Ukraine War Disorder Type and Machine Learning

Justin Abner

Data Science Program, Bellevue University

DSC650-T301 Big Data

Professor Nasheb Ismaily

March 3, 2024

Ukraine War Disorder Type and Machine Learning

Problem Statement

As it has become known to many, there is currently a war happening in Ukraine with Russia. It is saddening to learn how the soldiers are impacted but let's not forget about the civilians that choose to stay at home or had no place to flee during the war and have succumb to the environment. The intention of this project is to utilize past and current (withing a month of present day) data collected from Ukraine and see if we can predict the outcome type depending on the location and the involved parties. Doing so will allow the appropriate notices to be sent to necessary parties (i.e. message to civilians to flee or that the parties are harmless or to the military to be on high alert).

Data and Applied Methods

The data selected for was obtained from ACLED and contains real world data from Europe and Central Asia. Areas of interest within the data are fields involving Ukraine and Russia. The data ranges from civilian protests up to armed military battles. Transformations that were performed were done in NiFi and PySpark. NiFi transformations were selecting columns of interest (YEAR, DISORDER_TYPE, ACTOR1, ACTOR2, COUNTRY, and ADMIN2) and filtering the data by the year (2022 or later) and country (actions occurring in Ukraine). Within PySpark null fields in the ACTOR1 and ACTOR2 columns were replaced with the word "None". YEAR and COUNTRY were dropped as they were not necessary for the building of the model.

PySpark Decision Tree Classifier

By further utilizing PySpark a Decision Tree Classifier (DTC) was created. Being that all of the data columns were "strings"; the data was fit and transformed using a StringIndexer. From there, ACTOR1, ACTOR2 and ADMIN2 (which have been renamed to

'Indexed_Party_A','Indexed_Party_B','Indexed_Region', respectively) were transformed using a VectorAssembler which combined them all into one column named features. The target and feature columns were selected into one data frame and then split 70/30 (train/test). The DTC was tuned in a manner to increase the max bin size to accommodate the high variability within the columns (131 different categories within a column). Predictions were made and an accuracy of 98% was calculated.

Chosen Components

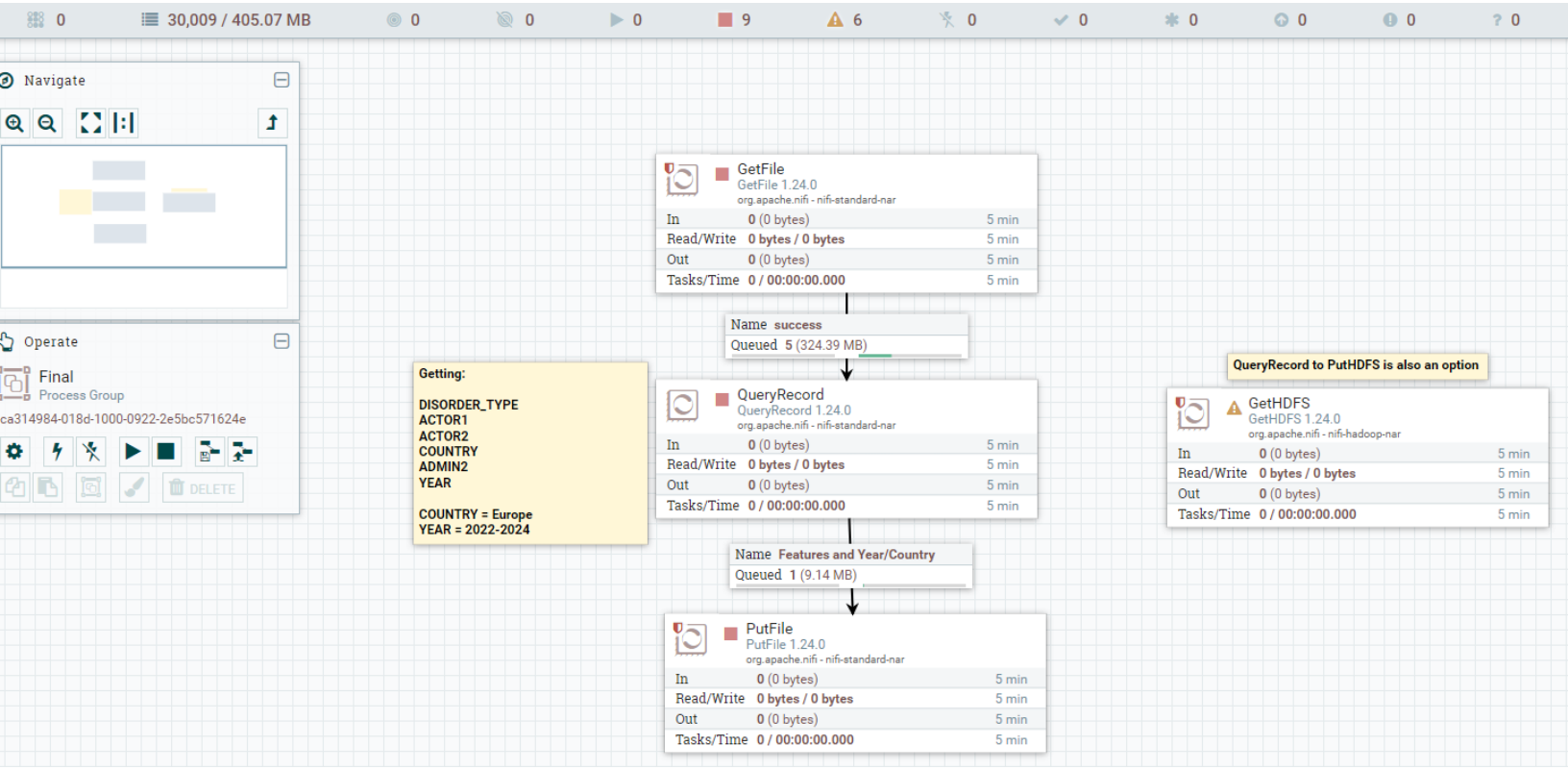
The components of NiFi and PySpark were chosen based on NiFi's ability to ingest large volumes of data, process it, and deliver it quickly. PySpark was selected based on its ability to utilize HDFS which will ensure data security and scalability, as well as PySpark's ability to utilize machine learning.

Reflection

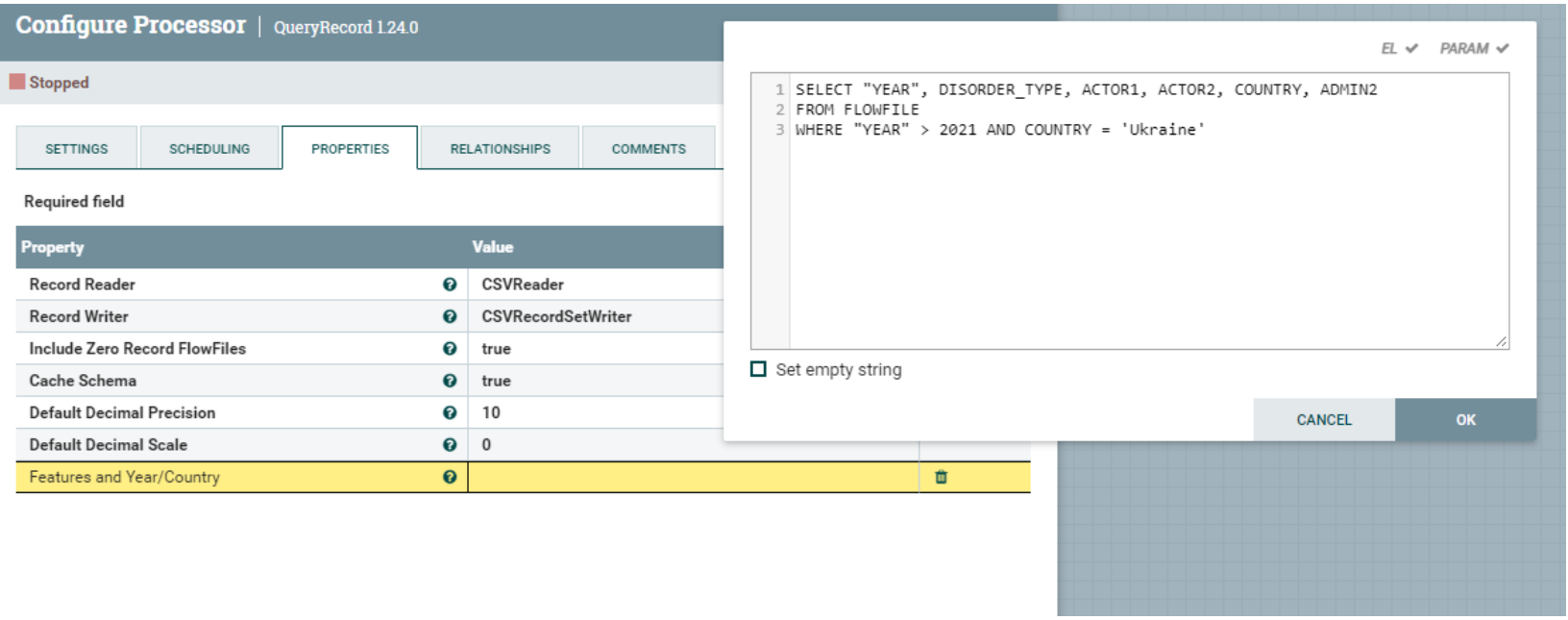
Implications are that the predicted outcomes can have dire consequences if followed closely and not viewed as suggestions. While the accuracy is impressive, I am under the impression that data is not distributed equally (much more battle data than civilian related data). I am very proud of my utilization of NiFi and how comfortable I am becoming with it. PySpark was a nice experience; however, I am excited about using it in a different environment (i.e. Jupyter Notebook).

NiFi

Complete Flow

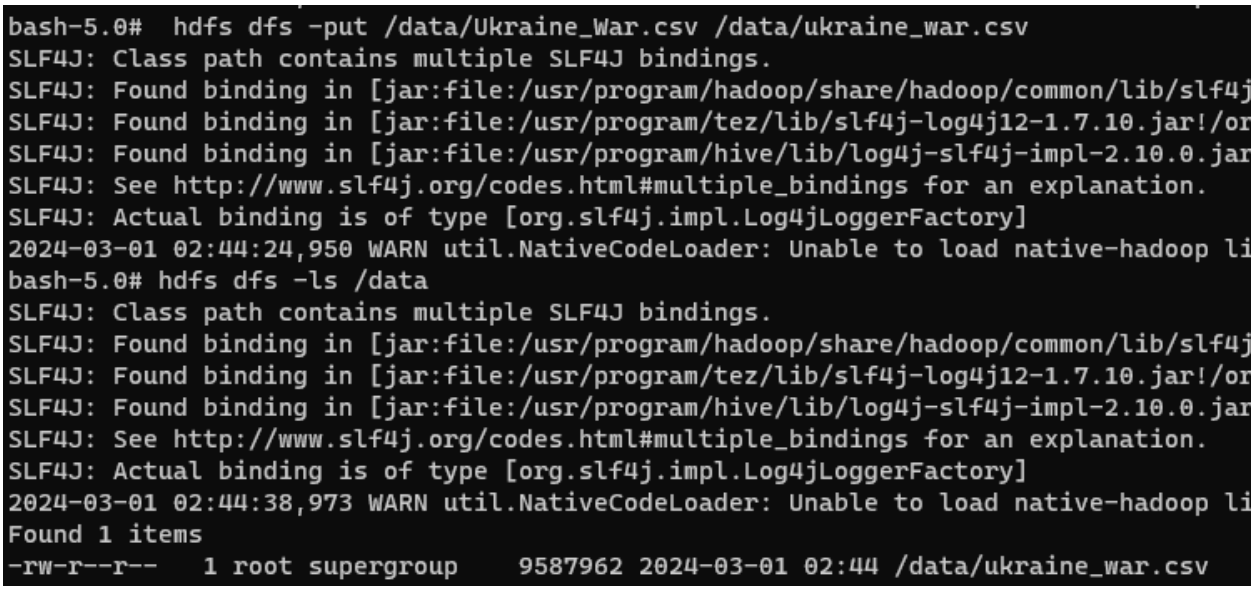


QueryRecord



HDFS

Moving file from local storage to HDFS



PySpark

Loading of Data, Filling Null Fields, and Selecting Columns

```
>>> ukr_df = spark.read.format('csv').option('header', 'true').load('/ukraine_war')
>>> ukr_df.show(5)
+----+-----------------+--------------------+--------------------+-------+----------------+
|YEAR|     DISORDER_TYPE|              ACTOR1|              ACTOR2|COUNTRY|           ADMIN2|
+----+-----------------+--------------------+--------------------+-------+----------------+
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|      Bakhmutskyi|
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|       Pokrovskyi|
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|       Pokrovskyi|
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|Sievierodonetskyi|
|2024|Political violence|Military Forces o...|                null|Ukraine|          Sumskyi|
+----+-----------------+--------------------+--------------------+-------+----------------+
only showing top 5 rows

>>> no_null_df = ukr_df.fillna(value='None')
>>> no_null_df.show(5)
+----+-----------------+--------------------+--------------------+-------+----------------+
|YEAR|     DISORDER_TYPE|              ACTOR1|              ACTOR2|COUNTRY|           ADMIN2|
+----+-----------------+--------------------+--------------------+-------+----------------+
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|      Bakhmutskyi|
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|       Pokrovskyi|
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|       Pokrovskyi|
|2024|Political violence|Military Forces o...|Military Forces o...|Ukraine|Sievierodonetskyi|
|2024|Political violence|Military Forces o...|                None|Ukraine|          Sumskyi|
+----+-----------------+--------------------+--------------------+-------+----------------+
only showing top 5 rows

>>> columns_df = no_null_df.select('DISORDER_TYPE', 'ACTOR1', 'ACTOR2', 'ADMIN2')
>>> columns_df.show(5)
+-----------------+--------------------+--------------------+----------------+
|     DISORDER_TYPE|              ACTOR1|              ACTOR2|           ADMIN2|
+-----------------+--------------------+--------------------+----------------+
|Political violence|Military Forces o...|Military Forces o...|      Bakhmutskyi|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|
|Political violence|Military Forces o...|Military Forces o...|Sievierodonetskyi|
|Political violence|Military Forces o...|                None|          Sumskyi|
+-----------------+--------------------+--------------------+----------------+
```

PySpark Cont.

Updating Column Names

```
>>> updated_col_names = ['Event_Type','Involved_Party_A','Involved_Party_B','Region']
>>> renamed_df = columns_df.toDF(*updated_col_names)
>>> renamed_df.show(5)
+------------------+--------------------+--------------------+-----------------+
|        Event_Type|    Involved_Party_A|    Involved_Party_B|           Region|
+------------------+--------------------+--------------------+-----------------+
|Political violence|Military Forces o...|Military Forces o...|      Bakhmutskyi|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|
|Political violence|Military Forces o...|Military Forces o...|Sievierodonetskyi|
|Political violence|Military Forces o...|                None|          Sumskyi|
+------------------+--------------------+--------------------+-----------------+
```

Indexing and Vectorizing Columns (“Independent_Features” is equivalent to “features”. Change made at a later time.)

```
>>> from pyspark.ml.feature import StringIndexer
>>> indexer=StringIndexer(inputCols=['Event_Type','Involved_Party_A','Involved_Party_B','Region'], outputCols=['Indexed_Type','Indexed_Party_A','Indexed_Party_B','Indexed_Region'])
>>> indexed_df = indexer.fit(renamed_df).transform(renamed_df)
>>> indexed_df.show(5)
+------------------+--------------------+--------------------+-----------------+------------+--------------+--------------+--------------+
|        Event_Type|    Involved_Party_A|    Involved_Party_B|           Region|Indexed_Type|Indexed_Party_A|Indexed_Party_B|Indexed_Region|
+------------------+--------------------+--------------------+-----------------+------------+--------------+--------------+--------------+
|Political violence|Military Forces o...|Military Forces o...|      Bakhmutskyi|         0.0|           0.0|           1.0|           0.0|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|         0.0|           0.0|           1.0|           1.0|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|         0.0|           0.0|           1.0|           1.0|
|Political violence|Military Forces o...|Military Forces o...|Sievierodonetskyi|         0.0|           0.0|           1.0|           9.0|
|Political violence|Military Forces o...|                None|          Sumskyi|         0.0|           2.0|           0.0|          12.0|
+------------------+--------------------+--------------------+-----------------+------------+--------------+--------------+--------------+
only showing top 5 rows

>>> from pyspark.ml.feature import VectorAssembler
>>> VectorAssembler(inputCols=['Indexed_Party_A','Indexed_Party_B','Indexed_Region'], outputCol='Independent_Features')
VectorAssembler_64b27d0c5ee3
>>> vec_assem = VectorAssembler(inputCols=['Indexed_Party_A','Indexed_Party_B','Indexed_Region'], outputCol='Independent_Features')
>>> output = vec_assem.transform(indexed_df)
>>> output.show()
+------------------+--------------------+--------------------+-----------------+------------+--------------+--------------+--------------+--------------------+
|        Event_Type|    Involved_Party_A|    Involved_Party_B|           Region|Indexed_Type|Indexed_Party_A|Indexed_Party_B|Indexed_Region|Independent_Features|
+------------------+--------------------+--------------------+-----------------+------------+--------------+--------------+--------------+--------------------+
|Political violence|Military Forces o...|Military Forces o...|      Bakhmutskyi|         0.0|           0.0|           1.0|           0.0|       [0.0,1.0,0.0]|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|         0.0|           0.0|           1.0|           1.0|       [0.0,1.0,1.0]|
|Political violence|Military Forces o...|Military Forces o...|       Pokrovskyi|         0.0|           0.0|           1.0|           1.0|       [0.0,1.0,1.0]|
|Political violence|Military Forces o...|Military Forces o...|Sievierodonetskyi|         0.0|           0.0|           1.0|           9.0|       [0.0,1.0,9.0]|
|Political violence|Military Forces o...|                None|          Sumskyi|         0.0|           2.0|           0.0|          12.0|      [2.0,0.0,12.0]|
|Political violence|Military Forces o...|                None|          Sumskyi|         0.0|           0.0|           0.0|          12.0|      [0.0,0.0,12.0]|
|Political violence|Military Forces o...|                None|Novhorod-Siverskyi|         0.0|           2.0|           0.0|          19.0|      [2.0,0.0,19.0]|
|Political violence|Military Forces o...|Military Forces o...|      Bakhmutskyi|         0.0|           0.0|           1.0|           0.0|       [0.0,1.0,0.0]|
|Political violence|Military Forces o...|                None|      Konotopskyi|         0.0|           0.0|           0.0|          34.0|      [0.0,0.0,34.0]|
|Political violence|Military Forces o...|                None|      Polohivskyi|         0.0|           2.0|           0.0|           2.0|       [2.0,0.0,2.0]|
|Political violence|Military Forces o...|                None|      Kharkivskyi|         0.0|           0.0|           0.0|          10.0|      [0.0,0.0,10.0]|
```

PySpark Cont.

Selecting Features and Splitting the Data

```
>>> ready_df = output.select('Indexed_Type', 'Independent_Features')
>>> ready_df.show(5)
+------------+--------------------+
|Indexed_Type|Independent_Features|
+------------+--------------------+
|         0.0|       [0.0,1.0,0.0]|
|         0.0|       [0.0,1.0,1.0]|
|         0.0|       [0.0,1.0,1.0]|
|         0.0|       [0.0,1.0,9.0]|
|         0.0|      [2.0,0.0,12.0]|
+------------+--------------------+
only showing top 5 rows

>>> train_df, test_df = ready_df.randomSplit([0.75, 0.25])
>>> print(train_df.count())
73513
>>> print(test_df.count())
24536
```

Importing the Decision Tree Classifier and Evaluator

```
>>> from pyspark.ml.classification import DecisionTreeClassifier
>>> from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

Creating a Decision Tree Classifier Instance and Increasing Bin Size

```
>>> classifier = DecisionTreeClassifier(labelCol="Indexed_Type", maxBins=131).fit(train_df)
```

Creating Predictions and Checking the Accuracy

```
>>> predictor = classifier.transform(test_df)
>>> accuracy = MulticlassClassificationEvaluator(labelCol='Indexed_Type', metricName='accuracy').evaluate(predictor)
>>> accuracy
0.9791327029670688
```

Data- https://acleddata.com/ukraine-conflict-monitor/

Data CodeBook- https://acleddata.com/acleddatanew/wp-content/uploads/dlm_uploads/2023/06/ACLED_Codebook_2023.pdf