LAPORAN TUGAS PERTEMUAN 13

BIG DATA

Oleh :

MOCHAMMAD ZAKARO AL FAJRI 2241720175

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

Tugas 13 – Machine Learning dengan Big Data

Soal

1. Implementasikan studi kasus pada slide ini di cluster Spark masing-masingImplementasikan studi kasus pada slide ini di cluster Spark masing-masing
2. Implementasikan juga studi kasus pada slide ini tetapi menggunakan Pipeline Spark MLlib.
3. Simpan Pipeline yang dibuat pada Soal No. 2, lalu load-lah ke Spark Job yang lain, namun dengan data latih yang berbeda. • Anda bisa membuat sendiri data latihnya yang berbeda.

Jawaban

1.
   - Persiapan data

```python
from pyspark.sql import SparkSession
from pyspark.ml.feature import Tokenizer, HashingTF
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# 1. Inisialisasi Spark
spark = SparkSession.builder \
    .appName("SentimenAnalisisSpark") \
    .master("spark://spark-master:7077") \
    .getOrCreate()

# 2. Contoh Data (teks+ label: 1 = positif, 0 = negatif)
data = spark.createDataFrame([
    (0, "Saya sangat senang belajar Spark", 1.0),
    (1, "Tutorial ini sangat membantu", 1.0),
    (2, "Saya bingung dan frustrasi", 0.0),
    (3, "Instalasi Spark terlalu rumit", 0.0),
    (4, "Belajar Spark itu menyenangkan", 1.0),
    (5, "Saya tidak suka error terus-menerus", 0.0)
], ["id", "teks", "label"])
```

   - Preprocessing

```python
# 3. Tokenisasi manual
tokenizer = Tokenizer(inputCol="teks", outputCol="kata")
data_tokenized = tokenizer.transform(data)

# 4. Konversi ke fitur numerik (HashingTF)
hashingTF = HashingTF(inputCol="kata", outputCol="fitur", numFeatures=20)
data_featurized = hashingTF.transform(data_tokenized)
```

- Melakukan split data

```
# 5. Split data
trainingData, testData = data_featurized.randomSplit([0.7, 0.3], seed=42)
```

- Melakukan training model

```
# 6. Latih model Logistic Regression
lr = LogisticRegression(featuresCol="fitur", labelCol="label")
model = lr.fit(trainingData)

# 7. Prediksi
prediksi = model.transform(testData)
```

- Evaluasi dan hasil

```
# 8. Evaluasi akurasi
evaluator = BinaryClassificationEvaluator(labelCol="label", rawPredictionCol="rawPrediction")
akurasi = evaluator.evaluate(prediksi)
print(f"Akurasi Model: {akurasi:.2f}")

# Stop Spark
# spark.stop()
```

- Hasil dari implementasi yaitu

```
Akurasi Model: 0.00
```

akurasi dari model 0.00 karena semua prediksi pada data uji salah. Dapat dilihat bahwa data uji terdiri dari Label Asli: 0.0 (negatif) dan Prediksi Model: 1.0 (positif)

```
+--------------------+-----+----------+--------------------+--------------------+
|                teks|label|prediction|       rawPrediction|         probability|
+--------------------+-----+----------+--------------------+--------------------+
|Saya bingung dan ...|  0.0|       1.0|[-2.7356786779175...|[0.06090057861597...|
|Instalasi Spark t...|  0.0|       1.0|[-11.681638690494...|[8.44743985729459...|
+--------------------+-----+----------+--------------------+--------------------+
```

yang keduanya salah sehingga hasil akurasi pun salah (0.00)

2.

- Persiapan Data

```
from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.feature import Tokenizer, HashingTF
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# 1. Inisialisasi Spark
spark = SparkSession.builder \
    .appName("SentimenAnalisisSparkPipeline") \
    .master("spark://spark-master:7077") \
    .getOrCreate()

# 2. Contoh Data
data = spark.createDataFrame([
    (0, "Saya sangat senang belajar Spark", 1.0),
    (1, "Tutorial ini sangat membantu", 1.0),
    (2, "Saya bingung dan frustrasi", 0.0),
    (3, "Instalasi Spark terlalu rumit", 0.0),
    (4, "Belajar Spark itu menyenangkan", 1.0),
    (5, "Saya tidak suka error terus-menerus", 0.0)
], ["id", "teks", "label"])
```

- Preprocessing

```
# 3. Definisikan tahapan pipeline
tokenizer = Tokenizer(inputCol="teks", outputCol="kata")
hashingTF = HashingTF(inputCol="kata", outputCol="fitur", numFeatures=20)
lr = LogisticRegression(featuresCol="fitur", labelCol="label")

pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
```

- Melakukan split data

```
# 4. Split data
trainingData, testData = data.randomSplit([0.7, 0.3], seed=42)
```

- Melakukan training model

```
# 5. Latih model dengan pipeline
model = pipeline.fit(trainingData)
```

- Evaluasi dan hasil

```
# 7. Evaluasi
evaluator = BinaryClassificationEvaluator(labelCol="label", rawPredictionCol="rawPrediction")
akurasi = evaluator.evaluate(prediksi)
print(f"Akurasi Model: {akurasi:.2f}")
```

```
Akurasi Model: 0.00
+--------------------+-----+----------+--------------------+--------------------+
|                teks|label|prediction|       rawPrediction|         probability|
+--------------------+-----+----------+--------------------+--------------------+
|Saya bingung dan ...|  0.0|       1.0|[-2.7356786779175...|[0.06090057861597...|
|Instalasi Spark t...|  0.0|       1.0|[-11.681638690494...|[8.44743985729459...|
+--------------------+-----+----------+--------------------+--------------------+
```

3.

- Menyimpan pipeline yang dibuat

```
model.write().overwrite().save("full_pipeline_lr")
```

- menggunakan job yang berbeda

```
# 2. Data baru (berbeda dari sebelumnya)
data_baru = spark.createDataFrame([
    (0, "Spark membantu pekerjaan saya", 1.0),
    (1, "Saya sangat kecewa dengan Spark", 0.0),
    (2, "Spark membuat analisis lebih mudah", 1.0),
    (3, "Error terus muncul saat training", 0.0),
    (4, "Saya puas dengan performa Spark", 1.0),
    (5, "Spark sangat lambat dan mengecewakan", 0.0),
    (6, "Tutorial Spark sangat mudah diikuti", 1.0),
    (7, "Saya frustrasi saat menggunakan Spark", 0.0),
    (8, "Spark membuat pekerjaan lebih efisien", 1.0),
    (9, "Proses debugging di Spark sangat rumit", 0.0),
    (10, "Pengolahan data dengan Spark sangat cepat", 1.0),
    (11, "Saya kesal karena Spark sering error", 0.0),
], ["id", "teks", "label"])

# 3. Load pipeline model dari penyimpanan sebelumnya
model_loaded = PipelineModel.load("full_pipeline_lr")

# 4. Gunakan pipeline untuk memproses dan memprediksi data baru
hasil = model_loaded.transform(data_baru)

# 5. Tampilkan hasil prediksi
hasil.select("teks", "label", "prediction", "probability").show(truncate=False)
```

- evaluasi dan hasil

```
+----------------------------------------+-----+----------+----------------------------------------+
|teks                                    |label|prediction|probability                             |
+----------------------------------------+-----+----------+----------------------------------------+
|Spark membantu pekerjaan saya           |1.0  |0.0       |[0.9999438071787333,5.619282126667624E-5] |
|Saya sangat kecewa dengan Spark         |0.0  |0.0       |[0.8191737639224606,0.18082623607753945]  |
|Spark membuat analisis lebih mudah      |1.0  |0.0       |[0.9999999667375052,3.326249475854581E-8] |
|Error terus muncul saat training        |0.0  |1.0       |[0.006649578207338286,0.9933504217926618] |
|Saya puas dengan performa Spark         |1.0  |1.0       |[0.14831413534476345,0.8516858646552365]  |
|Spark sangat lambat dan mengecewakan    |0.0  |0.0       |[0.9999976048576855,2.3951423144819017E-6]|
|Tutorial Spark sangat mudah diikuti     |1.0  |1.0       |[2.217631342881039E-8,0.9999999778236865] |
|Saya frustrasi saat menggunakan Spark   |0.0  |1.0       |[0.3371114988131191,0.6628885011868809]   |
|Spark membuat pekerjaan lebih efisien   |1.0  |0.0       |[0.9999999999999998,2.220446049250313E-16]|
|Proses debugging di Spark sangat rumit  |0.0  |1.0       |[0.011212250970830805,0.9887877490291692] |
|Pengolahan data dengan Spark sangat cepat|1.0 |1.0       |[4.830666003319861E-4,0.999516933399668]  |
|Saya kesal karena Spark sering error    |0.0  |0.0       |[0.9999965316219306,3.4683780694155786E-6]|
+----------------------------------------+-----+----------+----------------------------------------+
```