УДК 629.33:681.51 **DOI: 10.30977/BUL.2219-5548.2023.101.1.14**

СТВОРЕННЯ РЕЛЕВАНТНОГО КОНТЕНТУ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Пронін С. В., Циганок О. П.
Харківський національний автомобільно-дорожній університет

Анотація: *У статті розглядаються інструменти для створення релевантних інформаційних наборів з великих масивів даних для її подальшого використання цільовою аудиторією, проаналізовано інструментарій для побудови систем аналізу даних.*
Ключові слова: *машинне навчання, аналіз даних, контент, pandas*

Вступ

Користувачі під час роботи в мережі інтернет генерують різноманітні дані. Через це виникають великі обсяги інформації. Вона може бути корисна як звичайним користувачам, так і великим компаніям. Однією з проблем використання інформації, що накопичується, є її неструктурованість Крім того, для різних груп користувачів необхідний не весь набір даних, а своя цільова вибірка. Для вирішення цього завдання на сьогодні розроблені різноманітні програмні інструменти.

У роботі аналізують проблеми створення релевантних інформаційних наборів з великих масивів даних для її подальшого використання цільовою аудиторією.

Аналіз публікацій

Технології машинного навчання для розв'язання задач з аналізу даних використовують вже майже півстоліття. За цей період було розроблено багато методів, які належать до машинного навчання та визначився принцип їх застосування, а основним недоліком були недостатні обчислювальні можливості комп'ютерів. Ситуація змінилося наприкінці XX-го століття завдяки збільшенню обчислювальних потужностей комп'ютерів, підвищенню можливостей операційних систем, розвитку мов програмування та розробленню різноманітних спеціалізованих фреймворків, що є бібліотеками функцій з машинного навчання та аналізу даних. Це дало можливість розробляти додатки для вирішення завдань з використанням машинного навчання [1].

Алгоритм машинного навчання складається з таких дій [1]:

- пошук необхідної інформації з предметної галузі;
- визначити інформаційний масив, який можна буде використовувати в додатку;
- розподілити цей масив на навчальну та тестову частини. За допомогою першої частині модель буде навчатися, а за допомогою другої ми зможемо перевірити результати навчання;
- вибрати алгоритм розв'язання задачі, який буде обробляти запити.

Систему можна налаштувати так, щоб вона працювала в режимі до навчання, додаючи нові дані та корегуючи результат своєї роботи.

Таким чином, застосовуючи технології машинного навчання, ми деякою мірою налаштовуємо комп'ютер, щоб він "вчився" відокремлювати з будь-яких даних корисні знання.

У процесі визначення дій зі створення систем машинного навчання актуальним є питання акумуляції, зберігання та організації даних. Водночас дані мають бути внесені до спеціальних структури, щоб була можливість зручної роботи з ними, зокрема зручного доступу до даних, визначення необхідного набору даних, їх структурування та фільтрації тощо.

Все це дасть змогу для подальшої роботи з даними за допомогою різноманітних моделей з метою отримання в процесі роботи потрібного результату.

На сьогодні для цього можна використовувати технології, які мають загальну назву «Великі дані» (Big data).

Технологіями Big data є такі:

- Apache Spark – це цілісна обчислювальна система з набором бібліотек для паралельного оброблення даних на кластерах комп'ютерів [2]. На сьогодні Spark є найбільш розроблюваним засобом з відкритим кодом для вирішення подібних завдань, що дозволяє йому бути корисним інструментом для будь-якого розробника або дослідника-

фахівця. Spark підтримує велику кількість мов програмування, що використовуються (Python, Java, Scala і R), а також бібліотеки для різноманітних завдань (SQL, стримінг, машинне навчання), а запустити його можна як з ноутбука, так і з кластера, що складається з тисячі серверів. Завдяки цьому Apache Spark є зручною системою для самостійної роботи з оброблення великих даних у великих масштабах.

MapReduce – це модель розподілених обчислень від компанії Google, яка використовується в технологіях Big Data для паралельних обчислень дуже великих (до декількох петабайт) наборів даних у комп'ютерних кластерах, і фреймворк для обчислення розподілених задач на вузлах кластера [3].

MapReduce є головною технологією Big Data, тому що вона насамперед орієнтована на паралельні обчислення в розподілених кластерах. Функція MapReduce полягає в поділі інформаційного масиву на частини, паралельному обробленні кожної частини на окремому вузлі і фінальному об'єднанні всіх результатів.

Програми використовують на розподілених вузлах кластера, водночас виконавча система сама здійснює процес реалізації.

Технологія є універсальною: вона може використовуватися для індексації вебконтенту, підрахунку слів у великому файлі, лічильників частоти звернень до заданої адреси, обчислення обсягів всіх вебсторінок з кожної URL-адреси конкретного хост-вузла, створення списку всіх адрес з необхідними даними й інших завдань оброблення величезних масивів розподіленої інформації. Також до галузі застосування MapReduce належить розподілений пошук і сортування даних, звернення графа вебпосилань, оброблення статистики логів мережі, побудова інвертованих індексів, кластеризація документів, машинне навчання та статистичне машинне перекладання. Також MapReduce адаптована під багатопроцесорні системи, добровільні обчислювальні, динамічні хмарні й мобільні середовища.

Але незважаючи на переваги розглянуті вище технології мають один недолік – їхня масштабність. Під час розроблення великих проектів це не має значення, а от для середніх та малих проектів їхні можливості можна визначити як надлишкові.

Тому в роботі розглянемо менш потужні технології, зокрема групу фреймворків, яка працює в середовищі Python [4]. До таких технологій належить Scikit-learn [5], яка побудована на основі стека SciPy (Scientific Python) і містить:

- NumPy – додає підтримку великих багатовимірних масивів і матриць, а також бібліотеку високорівневих математичних функцій для операцій з ними;

- Pandas – реалізує різні структури даних й аналіз

За допомогою Pandas ми можемо формувати інформаційні масиви та здійснювати операції поділу масивів, виділення необхідної інформації з масивів тощо.

Pandas [6] є пакетом Python з відкритим кодом, який надає ефективні й прості у використанні структури даних та інструменти аналізу для даних, що спрощує розв'язання задачі.

Pandas підтримує всі найпопулярніші формати зберігання даних: csv, excel, sql, буфер обміну, html тощо.

Це спрощує завдання, оскільки не потрібно зосереджуватись на формі зберігання даних. Для цього є спеціальна структура – датафрейм. Для доступу до даних достатньо написати одну команду:

company = pd.read_csv(os.path.join(PATH, "Company.csv"))

У цьому записі ми використовуємо метод pd.read, який дає нам можливість читати датафрейм "Company.csv". Префікс pd вказує, що ми використовуємо метод з пакета Pandas. Шлях до датафрейму ми отримуємо за допомогою os.path.join. Тут працює метод join з бібліотеки з роботи з операційною системою os, який з'єднує шляхи path з огляду на особливості операційної системи.

Pandas додає до Python нові структури даних – серії (series) і датафрейми (dataframe).

Об'єкт DataFrame найкраще використовувати як звичайну таблицю, адже DataFrame є табличною структурою даних. У будь-якій таблиці завжди наявні рядки й стовпці. Стовпцями в об'єкті DataFrame є об'єкти Series, рядки яких є їхніми безпосередніми елементами.

Мета та постановка завдання

Метою роботи є розроблення системи аналізу великих даних для отримання нової інформації з метою її аналізу

Для досягнення поставленої мети необхідно вирішити такі завдання:

- зібрати необхідну інформацію з предметної сфери;

- визначити інформаційний масив, який можна буде використовувати в додатку;
- здійснити дії над інформаційним масивом для вилучення нових даних.

Вибір набору даних

Для побудови моделі було вибрано набір даних з твітами про провідні компанії з 2015 по 2020 роки.

Цей набір даних є частиною статті, опублікованої на Міжнародній конференції великих даних IEEE 2020 року під час 6-ї спеціальної сесії з питань інтелектуального видобутку даних, що створена для визначення можливих спекулянтів та впливових факторів на фондовому ринку. Цей набір даних корисний для тих, хто цікавиться твітами, написаними про Amazon, Apple, Google, Microsoft та Tesla, використовуючи відповідні тикери для спільного використання.

Набір даних містить понад 3 мільйони унікальних твітів із інформацією про ідентифікатор твіту, його автора, дату публікації, текст твіту та кількість коментарів, лайків і ретвітів твітів, що відповідають компанії.

Створення релевантного контенту з масиву даних

Спершу визначаємо шлях до директорії з датасетами та задаємо налаштування pandas і seaborn (лістинг 1).

Лістинг 1 – Визначення шляху до директорії з дата сетами:

```
PATH = 'dataset/'

pd.set_option('display.max_columns', 30)
sn.set_context("paper", font_scale = 2)
```

Зчитаємо дані з csv-файлу з компаніями і перетворюємо в DataFrame функцією read_csv. Функцією set_index задаємо індекс стовпця (лістинг 2).

Лістинг 2 – Читання даних з csv-файлу:

```
company=pd.read_csv(os.path.join(PATH,
"Company.csv"))
company=company.set_index("ticker_symbol").to_dict()["company_name"]
company.
```

За допомогою програми отримаємо відповідь:

```
{'AAPL': 'apple',
 'GOOG': 'Google Inc',
 'GOOGL': 'Google Inc',
 'AMZN': 'Amazon.com',
 'TSLA': 'Tesla Inc',
 'MSFT': 'Microsoft'}
```

Далі читаємо та редагуємо датасет з індексами твітів та компаніями, до яких звертаються (лістинг 3).

Лістинг 3 – Редагування датасету з індексами твітів та компаніями:

```
company_tweet = pd.read_csv(os.path.join
(PATH, "Company_Tweet.csv"))
company_tweet.loc[company_tweet
['ticker_symbol'] == 'GOOGL',
'ticker_symbol'] = 'GOOG'
company_tweet.head().
```

Відповіддю буде редагований датасет з індексами твітів та компаніями, до яких звертаються (рис.1):

	tweet_id	ticker_symbol
0	550803612197457920	AAPL
1	550803610825928706	AAPL
2	550803225113157632	AAPL
3	550802957370159104	AAPL
4	550802855129382912	AAPL

Рис. 1. Редагований датасет

Визначаємо кількість записів, зокрема кількість унікальних, кількість авторів та відсоток дубльованих твітів (лістинг 4).

Лістинг 4 – Отримання кількості записів, авторів та визначення їхньої унікальності:

```
print("Всього записів: {} | Унікальних індексів твітів: {}"
      .format(len(company_tweet),
company_tweet['tweet_id'].nunique()))
Всього записів: 4336445 | Унікальних індексів твітів: 3717964
tweet = tweet.dropna()
print(f"Кількість авторів:
{tweet['writer'].nunique()}")
Кількість авторів: 140131
print("Відсоток дубльованих твітів:
{:.2f}
%".format(sum(tweet['body'].duplicated())/len(tweet) * 100))
Відсоток дубльованих твітів: 10.35 %
```

Для аналізу оцінки користувачів цих компаній за допомогою фреймворку NumPy створюємо модуль Negative з негативними словами, за якими будемо визначати негативність твітам (лістинг 5).

Лістинг 5 – Створння модуля Negative:

```
import numpy as np
negs = np.array(['if only ', 'assholes', 'negative', 'conspiracy', 'hate', 'fuck', 'flip', 'don\'t understand',
                 'doesn\'t understand', 'upset', 'plummet', 'misinformation', ' lies', ' rip ', 'bankruptcy',
                 'bizarre', 'damn', 'deception', 'fraud', ' idiot', ' crap ', 'stupid', 'losing focus',
                 'angry', ' ass ', 'bitch', ' shit', 'disaster', 'drop', 'dumb', 'despise',
                 'racist', 'sexist', 'illegal', 'bias'])

def wordpresent(s):
    return [negs[i] in s.lower() for i in range(len(negs))]
```

За допомогою модуля та класу Pool прискорюємо пошук твітів з негативними словами (лістинг 6).

Лістинг 6 – Пошук негативних висловлень за допомогою модуля Negative:

```
import Negative
%%time
with Pool(processes = 8) as pool:
    outcome = pool.map(Negative.wordpresent, dataframe.body.values)
```

Далі виводимо на екран список негативних слів і кількість твітів з розподілом за словами в датасеті (лістинг 7).

Лістинг 7 – Список негативних слів:

```
For w, cnt in
zip(Negative.negs,np.sum(outcome, axis=0)):
    print(w, " ", cnt)
```

Під час запуску методу отримаємо такі записи:

```
if only   956
assholes   162
negative   10677
conspiracy   1069
……….
dumb   4703
despise   131
racist   382
sexist   155
illegal   2290
bias   4798
```

У датафреймі створюємо стовпці, які будуть визначати негативний твіт чи нейтральний (лістинг 8).

Лістинг 8 – Метод для виділення з датасету негативних та нейтральних твітів:

```
dataframe["NegativeTweet"] = np.max(outcome, axis = 1)
dataframe["NeutralTweet"] = ~dataframe.NegativeTweet.values
dataframe.head().
```

У процесі запуску програми отримаємо такий результат (рис.2):

	tweet_id	writer	post_date
0	550441509175443456	VisualStockRSRC	1420070457
1	550441672312512512	KeralaGuy77	1420070496
2	550441732014223360	DozenStocks	1420070510

Рис. 2. Приклад виведення негативних та нейтральних твітів

Для виведення інформації створюємо метод DrawLinePlot, який буди видаляти з датасету негативні та нейтральні твіти (лістинг 9).

Лістинг 9 – Метод DrawLinePlot для виведення інформації у вигляди графіка:

```
def DrawLineplot(dataframe, companyName, title, drawNegative = False, drawNeutral = False):
    if companyName is None:
        company = dataframe
    else:
        company = dataframe.iloc[dataframe.ticker_symbol.values == companyName]
    plt.figure(figsize = (18, 8))
    if drawNeutral:
        companyNeutralTweetCount = company[["date","NeutralTweet"]].groupby(["date"]).sum()
    sn.lineplot(x = companyNeutralTweetCount.index,y = companyNeutralTweetCount.NeutralTweet.values,
                color='tab:blue', label = 'Нейтральні твіти')
    if drawNegative:
        companyNegativeTweetCount = company[["date","NegativeTweet"]].groupby(["date"]).sum()
```

```
sn.lineplot(x = companyNegativeTweetCount.index, y = companyNegativeTweetCount.NegativeTweet.values,
            color = 'tab:red', label = 'Негативні твіти')

    plt.xlabel("Дата", fontsize = 12)
    plt.ylabel("Кількість твітів", fontsize = 12)
    plt.title(title, fontsize = 16)
    plt.show()
```

За допомогою бібліотеки Seaborn побудуємо графік для кожної компанії. Як приклад наведено графік негативних та нейтральних твітів за тижнями для компанії Tesla (рис. 3).

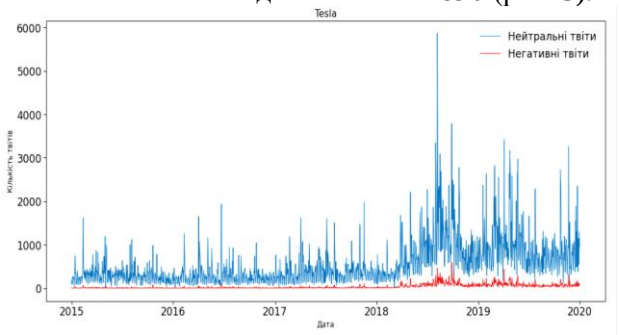

Рис. 3. Графік коментарів про компанію «Тесла»

На наступному етапі здійснюємо аналіз та класифікацію коментарів за допомогою методів машинного навчання. Побудуємо модель на основі логістичної регресії, на вхід якої буде подаватися коментар, а модель має оцінити його нейтральність або негативність.

Якщо модель повертає array([0]), то текст є нейтральним, якщо array([1]), то текст негативний.

Для цього створюємо два датафреми для навчання та тесту з розміром 150000 для навчання та 15000 для тесту (лістинг 10).

Лістинг 10 – Створення тестового набору даних та набору даних для навчання:

```
df = dataframe[["body","NegativeTweet"]]
df["NegativeTweet"] = df["NegativeTweet"].astype(int)
df.head()
test_df["NegativeTweet"].value_counts()
0   14515
1     485
Name: NegativeTweet, dtype: int64
train_df["NegativeTweet"].value_counts()
0   145258
1     4742
Name: NegativeTweet, dtype: int64.
```

Створюємо функцію для лексеми речень (лістинг 11).

Лістинг 11 – Функція для лексем: *snowball = SnowballStemmer(language = "english")*

```
stop_words = stopwords.words("english")
def tokenize_sentence(sentence: str, remove_stop_words: bool = True):
    tokens = word_tokenize(sentence, language="english")
    tokens = [i for i in tokens if i not in string.punctuation]
    if remove_stop_words:
      tokens = [i for i in tokens if i not in stop_words]
    tokens = [snowball.stem(i) for i in tokens]
    return tokens.
```

Для подальшої роботи потрібно здійснити так зване очищення датасету, тобто видалення різноманітних знаків пунктуації, стопслів тощо. Щоб програма могла відокремити одну частину інформації від іншої, ми визначаємо формати файлів, тобто домовленість про те, як усередині файлу записано текст. Найпростішим є формат, в якому кожна одиниця інформації знаходиться на окремому рядку. Такий файл майже не вимагає додаткового оброблення, достатньо визначити його як засоби мови програмування, що використовується, і поділити на рядки. Більшість мов дозволяють поділити файл на рядки однією–двома командами. Але більшість файлів, які необхідно обробити, мають більш складний формат.

Тому для текстових файлів, які за структурою більш складні, ніж список рядків, застосовується метод поділу на токени за допомогою регулярних виразів. Поняттям «токен» (token) визначають невелику частину тексту, що знаходиться в певному місці цього тексту і має певне значення (лістинг 12).

Лістинг 12 – Поділ на токени:

```
sentence_example = df.iloc[1]["body"]
tokens = word_tokenize(sentence_example, language = "english")
tokens_without_punctuation = [i for i in tokens if i not in string.punctuation]
stop_words = stopwords.words("english")
tokens_without_stop_words_and_punctuation = [i for i in tokens_without_punctuation if i not in stop_words]
```

```
snowball = SnowballStemmer(language = "english")
stemmed_tokens = [snowball.stem(i) for i in tokens_without_stop_words_and_punctuation]
print (f"Оригінальний текст: {sentence_example}")
print ("------------------")
print (f"Токени: {tokens}")
print ("------------------")
print (f"Токени без пунктуації: {tokens_without_punctuation}")
print ("------------------")
print (f"Токени без пунктуації і стоп-слів: {tokens_without_stop_words_and_punctuation}")
print ("------------------")
print (f"Токени після стемінгу: {stemmed_tokens}")
print ("------------------")
```

```
Оригінальний текст: Insanity of today weirdo massive selling. $aapl bid up 45 cents after hours after non stop selling in trading hours
Токени: ['Insanity', 'of', 'today', 'weirdo', 'massive', 'selling', '.', '$', 'aapl', 'bid', 'up', '45', 'cents', 'after', 'hours', 'after', 'non', 'stop', 'selling', 'in', 'trading', 'hours']
Токени без пунктуації: ['Insanity', 'of', 'today', 'weirdo', 'massive', 'selling', 'aapl', 'bid', 'up', '45', 'cents', 'after', 'hours', 'after', 'non', 'stop', 'selling', 'in', 'trading', 'hours']
------------------
Токени без пунктуації і стоп-слів: ['Insanity', 'today', 'weirdo', 'massive', 'selling', 'aapl', 'bid', '45', 'cents', 'hours', 'non', 'stop', 'selling', 'trading', 'hours']
------------------
Токени після стемінгу: ['insan', 'today', 'weirdo', 'massiv', 'sell', 'aapl', 'bid', '45', 'cent', 'hour', 'non', 'stop', 'sell', 'trade', 'hour'].
```

На останньому етапі за допомогою моделі логістичної регресії створюємо конвеєр даних методу pipeline, запускаємо навчання та здійснюємо перевірку (лістинг 13).

Лістинг 13 – Навчання моделі

```
Model = LogisticRegression(random_state = 0)
model.fit(features, train_df["NegativeTweet"])
LogisticRegression(random_state = 0)
# Якщо array([0]), то текст нейтральний, якщо array([1]), текст негативний
model.predict(features[203])
array([0])
# Створюємо конвеєр даних пайплайн
model_pipeline = Pipeline([
  ("vectorizer", TfidfVectorizer(tokenizer = lambda x: tokenize_sentence(x, remove_stop_words = True))),
  ("model", LogisticRegression(random_state = 0))
])
# Навчання
model_pipeline.fit(train_df["body"],train_df["NegativeTweet"])
Pipeline(steps = [('vectorizer',
          TfidfVectorizer(tokenizer = <function <lambda> at 0x0000020592919670>)),
         ('model', LogisticRegression(random_state = 0))])
# Приклад негативного тексту
model_pipeline.predict(["You are stupid monkey"])
array([1])
# Приклад нейтрального тексту
model_pipeline.predict(["Man, how are you?"])
array([0]).
```

Точність розпізнавання негативного коментаря становить 98 %, що задовольняє вимогам.

Висновок

Розглянуто підхід щодо створення релевантного контенту та аналізу великих масивів даних за допомогою методів машинного навчання. У роботі створена програмна модель, яка дозволяє оцінювати співвідношення користувачів до діяльності різних компаній. Розроблена модель аналізує висловлювання користувачів у соціальних мережах, вона може поділити їх на позитивні, негативні або нейтральні (у статті наведено приклад роботу з негативними та нейтральними висловлюваннями) та на основі цих висловлювань може визначати часові зведення співвідношень користувачів до компаній, діяльність яких аналізується. Модель основана на використанні словників з ключовими словами та спеціальних функцій роботи з датасетами з бібліотек NumPy, Pandas та Scikit-learn.

Конфлікт інтересів

Автори зазначають, що немає конфлікту інтересів щодо публікації цієї статті.

Література

1. Jeroen Janssens. Data Science at the Command Line: Facing the Future with Time-Tested Tools 1st Edition. O'Reilly Media. 2014. 212 pages pages.
2. Holden Karau Learning Spark. Lightning-Fast Big Data Analysis 1st Edition Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia. O'Reilly Media. 2020. 276 pages.
3. Tom White Hadoop. The Definitive Guide Third Edition. Yahoo Press. 2012. 688 pages.
4. Plas J. Vander Python for Complex Tasks: data Science and Machine Learning. O'Reilly Bestsellers, 2021. 576 pages.
5. Luis Pedro Coelho. Building Machine Learning Systems with Python. Packt Publishing. 2018. 406 pages.
6. Matt Harrison Effective Pandas: Patterns for Data Manipulation (Treading on Python). Independently published. 2021. 497 pages.

References

1. Jeroen Janssens. Data Science at the Command Line: Facing the Future with Time-Tested Tools 1st Edition. O'Reilly Media. 2014. 212 pages pages.
2. Holden Karau Learning Spark. Lightning-Fast Big Data Analysis 1st Edition Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia. O'Reilly Media. 2020. 276 pages.
3. Tom White Hadoop. The Definitive Guide Third Edition. Yahoo Press. 2012. 688 pages.
4. Plas J. Vander Python for Complex Tasks: data Science and Machine Learning. O'Reilly Bestsellers, 2021. 576 pages.
5. Luis Pedro Coelho. Building Machine Learning Systems with Python. Packt Publishing. 2018. 406 pages.
6. Matt Harrison Effective Pandas: Patterns for Data Manipulation (Treading on Python). Independently published. 2021. 497 pages.

Пронін Сергій Вікторович, к.т.н., доцент кафедри комп'ютерних технологій і мехатроніки, psv59777@gmail.com, тел. 057-707-37-43
Харківський національний автомобільно-дорожній університет, вул. Ярослава Мудрого, 25, 61002, м. Харків, Україна.

Циганок Олексій Петрович студент ХНАДУ
godsoft@protonmail.com, тел. 099 717 6271
Харківський національний автомобільно-дорожній університет, вул. Ярослава Мудрого, 25, 61002, м. Харків, Україна.

Creating relevant content using machine learning methods

Abstract *Today, users generate various data when working on the Internet. As a result, there are large amounts of different information. This information can be useful for both regular users and large companies. One of the problems with the use of accumulated information is its crude, unstructured nature. In addition, different user groups do not need the entire data set, but their own target sample. To solve this problem, various software tools have been developed today. This work is devoted to solving the problem of creating relevant information sets from large arrays of data for its further use by the target audience.* ***Goal.*** *The development of big data analysis systems is carried out to obtain new, previously unknown information. The* ***methodology*** *of applying algorithms of work with large data sets and methods of machine learning is used, namely the pandas library for operations on a data set and logistic regression for information classification As a* ***result****, a system was built that allows the analysis of lexical information, translate it into numerical format and create on this basis the necessary statistical samples. The* ***originality*** *of the work lies in the use of specialized libraries of data processing and machine learning to create data analysis systems. The* ***practical value*** *of the work lies in the possibility of creating data analysis systems built using specialized machine learning libraries.*

Keywords: *machine learning, data analysis, content, pandas, scikit-learn.*

Pronin Serhiy Viktorovych, Ph.D., Associate Professor of Computer Technology and Mechatronics, psv59777@gmail.com, tel. 057-707-37-43
Kharkiv National Automobile and Highway University, 61002, Ukraine, Kharkiv, 25 Yaroslav Mudry str.

Tsyhanok Oleksiy Petrovych, student of KhNAHU
godsoft@protonmail.com, tel. 099 717 6271
Kharkiv National Automobile and Highway University, 61002, Ukraine, Kharkiv, 25 Yaroslav Mudry str.