# Treci projekat iz Sistema za obradu velike količine podataka

Danilo Veljović, 1120

Katedra za računarstvo i informatiku
Univerzitet u Nišu, Elektronski fakultet

18. februar 2021.

# Sadržaj

# Novi model podataka

Model podataka koji se koristi se nalazi u src/main/java/project3/ExtendedRecord.java klasi. Jedina razlika koja postoji u odnosu na Record klasu koja je korišćena u prethodnim primerima je to što ExtendedRecord klasa sadrži polje test i testBool. Test polje se formira kao linearna kombinacija polja tripDuration i koristi se za testiranje algoritma regresije. Polje testBool se formira kao indikator da li je dužina puta bila duža od 1000 i da li je pol muški, i može služiti kao svojevrsni indikator da li je tu vožnju vozio muškarac koji je u formi. Ovo polje se koristi za testiranje algoritma klasifikacije. (U ovom projektu će oba algoritma biti testirana). Skup podataka se deli na trening i test skup podataka. Trening skup podataka se čita u Kafka Analyzer-u. Test skup se čita i šalje kroz Kafka producer.

# Kafka producer

Na slici ispod je dat deo koda koji implementira Kafka Producer klasu. Ona čita test skup podataka i šalje ga na topic.

```
    while(line != null){

        ProducerRecord<String, String> rec = new ProducerRecord<>(topic,  line);

        kafkaProducer.send(rec);

        System.out.println("[PRODUCER] Sent message: " + line);
        //LOGGER.debug("[PRODUCER] Sent message: " + line);
        line = reader.readLine();

        Thread.sleep(millisToSleep);
    }
```

# Kafka Analyzer

Kafka Analyzer mora da se poveže na topic i na HDFS. Na HDFS-u je smešten train.csv dataset koji služi da se njime trenira model. Na slici ispod je dat kod koji ilustruje treniranje modela za lineranu regresiju (zakomentarisani deo koda je kod koji ilustruje treniranje modela za klasifikaciju). Parametri train metode su skup podataka, broj iteracija i parametar alfa tj brzina učenja, tj brzina kojom se metoda kreće ka globalnom minimumuu.

```
//training the model
JavaRDD<ExtendedRecord> rdd_records = sc.textFile(DATA_FILE).map(Parse::parseStringToExtendedRecord);

JavaRDD<LabeledPoint> parsedData = rdd_records.map(s -> new LabeledPoint(s.getTripDuration(), Vectors.dense(s.getTest()))); //regression
LinearRegressionModel model = LinearRegressionWithSGD.train(parsedData.rdd(), 100, 0.00000001); //regression

/*JavaRDD<LabeledPoint> parsedData = rdd_records.map(s -> new LabeledPoint(s.getTestBool(), Vectors.dense(s.getGender(), s.getTripDuration())));

LogisticRegressionModel logisticRegressionModel = new LogisticRegressionWithLBFGS().setNumClasses(2).run(parsedData.rdd());

ClassificationModel naiveBayesModel = new NaiveBayes().run(parsedData.rdd());*/
```

# Kafka Analyzer

Nakon treniranja modela Analyzer se povezuje na topic i na njemu čita podatke. Zatim nad tim podacima vrši testiranje modela odnosno za pročitani podatak određuje, u slučaju regresije, vrednost test atributa, a u slučaju klasifikacije određuje labelu klase kojoj pripada.Na slici ispod je dat deo koda za parsiranje podataka primljenih sa topica.

```
JavaDStream<ExtendedRecord> accessLogDStream = logDataDStream.map(
        new Function<Tuple2<String, String>, ExtendedRecord>() {
            public ExtendedRecord call(Tuple2<String, String> message) {
                String strLogMsg = message._2();
                return Parse.parseStringToExtendedRecord(strLogMsg);
            }
        }
);
```

# Kafka Analyzer

Na slici ispod je dat kod za testiranje modela linearne regresije.

```
JavaRDD<Tuple2<Integer, Double>> valuesAndPred = rdd_records1
        .map(point -> new Tuple2<>(point.getTripDuration(), model
                .predict(Vectors.dense(point.getTest())))
        ));

valuesAndPred.collect().forEach(t -> System.out.println(t._1 + " " + t._2));

return null;
```

# Kafka Analyzer

Na slici ispod je dat kod za testiranje modela klasifikacije. I to logističke regresije i Naive Bayes modela.

```
        System.out.println("LOGISTIC REGRESSION:");

    JavaRDD<Tuple2<Integer, Double>> valuesAndPred = rdd_records1
                .map(s -> new Tuple2<>(s.getTripDuration(), logisticRegressionModel
                        .predict(Vectors.dense(s.getTestBool(), s.getTripDuration())))
                );

        valuesAndPred.collect().forEach(t -> System.out.println(t._1 + " " + t._2));

        System.out.println("NAIVE BAYES:");

        valuesAndPred = rdd_records1
                .map(s -> new Tuple2<>(s.getTestBool(), naiveBayesModel
                        .predict(Vectors.dense(s.getGender(), s.getTripDuration())))
                );

        valuesAndPred.collect().forEach(t -> System.out.println(t._1 + " " + t._2));
```

# The End