**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 №6 з дисципліни

«Обробка та аналіз текстових даних на Python»

„ Аналіз настроїв”

**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 вирішенням задач пошуку ключових слів та

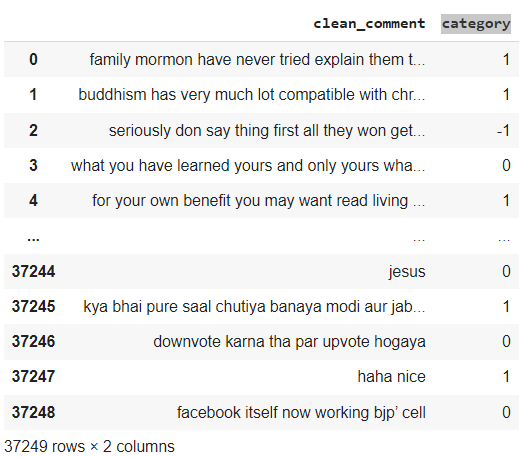
моделювання тем.

**Завдання до лабораторної роботи:**

Створити програму, яка виконує завдання відповідно до варіанту.

**Результат виконання коду:**

**Для початку імпортуємо модулі та зчитаємо файл.**



Визначив стоп-слова англійської мови.

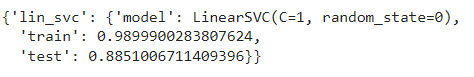


Я використав TF-IDF модель для представлення тексту у векторному вигляді.

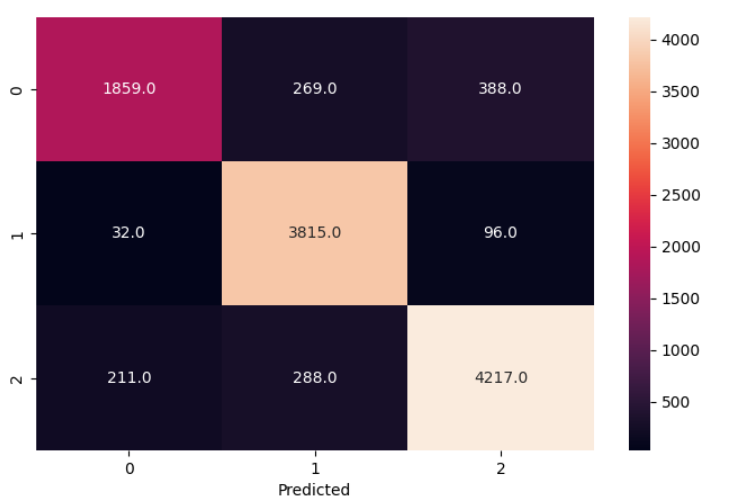
Побудував модель, розділивши дані на навчальні та тестові



Метод класифікації опорних векторів(Натренував та вивів результат):

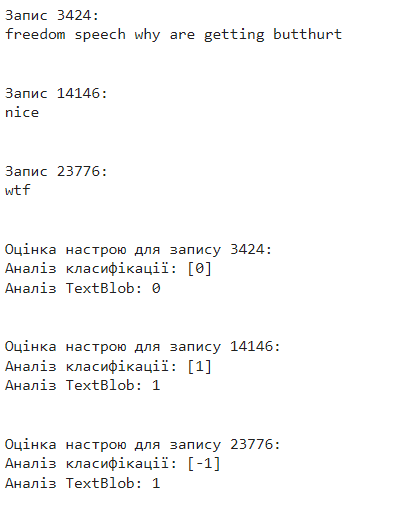


Матриця невідповідності:



Далі імпортував textblob

Розбив знову дані на тренувальні та тестові, зробив прогнози настроїв за допомогою textblob   
Та вивів оцінки настрою для трьох випадкових записів:

  
Завдання 2:

# Аналіз тексту за допомогою spaCy для розпізнавання іменованих сутностей

spacy\_doc = spacy\_nlp(text)

# Аналіз тексту за допомогою Stanza для частин мови та лемматизації

stanza\_doc = stanza\_nlp(text)



**Висновок:** Під час розробки лабораторної роботи, я ознайомився на практиці з вирішенням задачі аналізу настроїв та базовими можливостями бібліотеки spaCy.

**Код програми:**

import numpy as np

import pandas as pd

import nltk

import re

df = pd.read\_csv('Reddit.csv', sep=',')

df

from sklearn.preprocessing import LabelEncoder

category\_encoder = LabelEncoder()

df['category'] = category\_encoder.fit\_transform(df.category)

df

from sklearn.feature\_extraction.text import TfidfVectorizer

# Fill missing values in 'clean\_comment' column with an empty string

df['clean\_comment'] = df['clean\_comment'].fillna('')

tv = TfidfVectorizer(use\_idf=True, min\_df=0.0, max\_df=1.0)

df\_x = tv.fit\_transform(df.clean\_comment)

df\_y = df['category'].values

df\_x

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

def conf\_mat(model, x\_test, y\_test):

    y\_predicted = model.predict(x\_test)

    cm = confusion\_matrix(y\_test, y\_predicted)

    plt.figure(figsize=(8, 5))

    sns.heatmap(cm, annot=True, fmt=".1f")

    plt.xlabel('Predicted')

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

    df\_x, df\_y, test\_size=0.3, random\_state=0)

x\_train.shape, x\_test.shape

from sklearn.naive\_bayes import MultinomialNB

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import LinearSVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

models = {

    'lin\_svc': LinearSVC(penalty='l2', C=1, random\_state=0),

}

def get\_results(models, x\_train, y\_train, x\_test, y\_test) -> dict:

    results = []

    for name, model in models.items():

        model.fit(x\_train, y\_train)

        results.append((name, {

                'model': model,

                'train': model.score(x\_train, y\_train),

                'test': model.score(x\_test, y\_test),

            })

        )

    return dict(results)

results = get\_results(models, x\_train, y\_train, x\_test, y\_test)

results

conf\_mat(results['lin\_svc']['model'], x\_test, y\_test)

from textblob import TextBlob

# Вибрати три випадкові записи

random\_samples = df.sample(3)

# Вивести вміст цих записів

for idx, row in random\_samples.iterrows():

    print(f"Запис {idx}:")

    print(row['clean\_comment'])

    print("\n")

# Оцін настрій

for idx, row in random\_samples.iterrows():

    print(f"Оцінка настрою для запису {idx}:")

    prediction\_a = results['lin\_svc']['model'].predict(tv.transform([row['clean\_comment']]))

    print(f"Аналіз класифікації: {category\_encoder.inverse\_transform(prediction\_a)}")

    tb\_result = TextBlob(row['clean\_comment']).sentiment.polarity

    if -0.2 <= tb\_result < 0:

        prediction\_b = -1

    elif 0 <= tb\_result < 0.5:

        prediction\_b = 0

    else:

        prediction\_b = 1

    print(f"Аналіз TextBlob: {prediction\_b}")

    print("\n")

import spacy

import stanza

# Завантаження універсальної моделі spaCy для розпізнавання іменованих сутностей

spacy\_nlp = spacy.load('xx\_ent\_wiki\_sm')

# Завантаження української моделі Stanza для визначення частин мови та лемматизації

stanza.download('uk')

stanza\_nlp = stanza.Pipeline('uk')

# Читання файлу

file\_path = 'lab6-4.txt'

with open(file\_path, 'r', encoding='utf-8') as file:

    text = file.read()

# Аналіз тексту за допомогою spaCy для розпізнавання іменованих сутностей

spacy\_doc = spacy\_nlp(text)

# Аналіз тексту за допомогою Stanza для частин мови та лемматизації

stanza\_doc = stanza\_nlp(text)

# Завдання а): Знайти іменники жіночого роду

female\_nouns = [word.text for sentence in stanza\_doc.sentences for word in sentence.words if word.upos == 'NOUN' and word.feats and 'Gender=Fem' in word.feats]

print("Іменники жіночого роду:", female\_nouns)

# Завдання б): Вивести леми слів першого речення

first\_sentence = stanza\_doc.sentences[0]

first\_sentence\_lemmas = [word.lemma for word in first\_sentence.words]

print("Леми слів першого речення:", first\_sentence\_lemmas)

# Завдання в): Знайти та вивести організації

organizations = [ent.text for ent in spacy\_doc.ents if ent.label\_ == 'ORG']

print("Організації:", organizations)