recruitML

June 6, 2024

0.1 : ML

[1]:
```
!pip install diffprivlib
```

```
Requirement already satisfied: diffprivlib in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (0.6.4)
Requirement already satisfied: numpy>=1.21.6 in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (from diffprivlib)
(1.26.4)
Requirement already satisfied: scikit-learn>=0.24.2 in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (from diffprivlib)
(1.4.2)
Requirement already satisfied: scipy>=1.7.3 in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (from diffprivlib)
(1.13.0)
Requirement already satisfied: joblib>=0.16.0 in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (from diffprivlib)
(1.4.2)
Requirement already satisfied: setuptools>=49.0.0 in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (from diffprivlib)
(69.5.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/home/jun/.pyenv/versions/3.11.8/lib/python3.11/site-packages (from scikit-
learn>=0.24.2->diffprivlib) (3.5.0)
```

[4]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB as SklearnGaussianNB
import numpy as np
import spacy
from typing import List, Tuple
import itertools
import matplotlib.pyplot as plt
from diffprivlib.models import GaussianNB as DPGaussianNB
import matplotlib.font_manager as fm

#
```

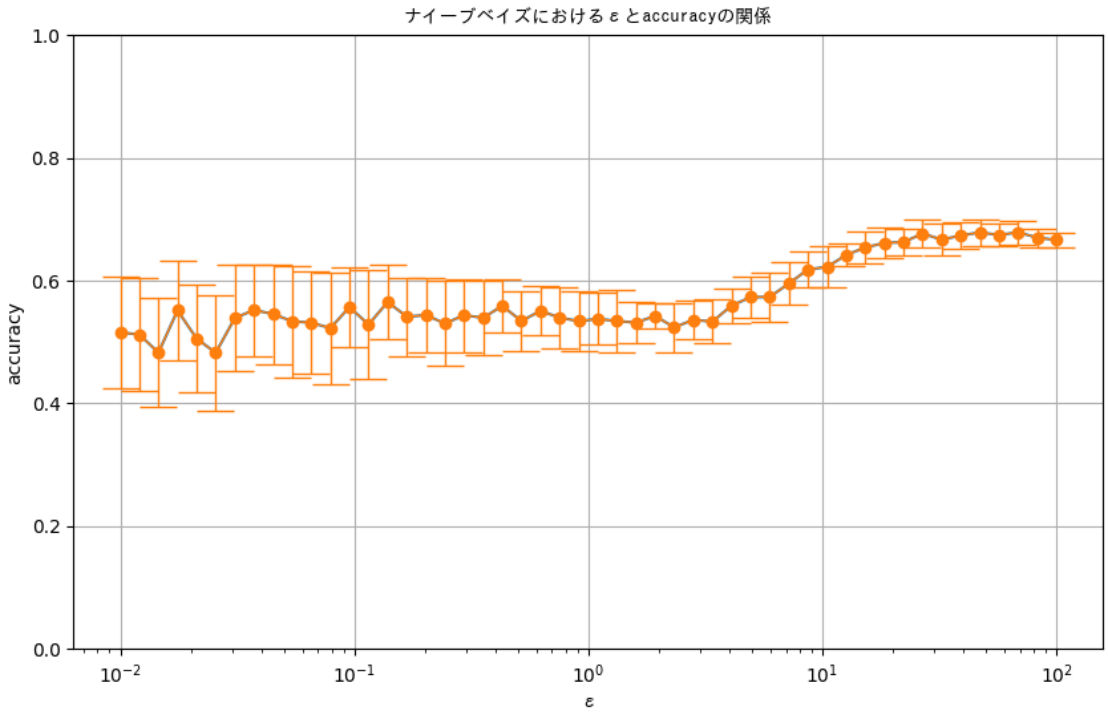
```
data_path = './data/reviews_with_sentiment.csv'
df = pd.read_csv(data_path)
font_path = '/usr/share/fonts/truetype/fonts-japanese-gothic.ttf'  #
font_prop = fm.FontProperties(fname=font_path)
# spaCy
nlp = spacy.load('ja_ginza')

#
POS = ['ADJ', 'ADV', 'INTJ', 'PROPN', 'NOUN', 'VERB']
MAX_TERMS_IN_DOC = 5
NGRAM = 1
MAX_DF = 1.0
MIN_DF = 0.01
NUM_VOCAB = 10000

def flatten(*lists) -> list:
    res = []
    for l in list(itertools.chain.from_iterable(lists)):
        for e in l:
            res.append(e)
    return res

def remove_duplicates(l: List[Tuple[str, float]]) -> List[Tuple[str, float]]:
    d = {}
    for e in l:
        d[e[0]] = e[1]
    return list(d.items())

#    BoW
tokens = []
for doc in df["review"]:
    parsed_doc = nlp(doc)
    similarities = [(token.similarity(parsed_doc), token.lemma_) for token in␣
 ↪parsed_doc if token.pos_ in POS]
    similarities = remove_duplicates(similarities)
    similarities = sorted(similarities, key=lambda sim: sim[1], reverse=True)[:
 ↪MAX_TERMS_IN_DOC]
    tokens.append([similarity[1] for similarity in similarities])

cv = CountVectorizer(ngram_range=(1, NGRAM), max_df=MAX_DF, min_df=MIN_DF,␣
 ↪max_features=NUM_VOCAB)
bow = cv.fit_transform([" ".join(ts) for ts in tokens]).toarray()

#
m = {
    "positive": 1,
    "neutral": 0,
```

```
    "negative": 0,
}
df["sentiment"] = df["sentiment"].map(m)
df["bow"] = bow.tolist()

X_train, X_test, y_train, y_test = train_test_split(df["bow"], df["sentiment"],␣
 ↪test_size=0.2)
X_train = [list(x) for x in X_train]
X_test = [list(x) for x in X_test]

#
clf = SklearnGaussianNB()
clf.fit(X_train, y_train)
print("Non-DP accuracy: ", clf.score(X_test, y_test))

#
epsilons = np.logspace(-2, 2, 50)
dim = np.array(X_train).shape[1]
lowers = np.zeros(dim)
uppers = np.ones(dim)
accuracies = {}

for epsilon in epsilons:
    accuracy = []
    for _ in range(20):
        dp_clf = DPGaussianNB(bounds=(lowers, uppers), epsilon=epsilon)
        dp_clf.fit(X_train, y_train)
        accuracy.append(dp_clf.score(X_test, y_test))
    accuracies[epsilon] = accuracy

#
x = epsilons
y = [np.mean(accuracies[eps]) for eps in epsilons]
e = [np.std(accuracies[eps]) for eps in epsilons]

plt.figure(figsize=(10, 6))
plt.semilogx(x, y)
plt.errorbar(x, y, yerr=e, marker='o', capthick=1, capsize=10, lw=1)
plt.xlabel(' ', fontproperties=font_prop)
plt.ylabel('accuracy')
plt.ylim(0, 1)
plt.title('       accuracy  ', fontproperties=font_prop)
plt.grid(True)
plt.show()
```

```
/tmp/ipykernel_362227/3754234034.py:46: UserWarning: [W008] Evaluating
Token.similarity based on empty vectors.
```

```
  similarities = [(token.similarity(parsed_doc), token.lemma_) for token in
parsed_doc if token.pos_ in POS]
```

```
Non-DP accuracy:  0.6552655265526552
```



[5]:
```
from sklearn.linear_model import LogisticRegression

#
clf = LogisticRegression(random_state=0).fit(X_train, y_train.to_numpy())
print("Non-DP accuracy: ", clf.score(X_test, y_test.to_numpy()))
```

```
Non-DP accuracy:  0.711971197119712
```

[6]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
from diffprivlib.models import LogisticRegression as DPLR

epsilons = np.logspace(-2, 2, 50)
dim = np.array(X_train).shape[1]
data_norm = math.sqrt(dim)
accuracies = {}

for epsilon in epsilons:
    accuracy = []
```

```
    for i in range(20):
        clf = DPLR(data_norm=data_norm, epsilon=epsilon).fit(X_train, y_train.
 ↪to_numpy())
        accuracy.append(clf.score(X_test, y_test.to_numpy()))
    accuracies[epsilon] = accuracy

#
x = epsilons
y = [np.mean(accuracies[eps]) for eps in epsilons]
e = [np.std(accuracies[eps]) for eps in epsilons]

plt.figure(figsize=(10, 6))
plt.semilogx(x, y)
plt.errorbar(x, y, yerr=e, marker='o', capthick=1, capsize=10, lw=1)
plt.xlabel(' ')
plt.ylabel('accuracy', fontproperties=font_prop)
plt.ylim(0, 1)
plt.title('        accuracy  ', fontproperties=font_prop)
plt.grid(True)
plt.show()
```