**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи №4 з дисципліни

«Обробка та аналіз текстових даних на Python»

„**Класифікація текстових даних**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Головня Олександр Ростиславович*

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2024

**Мета роботи:** Ознайомитись з класифікацією документів за допомогою

моделей машинного навчання.

**Завдання до лабораторної роботи:**

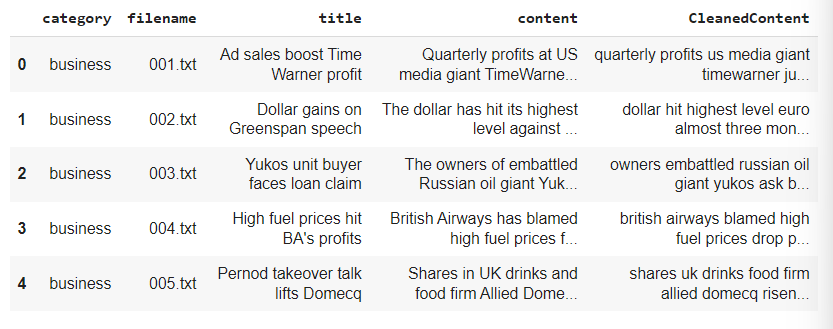
Створити програму, яка виконує завдання відповідно до варіанту.

В якості текстової моделі використати модель «Сумка слів». Виконати класифікацію за допомогою алгоритмів логістична регресія та градієнтний бустинг, порівняти їх точність. Спробувати покращити моделі за допомогою  GridSearchCV.

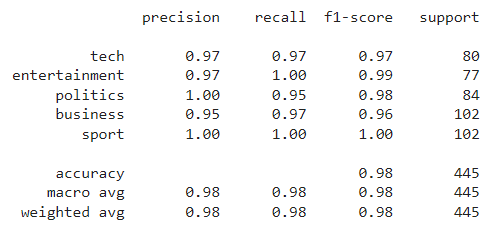
**Результат виконання коду:**

Виконана попередня обробка корпусу:

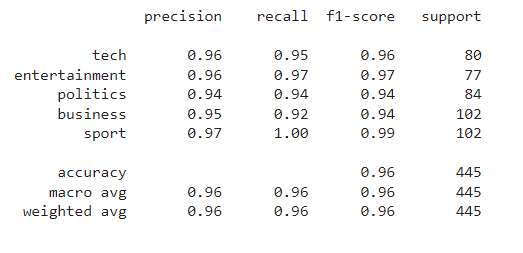




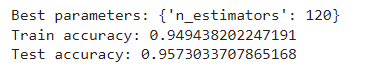
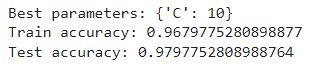
Класифікація за допомогою логістичної регресії:



Класифікація за допомогою градієнтого бустингу:



Покращення моделей за допомогою GridSearchCV:



**Код програми:**

import pandas as pd

content\_df = pd.read\_csv('bbc-news-data.csv', sep='\t')

content\_df

print(f'Duplicates count: {content\_df.duplicated().sum()}')

content\_df.isna().sum()

import re

from nltk.tokenize import WordPunctTokenizer

from nltk.corpus import stopwords

import nltk

nltk.download('stopwords')

stop\_words = stopwords.words('english')

tokenizer = WordPunctTokenizer()

def preprocess\_document(doc):

    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I | re.A)

    doc = doc.lower()

    doc = doc.strip()

    tokens = tokenizer.tokenize(doc)

    filtered\_tokens = [token for token in tokens if token not in stop\_words]

    doc = ' '.join(filtered\_tokens)

    return doc

content\_df['CleanedContent'] = content\_df['content'].apply(preprocess\_document)

content\_df.head()

from sklearn.model\_selection import train\_test\_split

train\_corpus, test\_corpus, train\_label\_names, test\_label\_names = train\_test\_split(content\_df['CleanedContent'],

                                                                                  content\_df['category'], test\_size=0.2,

                                                                                  stratify=content\_df['category'],

                                                                                  random\_state=1234)

from sklearn.feature\_extraction.text import CountVectorizer

bow\_vectorizer = CountVectorizer()

bow\_train\_features = bow\_vectorizer.fit\_transform(train\_corpus)

bow\_test\_features = bow\_vectorizer.transform(test\_corpus)

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

unique\_classes = list(set(test\_label\_names))

logistic\_regression = LogisticRegression()

logistic\_regression.fit(bow\_train\_features, train\_label\_names)

predicted\_labels = logistic\_regression.predict(bow\_test\_features)

print(classification\_report(test\_label\_names, predicted\_labels, labels=unique\_classes))

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification\_report

gbc = GradientBoostingClassifier()

gbc.fit(bow\_train\_features, train\_label\_names)

predicted\_labels = gbc.predict(bow\_test\_features)

print(classification\_report(test\_label\_names, predicted\_labels, labels=unique\_classes))

from sklearn.model\_selection import GridSearchCV

def tune\_model(model, param\_grid):

    model\_cv = GridSearchCV(model, param\_grid=param\_grid, n\_jobs=-1)

    model\_cv.fit(bow\_train\_features, train\_label\_names)

    print(f"Best parameters: {model\_cv.best\_params\_}")

    print(f"Train accuracy: {model\_cv.best\_score\_}")

    print(f"Test accuracy: {model\_cv.score(bow\_test\_features, test\_label\_names)}")

    return model\_cv.best\_estimator\_

logistic\_regression = LogisticRegression(max\_iter=1000)

param\_grid = {'C': [0.001, 0.01, 0.1, 0.5, 1, 5, 10]}

logistic\_regression\_cv = tune\_model(logistic\_regression, param\_grid)

import numpy as np

param\_grid = {'n\_estimators': np.arange(100, 140, 10)}

gbc = GradientBoostingClassifier()

gbc\_cv = tune\_model(gbc, param\_grid)

**Висновок:** Під час розробки лабораторної роботи, я ознайомився на практиці з класифікацією документів за допомогою моделей машинного навчання, а саме: в якості моделі використав модель «Сумка слів», виконавши класифікацію за допомогою алгоритмів логістична регресія та градієнтний бустинг, порівняв їх точність. Спробував покращити моделі за допомогою  GridSearchCV.