spark_ml_model-Copy1

May 9, 2024

[1]:
```
import datetime
from numpy import array, sqrt

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf, unix_timestamp, expr, when
from pyspark.sql.types import FloatType

from pyspark.ml import Pipeline
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import StringIndexer, VectorAssembler, StandardScaler,␣
 ↪OneHotEncoder
from pyspark.ml.evaluation import ClusteringEvaluator,␣
 ↪MulticlassClassificationEvaluator, BinaryClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
```

[2]:
```
# Initialize Spark Session
spark = SparkSession.builder \
    .appName("Anomaly Detection Model") \
    .getOrCreate()
spark = SparkSession.builder \
    .appName("KafkaDataSparkAnalysis") \
    .config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10_2.12:
 ↪3.3.0") \
    .getOrCreate()

# the following line gets the bucket name attached to our cluster
bucket = spark._jsc.hadoopConfiguration().get("fs.gs.system.bucket")

# specifying the path to our bucket where the data is located (no need to edit␣
 ↪this path anymore)
data = "gs://" + bucket + "/notebooks/jupyter/"

df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load(data + "data.csv")\
    .coalesce(5)
```

```
# df = df.withColumn("timestamp_unix", unix_timestamp("Timestamp"))


df.cache()
df.printSchema()
print("This datasets consists of {} rows.".format(df.count()))
```

```
24/05/09 16:43:43 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
24/05/09 16:43:43 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
```

```
root
 |-- EventType: string (nullable = true)
 |-- Timestamp: timestamp (nullable = true)
 |-- Location: string (nullable = true)
 |-- Severity: string (nullable = true)
 |-- Details: string (nullable = true)
 |-- Is_Anomaly: integer (nullable = true)
```

```
[Stage 2:==============================>                          (1 + 1) / 2]
```

```
This datasets consists of 1000000 rows.
```

[3]:
```
df.show(25)
```

```
+--------------------+-------------------+-----------+--------+----------------
---+----------+
|           EventType|          Timestamp|   Location|Severity|
Details|Is_Anomaly|
+--------------------+-------------------+-----------+--------+----------------
---+----------+
|  emergency_incident|2022-01-01 00:00:00|     Boston|    high|This is a
simulat…|         0|
|      health_mention|2022-01-01 00:01:00|      Tokyo|     low|This is a
simulat…|         0|
|      health_mention|2022-01-01 00:01:00|      Tokyo|  medium|This is a
simulat…|         0|
|         vaccination|2022-01-01 00:01:00|     Boston|  medium|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:03:00|      Tokyo|  medium|This is a
simulat…|         0|
|  hospital_admission|2022-01-01 00:03:00|    Chicago|  medium|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:03:00|    Chicago|  medium|This is a
```

```
simulat…|         0|
|general_health_re…|2022-01-01 00:05:00|Los Angeles|  medium|This is a
simulat…|         0|
|      health_mention|2022-01-01 00:06:00|      Paris|  medium|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:07:00|      Paris|  medium|This is a
simulat…|         0|
|      health_mention|2022-01-01 00:08:00|   New York|     low|This is a
simulat…|         0|
|      health_mention|2022-01-01 00:09:00|    Chicago|  medium|This is a
simulat…|         0|
|  hospital_admission|2022-01-01 00:10:00|    Chicago|    high|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:10:00|     Berlin|  medium|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:11:00|     Berlin|  medium|This is a
simulat…|         0|
|     routine_checkup|2022-01-01 00:13:00|   Bordeaux|     low|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:15:00|     Berlin|  medium|This is a
simulat…|         0|
|  hospital_admission|2022-01-01 00:17:00|      Paris|    high|This is a
simulat…|         0|
|  emergency_incident|2022-01-01 00:19:00|   New York|    high|This is a
simulat…|         0|
|         vaccination|2022-01-01 00:20:00|      Tokyo|  medium|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:20:00|      Tokyo|  medium|This is a
simulat…|         0|
|general_health_re…|2022-01-01 00:22:00|   Bordeaux|  medium|This is a
simulat…|         0|
|  hospital_admission|2022-01-01 00:23:00|   Bordeaux|  medium|This is a
simulat…|         0|
|         vaccination|2022-01-01 00:24:00|     Berlin|     low|This is a
simulat…|         0|
|  hospital_admission|2022-01-01 00:25:00|Los Angeles|  medium|This is a
simulat…|         0|
+--------------------+-------------------+-----------+--------+----------------
---+----------+
only showing top 25 rows
```

[4]:
```
train_df, test_df = df.randomSplit([0.7, 0.3], seed=42)
print("Training data size: {}".format(train_df.count()))
print("Testing data size: {}".format(test_df.count()))
```

```
Training data size: 700072
```

```
[Stage 9:============================>                            (1 + 1) / 2]
Testing data size: 299928
```

[5]:
```
# Index and encode categorical features
indexer_event = StringIndexer(inputCol="EventType", outputCol="EventType_Index")
indexer_location = StringIndexer(inputCol="Location",␣
 ↪outputCol="Location_Index")

# Convert 'Severity' to a numerical scale
severity_scale = {"low": 1, "medium": 2, "high": 3}
train_df = train_df.withColumn("Severity_Num", when(col("Severity") == "low",␣
 ↪severity_scale["low"])
                                                .when(col("Severity")␣
 ↪== "medium", severity_scale["medium"])
                                                .when(col("Severity")␣
 ↪== "high", severity_scale["high"]))
test_df = test_df.withColumn("Severity_Num", when(col("Severity") == "low",␣
 ↪severity_scale["low"])
                                               .when(col("Severity") ==␣
 ↪"medium", severity_scale["medium"])
                                               .when(col("Severity") ==␣
 ↪"high", severity_scale["high"]))
```

[6]:
```
# Assemble features into a single vector column
assembler = VectorAssembler(inputCols=["EventType_Index", "Location_Index",␣
 ↪"Severity_Num"], outputCol="features")

# Scale the features
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures")
```

[7]:
```
# Define the KMeans model
kmeans = KMeans().setK(9).setSeed(7).setFeaturesCol("scaledFeatures")

# Build the pipeline
pipeline = Pipeline(stages=[indexer_event, indexer_location, assembler, scaler,␣
 ↪kmeans])
```

[8]:
```
# Define a parameter grid
paramGrid = (ParamGridBuilder()
             .addGrid(kmeans.k, [12])  # Number of clusters
             .addGrid(scaler.withStd, [True, False])  # Standard deviation␣
 ↪scaling
             .build())
```

[9]:
```
# Define the evaluator
evaluator = ClusteringEvaluator(predictionCol="prediction",␣
 ↪featuresCol="scaledFeatures", metricName="silhouette")
```

[10]:
```
# Set up the CrossValidator
crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=paramGrid,
                          evaluator=evaluator,
                          numFolds=9)  # Adjusted for demonstration

# Run cross-validation on the training data, and choose the best set of␣
 ↪parameters.
cvModel = crossval.fit(train_df)
```

[11]:
```
# Fetch the best model
bestModel = cvModel.bestModel
bestKMeansModel = bestModel.stages[-1]  # The last stage is KMeans in the␣
 ↪pipeline
```

[12]:
```
# Make predictions on the test data
predictions = bestModel.transform(test_df)
```

[13]:
```
# Now choose an anomaly cluster as before (this part would be manual and␣
 ↪interpretive)
anomaly_cluster = 1  # This needs to be checked based on new cluster centers
predictions = predictions.withColumn("predicted_label", (col("prediction") ==␣
 ↪anomaly_cluster).cast("double"))
```

[14]:
```
predictions.show(25)
```

```
+------------------+-------------------+----------+--------+-------------------
-+----------+-----------+---------------+-------------+------------+---------
----------+----------+---------------+
|         EventType|          Timestamp|  Location|Severity|
Details|Is_Anomaly|Severity_Num|EventType_Index|Location_Index|     features|
scaledFeatures|prediction|predicted_label|
+------------------+-------------------+----------+--------+-------------------
-+----------+-----------+---------------+-------------+------------+---------
----------+----------+---------------+
|emergency_incident|2022-01-01 00:31:00|  New York|    high|This is a
simulat…|         0|          3|            3.0|
5.0|[3.0,5.0,3.0]|[1.75582044243775…|          4|           0.0|
|emergency_incident|2022-01-01 00:42:00|Los Angeles|    high|This is a
simulat…|         0|          3|            3.0|
4.0|[3.0,4.0,3.0]|[1.75582044243775…|          4|           0.0|
|emergency_incident|2022-01-01 00:47:00|   Chicago|    high|This is a
```

```
simulat…|         0|           3|            3.0|
3.0|[3.0,3.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 00:54:00|    Chicago|    high|This is a
simulat…|         0|           3|            3.0|
3.0|[3.0,3.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 01:15:00|   New York|    high|This is a
simulat…|         0|           3|            3.0|
5.0|[3.0,5.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 01:19:00|     Boston|    high|This is a
simulat…|         0|           3|            3.0|
2.0|[3.0,2.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 01:20:00|      Tokyo|    high|This is a
simulat…|         0|           3|            3.0|
1.0|[3.0,1.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 01:48:00|    Chicago|    high|This is a
simulat…|         0|           3|            3.0|
3.0|[3.0,3.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 02:01:00|   Bordeaux|    high|This is a
simulat…|         0|           3|            3.0|
7.0|[3.0,7.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 02:15:00|   New York|    high|This is a
simulat…|         0|           3|            3.0|
5.0|[3.0,5.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 02:18:00|      Paris|    high|This is a
simulat…|         0|           3|            3.0|
6.0|[3.0,6.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 02:54:00|     Berlin|    high|This is a
simulat…|         0|           3|            3.0|
0.0|[3.0,0.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 02:56:00|Los Angeles|    high|This is a
simulat…|         0|           3|            3.0|
4.0|[3.0,4.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 02:57:00|     Berlin|    high|This is a
simulat…|         0|           3|            3.0|
0.0|[3.0,0.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 03:05:00|Los Angeles|    high|This is a
simulat…|         0|           3|            3.0|
4.0|[3.0,4.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 03:24:00|   New York|    high|This is a
simulat…|         0|           3|            3.0|
5.0|[3.0,5.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 03:25:00|      Tokyo|    high|This is a
simulat…|         0|           3|            3.0|
1.0|[3.0,1.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 03:49:00|Los Angeles|    high|This is a
simulat…|         0|           3|            3.0|
4.0|[3.0,4.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 04:03:00|Los Angeles|    high|This is a
```

```
simulat…|         0|           3|            3.0|
4.0|[3.0,4.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 04:06:00|   New York|    high|This is a
simulat…|         0|           3|            3.0|
5.0|[3.0,5.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 04:12:00|     Boston|    high|This is a
simulat…|         0|           3|            3.0|
2.0|[3.0,2.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 04:19:00|      Paris|    high|This is a
simulat…|         0|           3|            3.0|
6.0|[3.0,6.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 04:22:00|      Tokyo|    high|This is a
simulat…|         0|           3|            3.0|
1.0|[3.0,1.0,3.0]|[1.75582044243775…|         2|            0.0|
|emergency_incident|2022-01-01 04:47:00|Los Angeles|    high|This is a
simulat…|         0|           3|            3.0|
4.0|[3.0,4.0,3.0]|[1.75582044243775…|         4|            0.0|
|emergency_incident|2022-01-01 04:58:00|   New York|    high|This is a
simulat…|         0|           3|            3.0|
5.0|[3.0,5.0,3.0]|[1.75582044243775…|         4|            0.0|
+------------------+-------------------+-----------+--------+-------------------
-+----------+------------+---------------+--------------+-------------+---------
-----------+----------+---------------+
only showing top 25 rows
```

[15]:
```
# Calculate Accuracy and F1 Score (as before, or consider using binary␣
 ↪evaluators)
evaluator_accuracy = MulticlassClassificationEvaluator(labelCol="Is_Anomaly",␣
 ↪predictionCol="predicted_label", metricName="accuracy")
accuracy = evaluator_accuracy.evaluate(predictions)

evaluator_f1 = MulticlassClassificationEvaluator(labelCol="Is_Anomaly",␣
 ↪predictionCol="predicted_label", metricName="f1")
f1_score = evaluator_f1.evaluate(predictions)

print(f"Best number of clusters: {bestKMeansModel.getK()}")
print(f"Accuracy: {accuracy}")
print(f"F1 Score: {f1_score}")
```

```
[Stage 657:>                                                        (0 + 2) / 2]
```

```
Best number of clusters: 12
Accuracy: 0.9371682537142247
F1 Score: 0.9673765009491587
```

[ ]:
```

```