UVOD U APACHE Spark

Petar Zečević

SV GROUP JavaCro'15

SV Group d.o.o.

- **Prvi** IBM Business Partner Premier Level (od 14.3. 2003.)
- *Kontinuirana priznanja IBM CEMA (od 2000. god.)*
- Gazela za 2007 i 2008.
- ISO 9001:2002 od 2007.
- ISO 9001:2008 od 2010.
- VMware Enterprise Solution Partner
- RedHat i Microsoft partner

SV GROUP

SV Group d.o.o.

- **Financijska industrija:**
  - Zagrebačka banka
  - Privredna banka Zagreb
  - Hypo Alpe Adria Bank
  - Hrvatska poštanska banka
  - Erste bank
  - Generali osiguranje, Uniqa , Prva stambena štedionica
  - Slatinska banka, Credo banka, Centar banka, BKS
- **Državne institucije**
  - HZMO
  - MUP
  - Ministarstvo financija
  - APIS IT, FINA
  - Regos, AKD
- **Distribucija**
  - Agrokor/mStart
  - Konzum, Multipluscard
  - Tisak
- **Industrija**
  - INA
- **HR IT tvrtke**

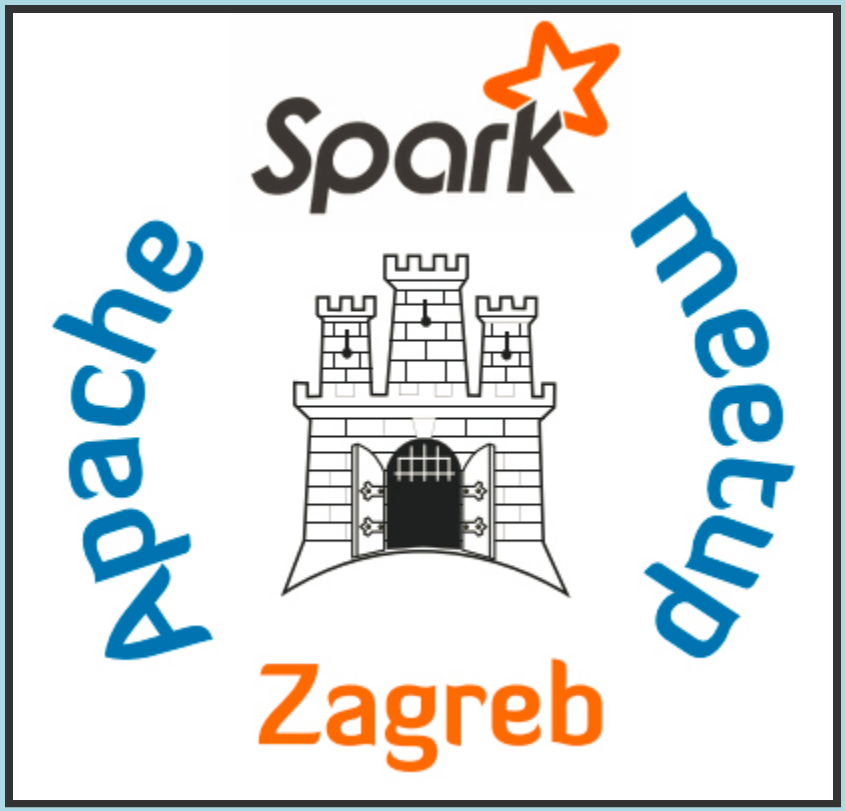
JavaCro'15

POPUST OD 50%!

http://www.manning.com/bonaci/

JavaCro'15

APACHE SPARK ZAGREB MEETUP GRUPA



http://www.meetup.com/Apache-Spark-Zagreb-Meetup/

JavaCro'15

MALA ANKETA

- Prvi put čujem za Spark
- Čuo sam za Spark, ali nisam ga koristio
- Instalirao sam Spark i pokretao jobove
- Spremam se koristiti Spark u produkciji
- Već ga koristim u produkciji

JavaCro'15

UVOD U APACHE SPARK

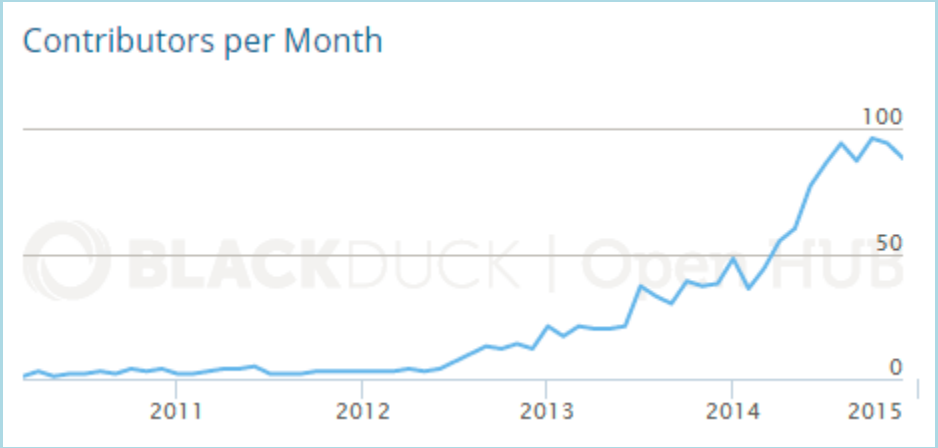
- Malo povijesti
- U čemu je problem?
- O Hadoopu
- Što donosi Spark?
- Runtime arhitektura
- Spark komponente
- Demo
- Pitanja i rasprava

MALO POVIJESTI

POVIJEST SPARKA

JavaCro'15

- Nastao 2009. godine u AMPLabu Berkley-a
- 2010. godine postao open source
- Strelovit rast:



- Danas ima više od 400 commitera
- Matei Zaharia

i databricks

SPARK DANAS

- Uključen u najveće Hadoop distribucije:

- Dvije godišnje konferencije:

- Tečajevi i certifikacije
- spark-packages.org - trenutno 55 paketa
- Polako zamjenjuje Hadoop kao sinonim za BigData

ZAŠTO SPARK?

JavaCro'15

KAKO EFIKASNO OBRADITI GOLEME KOLIČINE PODATAKA?

- 


ne stanu u RDBMS
- Ogroman rast broja korisnika (Facebook 8 k/s, Twitter 6 k/s, LinkedIn 2 k/s)
- Većina korisnika nije pasivna
- NASA, genetski algoritmi, industrijski senzori, simulacije...
- NASA generira 700 TB/s

HDFS - nastao kao implementacija Google FS-a (2003)

- Običan desktop (commodity) hardware
- Svaki podatak repliciran na 3 mjesta (default)

MapReduce

- Kako paralelizirati obradu podataka
- Program se šalje k podacima (program je manji)
- Oporavak od grešaka

YARN (MapReduce 2)

- Generalizacija resource managementa

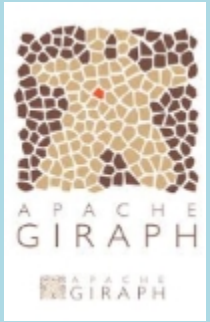
JavaCro'15

SLABOSTI HADOOPA

- Rigidan programski model
- Zadaci prikladni za funkcionalno programiranje
- Pisanje na disk je nužno, kao i sortiranje
- Ne može se koristiti za realtime (lambda arhitektura)
- Low-level framework

JavaCro'15

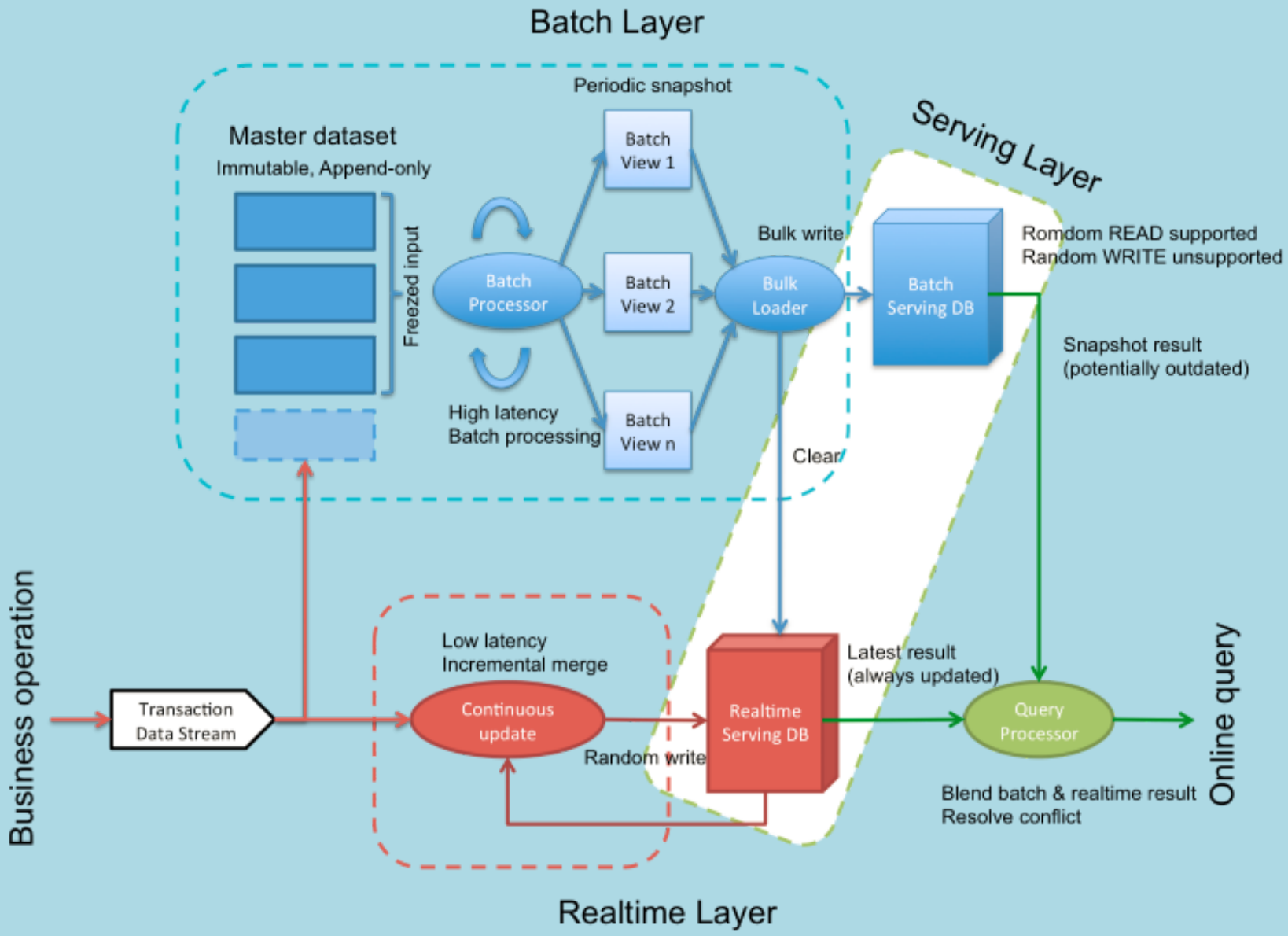
HADOOP ECOSYSTEM



LAMBDA ARHITEKTURA

JavaCro'15

Kako imati ažurnu sliku podataka?


Batch Layer
Periodic snapshot
Master dataset
Immutable, Append-only
Freezed input
Batch Processor
Batch View 1
Batch View 2
Batch View n
High latency
Batch processing
Bulk write
Bulk Loader
Serving Layer
Batch Serving DB
Romdom READ supported
Random WRITE unsupported
Snapshot result
(potentially outdated)
Clear
Business operation
Transaction Data Stream
Low latency
Incremental merge
Continuous update
Random write
Realtime Serving DB
Latest result
(always updated)
Query Processor
Blend batch & realtime result
Resolve conflict
Online query
Realtime Layer

ŠTO DONOSI SPARK?

JavaCro'15

APACHE SPARK

- Može se koristiti i za **batch i realtime** obradu podataka
- Često nije potrebno spremati međupodatke na disk
- Podatke uglavnom **drži u memoriji** - i do **100x** brži od Hadoop M/R-a
- Pisan u **Scali**, podržava **Python i Javu**
- Izvršava se na Standalone, YARN ili Mesos clusteru
- **Jedinstvena platforma** za različite vrste algoritama
- Brži razvoj kroz **Scala i Python shell**
- Pisanje distribuiranih programa **slično pisanju lokalnih**

JavaCro'15

HADOOP M/R WORD COUNT - MAIN

```
public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("Hadoop wordcount");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
  }
```

JavaCro'15

HADOOP M/R WORD COUNT - MAPPER

```
public static class Map extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable>
{
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output, Reporter reporter) throws
  {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
      word.set(tokenizer.nextToken());
      output.collect(word, one);
    }
  }
}
```

JavaCro'15

HADOOP M/R WORD COUNT - REDUCER

```
public static class Reduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable>
{
  public void reduce(Text key, Iterator<IntWritable> values,
    OutputCollector<Text, IntWritable> output, Reporter reporter) throws
  {
  int sum = 0;
  while (values.hasNext()) {
    sum += values.next().get();
  }
  output.collect(key, new IntWritable(sum));
}
}
```

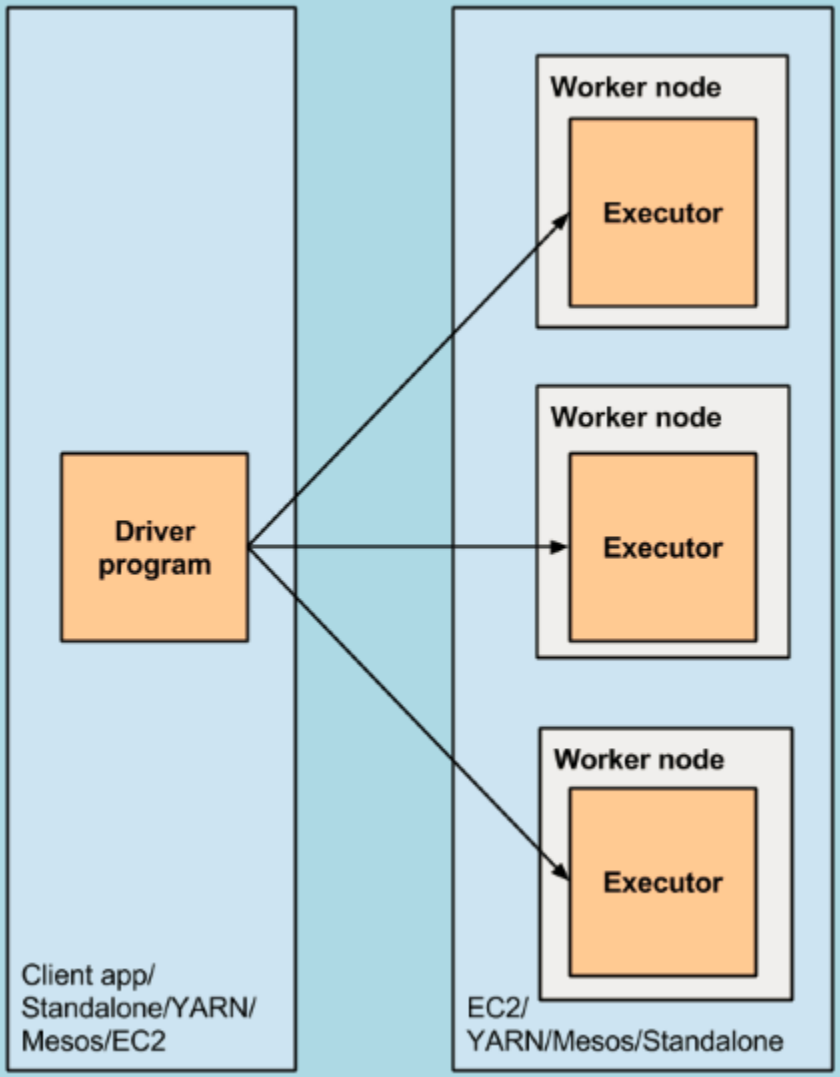
JavaCro'15

SPARK WORD COUNT IN SCALA

```
val conf = new SparkConf().setAppName("Spark wordcount")
val sc = new SparkContext(conf)
val file = sc.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
    .map(word => (word, 1))
    .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

SPARK KOMPONENTE

RUNTIME TOPOLOGIJA



JavaCro'15

SPARK CORE

- RDD - Resilient Distributed Dataset
- Spark Web UI
- Spark shell
- Učitavanje i spremanje datoteka
- Serijalizacija i deserijalizacija taskova
- Broadcast varijable i akumulatori

SPARK CORE

Detaljnije o RDDovima

- Lista particija
- Funkcija za izračun podataka unutar particija
- Transformacije - pretvaraju jedan RDD u drugi
  - map, reduce, filter, zip, pipe
  - mapByKey, reduceByKey, mapPartitions, subtract, union, ...
- Akcije - vraćaju neki rezultat
  - collect, count, foreach
  - first, take, aggregate, stats, ...
- Dependencies na druge RDD-ove
- Keširanje

JavaCro'15

SPARK SQL

- Obrada strukturiranih podataka
- Učitavanje i spremanje JSON, Parquet datoteka
- Schema inference
- Razumije SQL naredbe
- Spremanje podataka o tablicama u Hive Metastore
- Thrift JDBC server

JavaCro'15

SPARK STREAMING

- Spark API primijenjen na realtime obrade
- "Exactly once" semantika out of the box
- Čita podatke s TCPIP porta, HDFS-a, S3, Kafke, ZeroMQ, Flume-a, ...

JavaCro'15

SPARK STREAMING

JavaCro'15

SPARK STREAMING - EXAMPLE

```
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWord
val ssc = new StreamingContext(conf, Seconds(1))

val lines = ssc.socketTextStream("localhost", 9999)
val words = lines.flatMap(_.split(" "))

val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)

ssc.start()
ssc.awaitTermination()
```

JavaCro'15

SPARK GRAPHX

- Mnogi problemi se mogu rješavati pomoću grafova - socijalne mreže, GPS mape, napredovanje bolesti itd.
- Graf algoritmi uz ostale funkcionalnosti Sparka
- Vertices (vrhovi), edges (grane) i vezani user-defined objekti
- Implementacija Pregela - slanje poruka kroz graf
- Algoritmi: Page rank, shortest path, connected components, triangle counting ...

JavaCro'15

SPARK MLLIB

- Linearna algebra (vektori, matrice)
- Klasifikacija i regresija
  - Linearne metode
    - Logistička regresija
    - Support Vector Machines
    - Linearna regresija (Linear least squares)
  - Decision trees
  - Naive Bayes (klasifikacija dokumenata)
- k-means clustering
- Collaborative filtering (alternating least squares)
- Redukcija dimenzija

JavaCro'15

SPARK CORE - DEMO

```
spark-shell --master yarn-client --num-executors 3 --executor-memory 2G

val licrdd = sc.textFile("hdfs:///tmp/LICENSE.txt", 6)
val trimmed = licrdd.map(x => x.trim)
val noempty = trimmed.filter(x => x != "")
val split = noempty.flatMap(x => x.split(" ").filter(_ != ""))
val pairs = split.map(x => (x, 1))
val sums = pairs.reduceByKey(_ + _)
val sorted = sums.sortBy(_._2, false)
sorted.top(10)
```

JavaCro'15

SPARK SQL - DEMO

```
import org.apache.spark.sql.hive.HiveContext
val sqlContext = new HiveContext(sc)
import sqlContext._
val json = sqlContext.jsonFile("hdfs:///tmp/test.json")
json.schema
json.registerTempTable("myusers")
val sel = sqlContext.sql("select * from myusers")
val countries = sel.map(row => row.getString(0))
val countriesd = sqlContext.sql("select distinct country from myusers")
val franceusers = sqlContext.sql("select last_name, first_name from myuse
```

JavaCro'15

SPARK MLLIB - DEMO

```
import org.apache.spark.mllib.classification.NaiveBayes
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
val data = sc.textFile("hdfs:///tmp/numberdata.txt", 6)
val parsedData = data.map( line => {
  val parts = line.split(',')
  LabeledPoint(parts(0).toDouble, Vectors.dense(parts.slice(1, parts.leng
})
val model = NaiveBayes.train(parsedData, lambda=1.0)
val testitem = ""
val testdata = sc.parallelize(List(testitem))
val mapped = testdata.map(line => Vectors.dense(line.split(",").map(_.toI
model.predict(mapped).collect()
```

JavaCro'15

SPARK MLLIB - DEMO

Sliku  pretvorimo u niz 0 i 1:

```
val testitem = "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
```

Rezultat predikcije:

```
model.predict(mapped).collect()
res0: Array[Double] = Array(1.0)
```

PITANJA, KOMENTARI?