param_grid = {
    'loss':["hinge", "log_loss", "log", "modified_huber", "squared_hinge",␣
 ↪"perceptron", "squared_error", "huber", "epsilon_insensitive",␣
 ↪"squared_epsilon_insensitive"],
    'penalty': ["l2", "l1", "elasticnet"]}


rf = GridSearchCV(cv=3, estimator=RandomForestClassifier(),␣
 ↪param_grid=random_forest_param_grid)
mnb = GridSearchCV(cv=3, estimator=MultinomialNB(),␣
 ↪param_grid=multinomialnb_param_grid)
sgd = GridSearchCV(cv=3, estimator=SGDClassifier(),␣
 ↪param_grid=sgdclassifier_param_grid)
abc =␣
 ↪AdaBoostClassifier(base_estimator=RandomForestClassifier(),n_estimators=25)
xgb = XGBRFClassifier(n_estimators=100, subsample=0.9, colsample_bynode=0.2)

# Found best parameter
rf_param = {'criterion': 'entropy', 'max_depth': 7, 'max_features': 'auto',␣
 ↪'n_estimators': 500}
mnb_param = {'alpha': 0.1}
sgd_param = {'loss': 'log', 'penalty': 'elasticnet'}
```

GridsearchCV adalah sebuah teknik hyperparameter tuning yang dapat memberikan parameter terbaik yang dapat digunakan sebuah classifier. Keuntungan dilakukan hyperparameter tuning menggunakan GridsearchCV adalah karena fungsi built-in Cross Validation yang sudah terdapat didalamnya, memastikan parameter yang terpilih adalah yang terbaik secara tervalidasi.

[ ]:
```
sgd = SGDClassifier(loss='log', penalty='elasticnet')
sgd.fit(X_train, y_train)

mnb = MultinomialNB(alpha=0.1)
mnb.fit(X_train, y_train)
```

```
rf = RandomForestClassifier(criterion="entropy", max_depth=7,␣
 ↪max_features='auto', n_estimators=500)
rf.fit(X_train, y_train)

xgb.fit(X_train, y_train)

abc.fit(X_train, y_train)
```

```
c:\Users\nicho\PycharmProjects\DataMining2\.dm2\lib\site-
packages\sklearn\linear_model\_stochastic_gradient.py:173: FutureWarning: The
loss 'log' was deprecated in v1.1 and will be removed in version 1.3. Use
`loss='log_loss'` which is equivalent.
  warnings.warn(
c:\Users\nicho\PycharmProjects\DataMining2\.dm2\lib\site-
packages\sklearn\ensemble\_forest.py:427: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

[ ]:
```
AdaBoostClassifier(base_estimator=RandomForestClassifier(), n_estimators=25)
```

[ ]:
```
y_pred_mnb = mnb.predict(X_test)
y_pred_rf = rf.predict(X_test)
y_pred_xgb = xgb.predict(X_test)
y_pred_abc = abc.predict(X_test)
y_pred_sgd = sgd.predict(X_test)
```

2.7.4 Machine Learning Step 4: Evaluation

[ ]:
```
def clasrep(test, pred):
    print(f'{" Classification Report ":=^40}')
    print(classification_report(test, pred, target_names=['Bukan Clickkbait',␣
 ↪'Clickbait']))

print('RF')
clasrep(y_test, y_pred_rf)
print('XGB')
clasrep(y_test, y_pred_xgb)
print('ABC')
clasrep(y_test, y_pred_abc)
print('MNB')
clasrep(y_test, y_pred_mnb)
print('SGD')
clasrep(y_test, y_pred_sgd)
```

```
RF
======== Classification Report =========
```

```
                  precision    recall  f1-score   support

Bukan Clickkbait       0.71      0.93      0.81      1778
       Clickbait       0.90      0.60      0.72      1713

        accuracy                           0.77      3491
       macro avg       0.80      0.77      0.76      3491
    weighted avg       0.80      0.77      0.76      3491

XGB
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.57      0.99      0.72      1778
       Clickbait       0.95      0.22      0.35      1713

        accuracy                           0.61      3491
       macro avg       0.76      0.60      0.54      3491
    weighted avg       0.75      0.61      0.54      3491

ABC
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.86      0.89      0.87      1778
       Clickbait       0.88      0.85      0.86      1713

        accuracy                           0.87      3491
       macro avg       0.87      0.87      0.87      3491
    weighted avg       0.87      0.87      0.87      3491

MNB
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.81      0.85      0.83      1778
       Clickbait       0.83      0.79      0.81      1713

        accuracy                           0.82      3491
       macro avg       0.82      0.82      0.82      3491
    weighted avg       0.82      0.82      0.82      3491

SGD
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.80      0.89      0.85      1778
       Clickbait       0.87      0.77      0.82      1713
```

```
          accuracy                           0.83      3491
         macro avg       0.84      0.83      0.83      3491
      weighted avg       0.84      0.83      0.83      3491
```

[ ]:
```
dct = DecisionTreeClassifier()
dct.fit(X_train, y_train)
```

[ ]:
```
DecisionTreeClassifier()
```

[ ]:
```
y_pred_dct = dct.predict(X_test)
```

[ ]:
```
print("DecisionTreeClassifier")
clasrep(y_test, y_pred_dct)
```

```
DecisionTreeClassifier
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.83      0.79      0.81      1778
       Clickbait       0.79      0.84      0.81      1713

        accuracy                           0.81      3491
       macro avg       0.81      0.81      0.81      3491
    weighted avg       0.81      0.81      0.81      3491
```

[ ]:
```
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
y_pred_lr = lr.predict(X_test)
```

[ ]:
```
print("LogisticRegression")
clasrep(y_test, y_pred_lr)
```

```
LogisticRegression
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.82      0.86      0.84      1778
       Clickbait       0.85      0.80      0.83      1713

        accuracy                           0.83      3491
       macro avg       0.84      0.83      0.83      3491
    weighted avg       0.84      0.83      0.83      3491
```

```
[ ]: abc_lr =␣
      ↪AdaBoostClassifier(base_estimator=LogisticRegression(),n_estimators=500)
```

```
[ ]: abc_lr.fit(X_train, y_train)
```

```
[ ]: AdaBoostClassifier(base_estimator=LogisticRegression(), n_estimators=500)
```

```
[ ]: y_pred_abc_lr = abc_lr.predict(X_test)

     print("LogisticRegression")
     clasrep(y_test, y_pred_abc_lr)
```

```
LogisticRegression
======== Classification Report =========
                  precision    recall  f1-score   support

Bukan Clickkbait       0.80      0.83      0.81      1778
       Clickbait       0.81      0.79      0.80      1713

        accuracy                           0.81      3491
       macro avg       0.81      0.81      0.81      3491
    weighted avg       0.81      0.81      0.81      3491
```

Dari hasil evaluasi semua classifier, yaitu: - Random Forest - XGBoost - AdaBoost (Dengan base estimator RandomForest) - Multinomial Naive Bayes - SGD (Stochastic gradient descent) Classifier - Decision Tree - Logistic Regression - AdaBoost (Dengan base estimator LogisticRegression)

didapatkan bahwa rata-rata akurasi dari semua model adalah 79.375%. Dengan tertinggi didapatkan oleh ADABoost menggunakan estimator Random Forest. Tentunya hanyak karena nilai akurasi tertinggi bukanlah sebuah alasan suatu model lebih baik dari yang lainnya, saran dari pemodelan di praktikum ini di masa depan adalah dengan mengeksplorasi list hyperparameter yang lebih mendalam, melakukan text processing dan vectorizing yang lebih mendalam, dan juga menelusuri cara balancing yang lebih mendalam

Saran lainnya adalah dengan menggunakan GPU untuk mengurangi beban komputasi kepada CPU, namun hambatan dari teknik ini adalah untuk saat ini penggunaan libraru cuML dari NVIDIA hanya terdapat pada Linux.