Projekat 2

Sistemi za obradu i analizu velike količine podataka

Nemanja Petrović
1499

Opis projekta 3

- Pored funkcija iz biblioteke Pyspark koriste i funkcije vezane za mašinsko učenje (ML)
- Model se trenira iz postojećeg dataset-a i dobija s ena osnovu feature-a a to su id početne i id krajnje stanice
- Kreira se pipeline za obradu podataka
- Podaci se klasifikuju
- Na osnovu modela radi se predikcija vremena putovanja između date dve stanice
- Rezultat se upisuje u InfluxDB i prikazije u Grafani

Treniranje na Spark clusteru

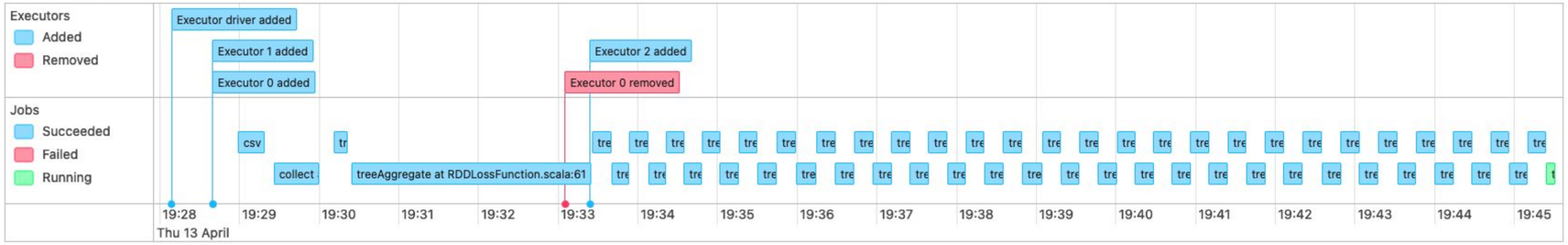
Spark 3.1.2 | Jobs | Stages | Storage | Environment | Executors | SQL | BigDataP3-TrainingNemanja application UI

Spark Jobs (?)

User: root
Total Uptime: 18 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 55

▾ Event Timeline

☐ Enable zooming



▾ Active Jobs (1)

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
55	treeAggregate at RDDLossFunction.scala:61 treeAggregate at RDDLossFunction.scala:61 (kill)	2023/04/13 19:45:23	6 s	0/1	0/1 (1 running)

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

▾ Completed Applications (7)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230413192807-0006	BigDataP3-TrainingNemanja	8	1024.0 MiB		2023/04/13 19:28:07	root	FINISHED	29 min

Značajni delovi koda treniranja

```
dataFrame = spark.read.csv(HDFS_DATA, header=True)

columns = ['start_station_id', 'end_station_id']

for column in columns:
    dataFrame = dataFrame.withColumn(column, F.col(column).cast(FloatType()))

vectorAssembler = VectorAssembler().setInputCols(columns).setOutputCol('features').setHandleInvalid('skip')
```

Kreiranje feature-a

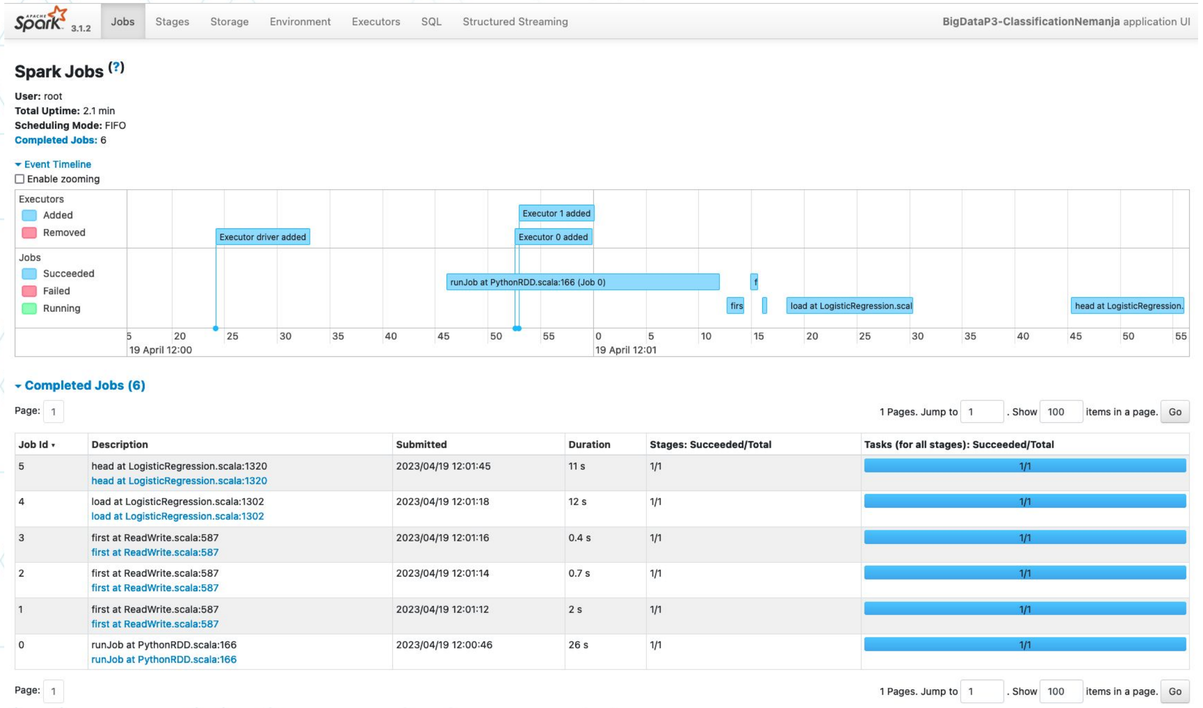
```
assembled = vectorAssembler.transform(dataFrame)

stringIndexer = StringIndexer().setInputCol('duration').setOutputCol('label')
indexedDataFrame = stringIndexer.fit(assembled).transform(assembled)

train_split, test_split = indexedDataFrame.randomSplit([0.8, 0.2], seed=1337)

print("Starting training")

regressionModel = LogisticRegression(maxIter=100, regParam=0.02, elasticNetParam=0.8)

pipeline = Pipeline(stages=[regressionModel])
regressionModelPipe = pipeline.fit(train_split)
```

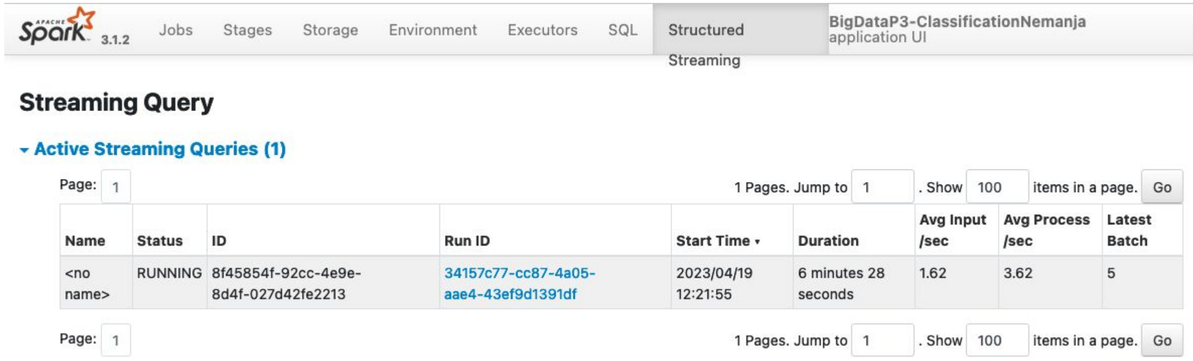
Treniranje

```
prediction = regressionModelPipe.transform(test_split)

evaluator = BinaryClassificationEvaluator(labelCol='label', rawPredictionCol='prediction',
                                          metricName='areaUnderPR')
print("Starting evaluation")
accuracy = evaluator.evaluate(prediction)

print('Accuracy\'s value for logistic regression model is ' + str(accuracy) + '!')

regressionModelPipe.write().overwrite().save(MODEL)
```

Evaluacija modela i njegovo čuvanje

Klasifikacija na Spark clusteru





Delovi koda aplikacije za klasifikaciju

```
def analyze(df, epoch, model):
    print("Epoch " + str(epoch))
    columns = ['start_station_id', 'end_station_id']

    for column in columns:
        df = df.withColumn(column, F.col(column).cast(FloatType()))

    vectorAssembler = VectorAssembler().setInputCols(columns).setOutputCol('features').setHandleInvalid('skip')

    assembled = vectorAssembler.transform(df)

    stringIndexer = StringIndexer().setInputCol('duration').setOutputCol('label')
    indexedDataFrame = stringIndexer.fit(assembled).transform(assembled)

    prediction = model.transform(indexedDataFrame)

    prediction.select('prediction', 'label')
    # prediction.show(truncate=False)

    predictionsMade = prediction.count()

    correctNumber = float(prediction.filter(prediction['label'] == prediction['prediction']).count())

    influxDBwrite(df.count(), predictionsMade, correctNumber)
```

Analiza jednog seta sa Kafke

```
model = PipelineModel.load(MODEL)
spark.sparkContext.setLogLevel("INFO")
sampleDataframe = (
    spark.readStream.format("kafka")
        .option("kafka.bootstrap.servers", kafka)
        .option("subscribe", topic)
        .option("startingOffsets", "earliest")
        .load()
).selectExpr("CAST(value as STRING)", "timestamp").select(
    from_json(col("value"), dataSchema).alias("sample"), "timestamp"
).select("sample.*")


sampleDataframe.writeStream \
    .foreachBatch(lambda df, epoch_id: analyze(df, epoch_id, model)) \
    .outputMode("update") \
    .trigger(processingTime="10 seconds") \
    .start().awaitTermination()
```

Učitavanje modela i čitanje sa Kafke

```
def influxDBconnect():
    return InfluxDBClient(dbhost, dbport, dbuser, dbpassword, dbname)


def influxDBwrite(count, predictions, accuracy):
    timestamp = datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%SZ')
    measurementData = [
        {
            "measurement": topic,
            "time": timestamp,
            "fields": {
                "number of rows": count,
                "predictions": count,
                "correct predictions": accuracy
            }
        }
    ]
    print(measurementData)
    influxDBConnection.write_points(measurementData, time_precision='ms')
    print("Count " + str(count))
    print("Accuracy " + str(accuracy))
```

Konekcija na InfluxDB i upis

Vizuelizacija u grafani

Prikazuje se broj podataka analiziranih u tom trenutku i broj tačno predviđenih

Performanse aplikacije

Treniranje aplikacije na klusteru je bilo jako sporo iz već objašnjenog razloga, kluster se nalazi na mapini sa M1 procesorom
Sama klasifikacija radi jako brzo i potrebno je oko 3 sekundi po jednoj epohi (Sa Kafke)

BADIN