哈尔滨工业大学

<<大数据分析>>

开放式大作业

(2023 年度秋季学期**)**

姓名:	
学号:	
学院:	
教师:	

基于 Spark 的大数据犯罪信息分类分析系统

一、 背景分析

在公共安全领域，对犯罪行为的深入研究对社会治安的有效推行有着显而易见的重要意义，在对现有犯罪信息材料深入研究的基础上，可以提取犯罪的时空特征、类别特性，经过科学的研究，可采取有效措施，预防未来犯罪行为的再次发生。随着时间的推移，犯罪信息的数据量日益增加，在人类社会生活的多样化发展趋势下，犯罪的形式和手段也愈发五花八门，对犯罪数据进行系统性研究的难度日益加大，传统分析方法费时费力，效果欠佳，实用性逐渐降低。随着数据科学理论的日益完善，启用大数据赋能犯罪行为信息研究成为了该领域工作的有效手段和必经之路。

本次开放式大作业就是在此背景下进行设计、实验的，旨在设计并实现一个利用大数据分析工具，对犯罪信息进行分类分析的系统。该系统结合了大数据技术与机器学习算法，可从大量的犯罪信息中提取特征，并进行分类分析，可有效识别不同类型的犯罪行为，帮助执法部门和相关机构制定更有效的犯罪预防和打击的策略。

二、 相关工作

使用数据科学工具进行犯罪研究方向的工作开展已久，成果众多。近年来，大数据技术的完善和普及，又为这方面的工作注入了许多新力量、新手段。此处将以时间顺序，对一些经典的工作进行简要的介绍。

最早，D.E. Brown 在其 1988 年的工作[1]中，利用了数据融合和数据挖掘两种技术，构建了一个 ReCAP（区域犯罪分析程序）系统，可帮助犯罪分析人员进行职业罪犯的抓捕。其中，数据融合技术可对来自多个来源的信息进行组织、组合和解释，克服了由相互冲突的报告和杂乱或嘈杂的背景造成的混乱。数据挖掘技术可实现在大型数据库中自动发现模式和关系。

2013 年，Kaumalee Bogahawatte 和 Shalinda Adikari 开发了一个智能犯罪识

别系统（Intelligent Criminal Identification System，ICIS）[2]，该系统使用数据挖掘技术、聚类和分类，来有效地侦查犯罪。系统可以根据从犯罪现场收集的证据潜在地识别罪犯。该解决方案在 21 类严重犯罪中提供了三种犯罪类别，即抢劫、入室盗窃和盗窃。

2014 年，Gera, Priyanka 和 Vohra, Rajan 提出了使用聚类分析的城市犯罪分析，该工作使用聚类算法（主要为 K-均值聚类）的数据挖掘方法来分析犯罪模式，从三个维度对犯罪数据进行分析：犯罪的严重程度、各类犯罪对城市区域（住宅区、商业区等）的敏感性，及每一种犯罪类型在每一地区类别中的分布情况。

2018 年，A. Stec 和 D. Klabjan 进行了利用深度学习技术进行犯罪预测的研究[3]，该工作利用了深度神经网络在细粒度城市分区中进行第二天的犯罪计数预测，除犯罪数据集外，这项工作还考虑了与其他数据集的结合，如天气、人口普查数据等。犯罪计数被分成 10 个类别，模型预测每个空间区域每天最可能发生的犯罪类别。

2020 年，Manjunatha S.和 Annappa B.构建了一个基于多数据源的实时大数据分析框架[4]，可投入智慧城市应用，其中主要涉及了公共安全方面的解决方案。该系统通过对多个数据源实施数据混合技术，并基于机器学习和分类方法，可使用此框架实现一个犯罪实时警报系统。该工作还指出，可从实验结果中得知朴素贝叶斯分类在生成紧急警报方面表现更好。

2023 年，Ruaa Mohammed Saeed 和 Husam Ali Abdulmohsin 发表了一篇综述，对近年来通过机器学习和使用文本的数据挖掘预测犯罪率的相关工作进行了系统性的总结和研究[5]。该研究指出，监督学习方法在犯罪预测研究中的应用比其他方法多，而逻辑回归是预测犯罪最有效的方法。

三、 应用框架

本次大作业设计中，使用 Spark 框架来进行实验。

Spark 是一个开源的大数据处理框架，旨在提供高效、可扩展和易用的数据处理和分析能力，其提供了一个统一的计算引擎，支持各种数据处理任务，包括批处理、交互式查询、实时流处理和机器学习。它通过将数据加载到内存中进行操作，实现了比传统批处理框架更快的速度。Spark 的核心概念是弹性分布式数据集（Resilient Distributed Dataset，RDD），是一个可并行操作的分布式对象集

合，可以在集群中进行高效的数据处理。

Spark 提供了一个 Python API，即 PySpark，可以无缝地与 Spark 生态系统中的其他组件集成。本次实验采用 PySpark 完成数据集的分析和处理，并使用其他相关的 API 进行数据输出和可视化等操作。

四、应用设计

要实现对犯罪数据信息的分类分析，首先要考虑的一点是需要的数据属性有哪些。基于此前的研究工作，支撑犯罪分析的数据属性主要有如下几方面：犯罪的时间信息、位置信息（包含地理位置信息，如经纬度、所在城市等信息；与位置的人文描述信息，如城市、郊区，或更为详细的商业区、居民区等描述信息）、犯罪类别、严重程度、细节描述等等。

另外要考虑的一点是数据输出的需求分析，即本系统所要完成的分类分析具体是哪方面的，这一点要考虑本系统面向的相关研究人员的需求进行设计。考虑到治安人员在犯罪现场等情境下对犯罪信息进行文字记录时，会产生各种各样的描述数据，而当相关研究人员、机构需要对犯罪数据进行系统的审查、研究时，查阅这些记录信息会产生很大的工作量，也不利于借助计算机进行对犯罪数据库系统的分析和调研，因此，在本次作业报告中，主要完成的是基于犯罪的描述文字信息，对犯罪所属类别进行分类这项工作。

本次大作业的实验部分，选用了旧金山犯罪记录（San Francisco Crime，下载：https://www.kaggle.com/c/sf-crime/data），此数据集记录了大量的旧金山犯罪信息，每条记录包含时间、位置、描述、处罚等数据信息。实验中需要完成的任务就是根据这些记录的文字描述信息，将其分为 33 个犯罪类别。如，输入犯罪描述信息为“STOLEN AUTOMOBILE”，输出的分类结果应为 VEHICLE THEFT。实验中测试了多种分类分析算法，对各算法的输出结果进行性能评估，选出了最佳的系统。

五、实验

1.数据的读取和处理

首先，通过 Spark SQL 的 DataFrame API 读入 csv 数据创建 RDD，并进行初

步的处理：去掉不需要的列，保留 Category 类别列和 Descript 描述列。查看 drop 列后的数据前五行，代码如下，输出结果见图 1。

```
from pyspark.sql import SQLContext
from pyspark import SparkContext

sc = SparkContext()
sqlContext = SQLContext(sc)
data = sqlContext.read.format('com.databricks.spark.csv').options
(header='true',inferschema='true').load('sf-crime/train.csv')
drop_list = ['Dates', 'DayOfWeek', 'PdDistrict', 'Resolution',
'Address', 'X', 'Y']
data = data.select([column for column in data.columns
if column not in drop_list])
data.show(5)
```

```
+--------------+--------------------+
|      Category|            Descript|
+--------------+--------------------+
|      WARRANTS|      WARRANT ARREST|
|OTHER OFFENSES|TRAFFIC VIOLATION...|
|OTHER OFFENSES|TRAFFIC VIOLATION...|
| LARCENY/THEFT|GRAND THEFT FROM ...|
| LARCENY/THEFT|GRAND THEFT FROM ...|
+--------------+--------------------+
only showing top 5 rows
```

图 1 显示数据样本结果

用以下代码显示数据结构，输出如图 2。

```
data.printSchema()
```

```
root
 |-- Category: string (nullable = true)
 |-- Descript: string (nullable = true)
```

图 2 显示数据结构结果

接下来，对数据集两个属性的字符串进行处理。首先处理描述 Descript，流程如下：首先利用 PySpark.ml.feature 中的 RegexTokenizer 模块利用正则进行单词的切分，用 StopWordsRemover 模块移除停用词，再用 CountVectorizer 模块进行词频向量的构建，代码如下。

```
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover,
CountVectorizer
# regular expression tokenizer
```

```
    regexTokenizer = RegexTokenizer(inputCol="Descript", outputCol="words", pattern="\\W")
    # stop words
    add_stopwords = ["http","https","amp","rt","t","c","the"]
    stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").
    setStopWords(add_stopwords)
    # bag of words count
    countVectors = CountVectorizer(inputCol="filtered", outputCol="features",
    vocabSize=10000, minDF=5)
```

然后，处理 Category 列，方法为，调用 StringIndexer 将类别字符串编码为 0-32 的整数值，代码如下。

```
    from pyspark.ml.feature import StringIndexer
    label_stringIdx = StringIndexer(inputCol = "Category", outputCol = "label")
```

然后，定义一个流水线（Pipeline）对象。PySpark 的 Pipeline 是一个用于构建和组织机器学习工作流程的工具，提供了一种简洁而灵活的方式来定义、训练和部署机器学习模型。其能将数据处理和模型训练流程无缝地整合在一起，使得系统编程更加高效。以词频作为特征向量的流水线定义操作如下：

```
    from pyspark.ml import Pipeline
    pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors, label_stringIdx])
    # Fit the pipeline to training documents.
    pipelineFit = pipeline.fit(data)
    dataset = pipelineFit.transform(data)
    dataset.show(5)
```

经过以上的处理，数据集如图 3 所示，新增的列就是经过处理后获得的所需要的新属性。

```
+--------------+--------------------+--------------------+--------------------+--------------------+-----+
|      Category|           Descript|               words|            filtered|            features|label|
+--------------+--------------------+--------------------+--------------------+--------------------+-----+
|      WARRANTS|      WARRANT ARREST|   [warrant, arrest]|   [warrant, arrest]|(809,[17,32],[1.0...|  7.0|
|OTHER OFFENSES|TRAFFIC VIOLATION...|[traffic, violati...|[traffic, violati...|(809,[11,17,35],[...|  1.0|
|OTHER OFFENSES|TRAFFIC VIOLATION...|[traffic, violati...|[traffic, violati...|(809,[11,17,35],[...|  1.0|
| LARCENY/THEFT|GRAND THEFT FROM ...|[grand, theft, fr...|[grand, theft, fr...|(809,[0,2,3,4,6],...|  0.0|
| LARCENY/THEFT|GRAND THEFT FROM ...|[grand, theft, fr...|[grand, theft, fr...|(809,[0,2,3,4,6],...|  0.0|
+--------------+--------------------+--------------------+--------------------+--------------------+-----+
only showing top 5 rows
```

图 3 经处理后的数据样本显示

最后，进行训练集和测试集的划分，比例为 7:3，随机种子设为 42，代码如下，划分结果输出见图 4。

```python
# set seed for reproducibility
(trainingData, testData) = dataset.randomSplit([0.7, 0.3], seed=42)
print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

```
Training Dataset Count: 615354
Test Dataset Count: 262695
```

图 4 样本划分结果

接下来，对几种不同的文本分类算法进行实现、测试和评估。

2.以词频作为特征的逻辑回归分类

代码实现如下：

```python
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Descript","Category","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)
```

分类结果如图 5。

```
+------------------------------+-------------+------------------------------+-----+----------+
|                      Descript|     Category|                   probability|label|prediction|
+------------------------------+-------------+------------------------------+-----+----------+
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.873926929553423l,0.02040...|  0.0|       0.0|
+------------------------------+-------------+------------------------------+-----+----------+
only showing top 10 rows
```

图 5 词频特征逻辑回归分类结果

用以下代码进行准确率性能评估：

```python
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
print(evaluator.evaluate(predictions))
```

评估结果为，准确率达到了 0.9720592483792263。

3.以 TF-IDF 作为特征的逻辑回归分类

TF-IDF，即词频-逆向文档频率，思想如下：词汇的重要性随着它在文档中出现的次数成正比，但同时会随着它在文档集中出现的频率成反比，若词项在某一文本中出现的频率高，且在其他文本中出现频率低，则认为该词具有良好的文本特征表示能力，赋予其较高的权重。数据处理的代码如下：

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import HashingTF, IDF
hashingTF = HashingTF(inputCol="filtered", outputCol="rawFeatures", numFeatures=10000)
idf = IDF(inputCol="rawFeatures", outputCol="features", minDocFreq=5)
#minDocFreq: remove sparse terms
pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, hashingTF, idf, label_stringIdx])
pipelineFit = pipeline.fit(data)
dataset = pipelineFit.transform(data)
(trainingData, testData) = dataset.randomSplit([0.7, 0.3], seed = 100)
```

逻辑回归进行分类部分的代码：

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Descript", "Category", "probability", "label", "prediction") \
    .orderBy("probability", ascending=False) \
    .show(n=10, truncate=30)
```

上述代码的输出见图 6。准确率评估及结果输出代码与前一部分一致，最终结果为 0.9728018507875786，与以词频作为特征的逻辑回归分类结果几乎一致。

```
+----------------------------+------------+----------------------------+-----+----------+
|                    Descript|    Category|                 probability|label|prediction|
+----------------------------+------------+----------------------------+-----+----------+
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.8832859441658517,0.01803...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.8832859441658517,0.01803...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.8832859441658517,0.01803...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.8832859441658517,0.01803...|  0.0|       0.0|
|THEFT, BICYCLE, <$50, NO SE...|LARCENY/THEFT|[0.8832859441658517,0.01803...|  0.0|       0.0|
|THEFT, GRAND, BY FIDUCIARY,...|LARCENY/THEFT|[0.8683010634577338,0.01417...|  0.0|       0.0|
|THEFT, GRAND, BY FIDUCIARY,...|LARCENY/THEFT|[0.8683010634577338,0.01417...|  0.0|       0.0|
|THEFT, GRAND, BY FIDUCIARY,...|LARCENY/THEFT|[0.8683010634577338,0.01417...|  0.0|       0.0|
|THEFT, GRAND, BY FIDUCIARY,...|LARCENY/THEFT|[0.8683010634577338,0.01417...|  0.0|       0.0|
|THEFT, GRAND, BY FIDUCIARY,...|LARCENY/THEFT|[0.8683010634577338,0.01417...|  0.0|       0.0|
+----------------------------+------------+----------------------------+-----+----------+
only showing top 10 rows

0.9728018507875786
```

图 6 TF-IDF 特征逻辑回归分类结果

4.朴素贝叶斯

朴素贝叶斯分类算法是一种基于贝叶斯定理和特征条件独立性假设的简单而有效的分类算法。其通过计算给定特征条件下每个类别的后验概率，选择具有最高概率的类别作为分类结果。朴素贝叶斯假设特征之间相互独立，简化概率计算。在之前的结果中，TF-IDF 作为特征的评估结果略微好于词频作为特征的评估结果，因此此次实验接下来继续以 TF-IDF 作为特征进行分类，数据处理代码与第 3 部分一致。朴素贝叶斯分类的代码实现如下，输出见图 7。

```python
from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=1)
model = nb.fit(trainingData)
predictions = model.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Descript","Category","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)
```

结果表明，朴素贝叶斯分类的评估准确率指标评估结果超越了逻辑回归分类，高达 0.9952050829693171。

```
+------------------+------------+----------------------------+-----+----------+
|          Descript|    Category|                 probability|label|prediction|
+------------------+------------+----------------------------+-----+----------+
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
|GRAND THEFT BICYCLE|LARCENY/THEFT|[1.0,1.415084567581933E-29...|  0.0|       0.0|
+------------------+------------+----------------------------+-----+----------+
only showing top 10 rows

23/12/11 20:18:01 WARN DAGScheduler: Broadcasting large task binary with size 3.2 MiB
0.9952050829693171
```

图 7 朴素贝叶斯分类结果

5.随机森林

随机森林分类算法是一种集成学习方法，通过构建多个决策树并基于投票或平均预测结果进行分类。每个决策树是通过从原始数据中随机选择样本和特征进行训练而生成的。随机森林算法通过引入随机性来降低单个决策树过拟合的风险，并通过集成多个决策树的结果来提高分类准确性。它在处理大规模数据集和高维特征的情况下表现良好，并具有较高的鲁棒性和泛化能力。代码实现如下：

```
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(labelCol="label",
                            featuresCol="features",
                            numTrees=100,
                            maxDepth=4,
                            maxBins=32)
# Train model with Training Data
rfModel = rf.fit(trainingData)
predictions = rfModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Descript", "Category", "probability", "label", "prediction") \
    .orderBy("probability", ascending=False) \
    .show(n=10, truncate=30)
```

代码输出的结果及准确率评估结果输出见图 8。可见，对犯罪描述高维稀疏数据来说，随机森林分类算法效果欠佳，评估结果只有 0.33728945500886065。

```
|                    Descript|    Category|                 probability|label|prediction|
+----------------------------+------------+----------------------------+-----+----------+
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
|GRAND THEFT FROM UNLOCKED AUTO|LARCENY/THEFT|[0.4040284748996252,0.10261...|  0.0|       0.0|
+----------------------------+------------+----------------------------+-----+----------+
only showing top 10 rows

                                                                  0.33728945500886065
```

图 8 随机森林分类结果

6.结论

从以上实验可看出，使用 TF-IDF 作为特征的朴素贝叶斯分类算法的准确度评估结果是最佳的，精度可达 0.99。利用此系统根据犯罪数据中的描述文本，对犯罪数据进行分类，效率和精度都很高，可以很大程度上方便公共安全领域的工作人员和机构。

六、心得体会

由于之前从未使用过 Hadoop 和 Spark，在实验初期遇到了很多困难，通过查阅网上的资料，慢慢掌握了 Spark 的使用，后面的实验就逐渐明朗了。在实验实施过程中，随机森林模型训练的过程中持续遇到内存不足的提示，由于此前对 Spark 的分布式计算结构缺乏认知，尝试了很多无效的解决方法，也是通过查阅网上资料才进行了正确的配置，得以获得正确的结果输出。

参考文献

[1] D. E B. The Regional Crime Analysis Program (ReCAP): a framework for mining data to catch criminals: SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)[C], 1998.

[2] K. B, S. A. Intelligent criminal identification system: 2013 8th International Conference on Computer Science & Education[C], 2013.

[3] STEC A, KLABJAN D. Forecasting crime with deep learning[J]. arXiv preprint arXiv:1806.01486, 2018.

[4] MANJUNATHA S, ANNAPPA B. Real-time big data analytics framework with data blending approach for multiple data sources in smart city applications[J]. Scalable Computing: Practice and Experience, 2020,21(4): 611-623.

[5] SAEED R M, ABDULMOHSIN H A. A study on predicting crime rates through machine learning and data mining using text[J]. Journal of Intelligent Systems, 2023,32(1; 20220223).