НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота № 3

З дисципліни: «Оброблення надвеликих масивів даних»

**Виконав:**

студент 5 курсу

групи ТР-02мп, ТЕФ

Круглий Д.В.

**Перевірила:**

проф. Федорова Н. В.

Київ – 2020

**Тема:** обробка текстової інформації та проектування ознак засобами Spark.

**Мета:** навчитися обробляти текстову інформацію та проектувати ознаки за допомогою засобів Spark.

**Теоретичні відомості**

***Машинне навчання***, включаючи навчання з учителем і без учителя – використання моделей, побудованих на базі статистичного аналізучи машинного навчання для отримання комплексних прогнозів на основі базових моделей.

***Статистичний аналіз*** — аналіз часових рядів, A/B-тестування (A/B testing, split testing — метод маркетингового дослідження; при його використанні контрольна група елементів порівнюється із набором тестових груп, у яких один чи кілька показників були змінені, щоб з’ясувати, які зі змін покращують цільовий показник.

***Поняття про кореляцію ознак***

Методи статистичного аналізу дають можливість вивчати окремі ознаки або властивості незалежно, наче ізольовано від інших. В багатьох агрономічних дослідженнях дуже важливо вияснити залежність між двома або декількома ознаками, встановити їх взаємний зв'язок. Але в таких дослідженнях рідко мають справу з точними і визначеними функціональними зв'язками, коли кожному значенню однієї величини відповідає строго визначене значення іншої величини. Частіше зустрічаються такі співвідношення між змінними, коли кожному значенню ознаки X відповідає не одна, а безліч можливих значень ознаки Y. Такі зв'язки появляються лише при масовому вивченні ознак і на відміну від функціональних називаються схоластичними (вірогідними) або кореляційними. В класифікаційному плані кореляції (взаємне співвідношення, залежність показників, явищ тощо) поділяються за напрямом, формою, силою та кількістю зв'язків.

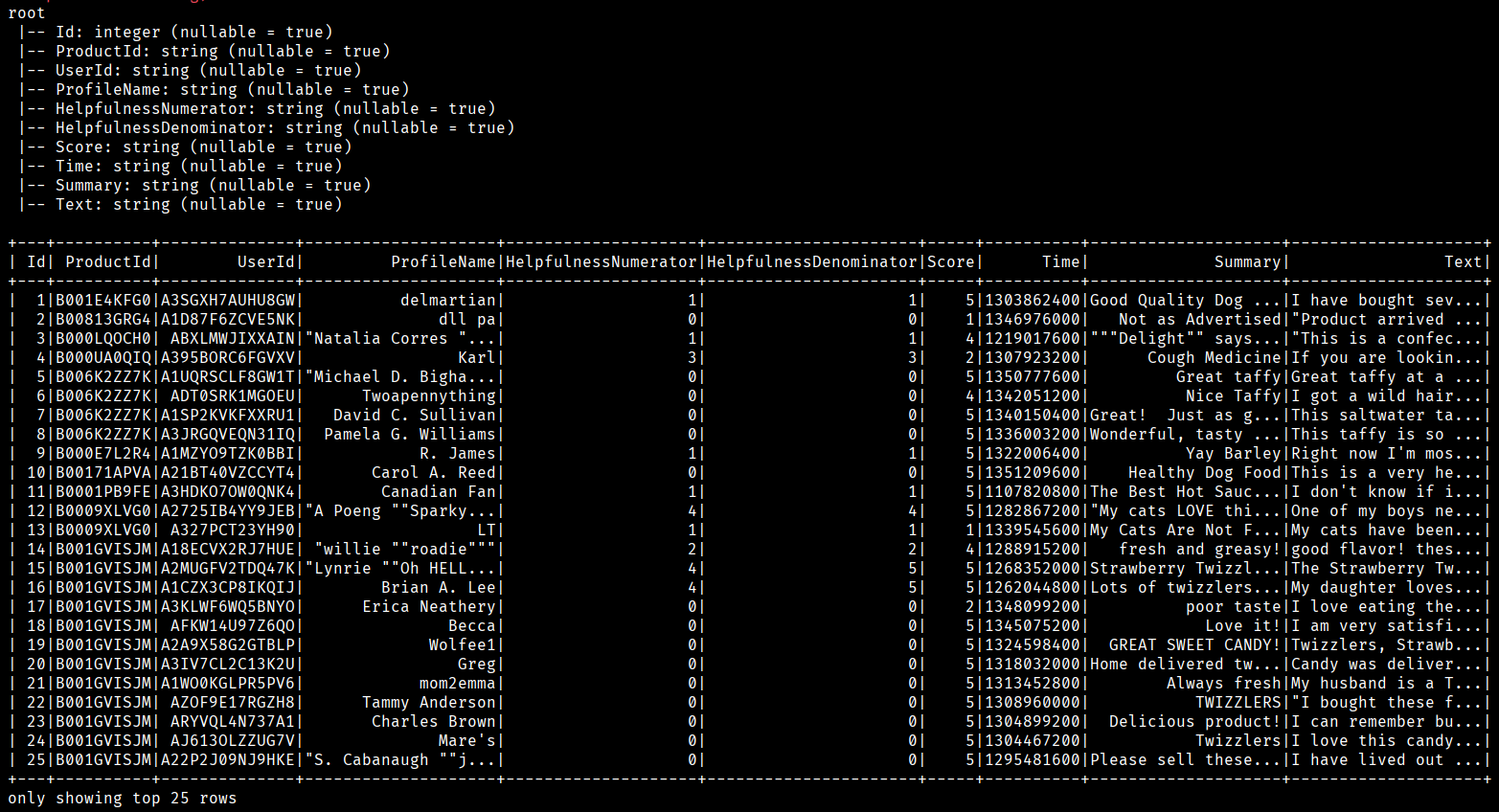
***Класифікація*** - це поширена задача машинного навчання, яка являє собою процес сортування вхідних даних за категоріями. Алгоритм класифікації визначає принцип призначення міток наданим вхідним даних. Наприклад, можна створити алгоритм машинного навчання, який приймає в якості вхідних даних інформацію про акції і ділить акції на дві категорії: акції, які слід продати, і акції, які варто залишити.

***Логістична регресія*** - один з алгоритмів, який можна використовувати для класифікації. API Spark для логістичної регресії підходить для задач двійковій класифікації або поділу вхідних даних на дві групи.

Дана лабораторна робота була виконана з використанням мови програмування Python.

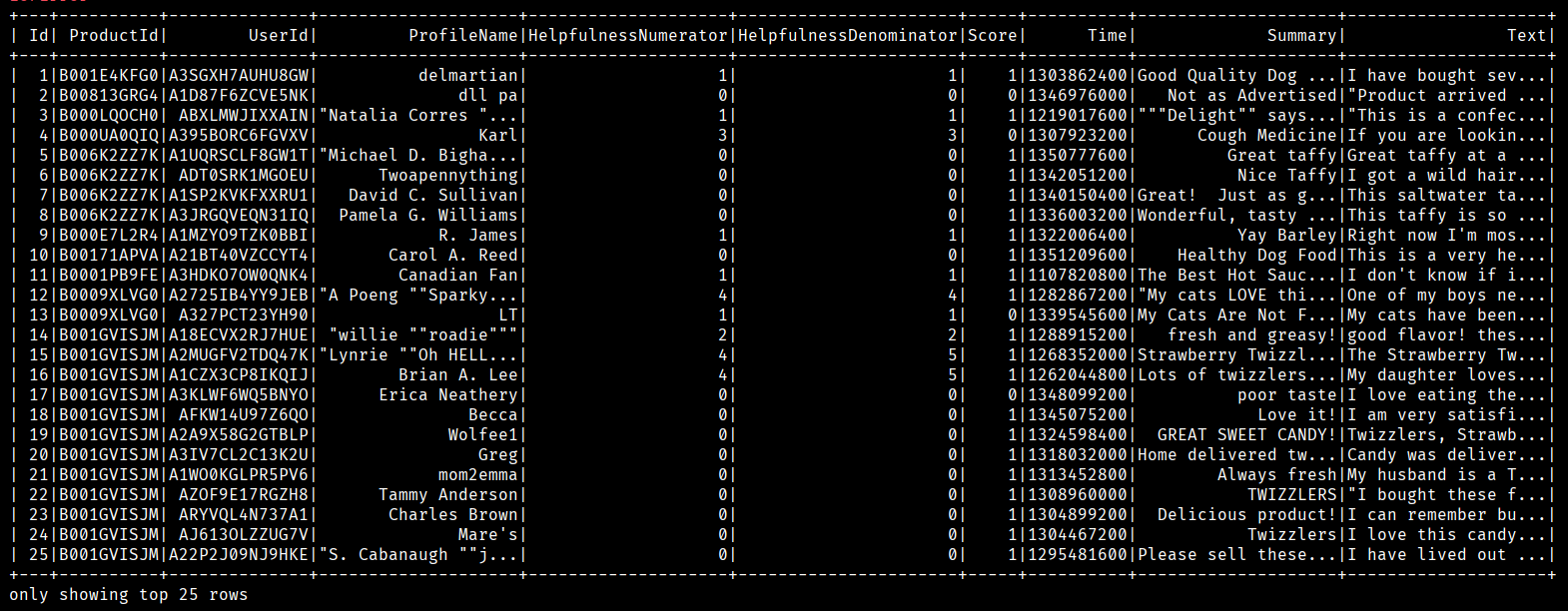
**Хід роботи**

1. Було обрано дані датасета з сайту https://www.kaggle.com/snap/amazon-fine-food-reviews.
2. Відображення схеми та фрагменту даних датасета:

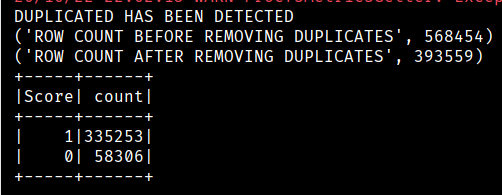


1. Перетворення задачі у бінарному вигляді:

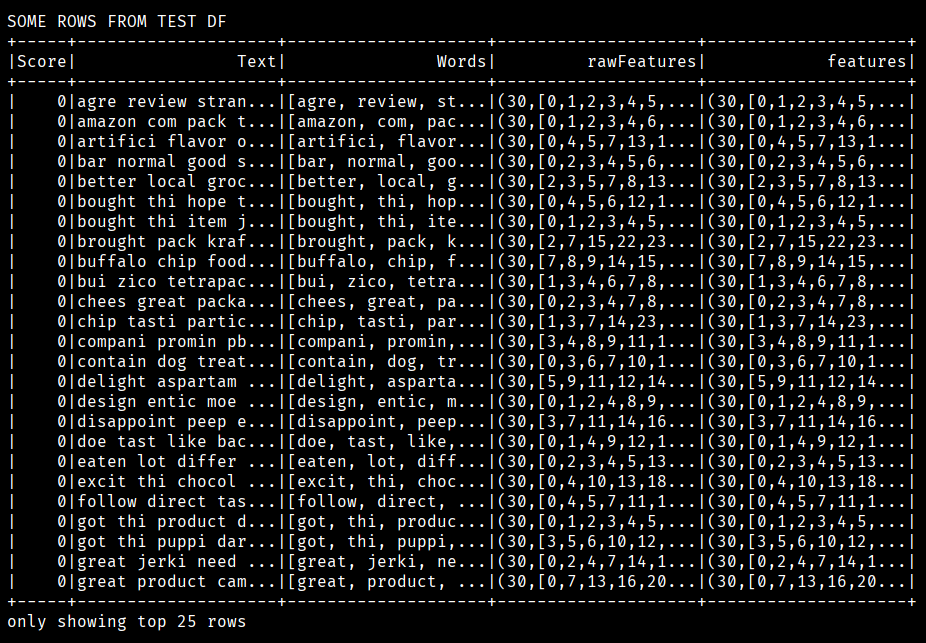
* 1,2 – негативний результат (0);
* 3,4,5 – позитивний результат (1).



1. Відображення загальної кількості даних та даних без дублікатів:



1. Відображення «зачищених» даних:



1. Виведення точності створеної моделі:

**C:\Users\Dima Kruhlyi\Desktop\віаіаві.png**

1. Лістинг програми:

**from** pyspark **import** SparkConf, SparkContext, SQLContext  
**from** pyspark.sql.types **import** StructType, StructField, DoubleType  
**from** pyspark.ml.evaluation **import** BinaryClassificationEvaluator  
**from** pyspark.ml.classification **import** LogisticRegression  
**from** pyspark.ml **import** Pipeline  
**from** pyspark.ml.tuning **import** CrossValidator, ParamGridBuilder  
  
**import** gensim.parsing.preprocessing **as** gsp  
**from** pyspark.sql.types **import** StringType  
**from** gensim **import** utils  
  
**from** pyspark.sql.types **import** IntegerType  
**from** pyspark.sql.functions **import** udf  
  
**from** pyspark.ml.feature **import** HashingTF, IDF, Tokenizer  
**from** pyspark.ml **import** Pipeline  
  
count\_printing\_rows = 25  
  
filters = [  
 gsp.strip\_numeric,  
 gsp.strip\_multiple\_whitespaces,  
 gsp.strip\_punctuation,  
 gsp.stem\_text,  
 gsp.strip\_tags,  
 gsp.strip\_short,  
 gsp.remove\_stopwords  
]  
  
  
**def** rating\_recognizer(rating):  
 rating\_condition = 2  
 **return** 1 **if** str(rating) > str(rating\_condition) **else** 0  
  
  
**def** text\_sanitizer(item):  
 text\_column\_index = 9  
 score\_column\_index = 6  
  
 text = item[text\_column\_index]  
  
 text = text.lower()  
 text = utils.to\_unicode(text)  
  
 **for** filterItem **in** filters:  
 text = filterItem(text)  
  
 **return** item[score\_column\_index], text  
  
  
  
spark\_context = SparkContext.getOrCreate(SparkConf().setMaster(**"local[\*]"**))  
spark = SQLContext(spark\_context)  
data\_df = spark.read.csv(**"D:\KPI\9-й семестр\Оброблення надвеликих масивів даних (Федорова Н.В.)\Laba\_3\data.csv"**, header=True, inferSchema=True)  
data\_df.printSchema()  
data\_df.show(count\_printing\_rows)  
  
common\_udf = udf(rating\_recognizer, IntegerType())  
data\_df = data\_df.withColumn(**'Score'**, common\_udf(**'Score'**))  
data\_df.show(count\_printing\_rows)  
  
**if** data\_df.count() > data\_df.dropDuplicates([**'UserId'**, **'ProfileName'**, **'Time'**, **'Text'**]).count():  
 **print**(**'DUPLICATED HAS BEEN DETECTED'**)  
  
**print**(**"ROW COUNT BEFORE REMOVING DUPLICATES"**, data\_df.count())  
  
data\_df = data\_df.dropDuplicates([**'UserId'**, **'ProfileName'**, **'Time'**, **'Text'**])  
  
**print**(**"ROW COUNT AFTER REMOVING DUPLICATES"**, data\_df.count())  
  
data\_df.groupBy(**'Score'**).count().show()  
  
san\_rdd = data\_df.rdd.map(**lambda** item: text\_sanitizer(item))  
  
san\_df = san\_rdd.toDF([**'Score'**, **'Text'**])  
san\_df.show(count\_printing\_rows)  
  
san\_df = san\_df.dropna()  
train\_df, test\_df = san\_df.randomSplit([.7, .3])  
  
tokenizer = Tokenizer(inputCol=**"Text"**, outputCol=**"Words"**)  
hashing\_TF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol=**"rawFeatures"**, numFeatures=30)  
idf = IDF(inputCol=hashing\_TF.getOutputCol(), outputCol=**"features"**, minDocFreq=5)  
pipeline = Pipeline(stages=[tokenizer, hashing\_TF, idf])  
  
model = pipeline.fit(train\_df)  
train\_df = model.transform(train\_df)  
test\_df = model.transform(test\_df)  
  
**print**(**"FRAGMENT OF ROWS FROM TEST DF"**)  
test\_df.show(count\_printing\_rows)  
  
  
**def** calculate\_accuracy(train\_data, test\_data):  
 logistic\_regression\_inst = LogisticRegression(labelCol=**"Score"**, featuresCol=**"features"**)  
  
 model = logistic\_regression\_inst.fit(train\_data)  
 predictions = model.transform(test\_data)  
  
 binary\_classification\_evaluator = BinaryClassificationEvaluator(labelCol=**"Score"**)  
 accuracy\_on\_predictions = binary\_classification\_evaluator.evaluate(predictions)  
  
 **print**(**"ACCURACY IS: %g"** % accuracy\_on\_predictions)  
  
  
calculate\_accuracy(train\_df, test\_df)

**Висновок:** під час виконання лабораторної роботи було реалізовано програму для обробки коментарів та було натреновано модель для передбачення майбутніх оцінок. Точність логістичної регресії набула значення у 58.7%, що не є досить точним значенням.