# LAPORAN

# RENCANA TUGAS MANDIRI (RTM) Ke-5

# MATA KULIAH BIG DATA (A)

# “AUTOMATED SCORING SYSTEM MENGGUNAKAN PYSPARK”


**DISUSUN OLEH:**

Reza Putri Angga (22083010006)

**DOSEN PENGAMPU:**

Tresna Maulana Fahrudin S.ST., M.T. (NIP. 199305012022031007)

**PROGRAM STUDI SAINS DATA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN” JAWA TIMUR**


**2024**

# STUDI KASUS DAN PEMBAHASAN

Dengan menggunakan PySpark dilakukan pembuatan sistem penskoran secara otomatis pada soal *essay*. Dengan menggunakan dataset sebagai data latih yang diambil dari hasil ujian mahasiswa menggunakan sejumlah soal tertentu. Algoritma yang digunakan untuk memprediksi menggunakan *Alternating Least Square* dengan *output* program berupa akurasi dan pengujian menggunakan data uji.

Pada penugasan ini dilakukan pembuatan kode *script* untuk melakukan penskoran secara otomatis dari soal *essay* menggunakan dua metode pendekatan, yakni pendekatan *alternating least square* (ALS) dan metode regresi dengan tujuan untuk mengetahui metode mana yang lebih baik untuk dipergunakan penskoran otomatis dari jawaban soal *essay*. Mengenai kode *script* yang berisi langkah-langkah dan perhitungan akan dijelaskan lebih lanjut sebagai berikut.

**Automated Scoring System Menggunakan Alrernating Least Square (ALS)**

*Automated scoring sytem* atau sistem penilaian otomatis merupakan sistem yang dirancang secara otomatis untuk mengevaluasi atau menilai jawaban dari pertanyaan *essay*. *Altermating least square* (ALS) merupakan algoritma kolaboratif filtering yang digunakan suatu sistem rekomendasi dan analisis data. Sehingga, *aturomated scoring system* menggunakan *alternating least square* (ALS) mengacu pada penggunaan algoritma ALS.

Dimana untuk membuat program ini diperlukan penggunakaan *library* atau *package* PysSpark yang berfungsi untuk melakukan pemorsesan data terdistribusi, pengolahan data paralel, analisis data interaktif, skalabilitas, dan integrasi dengan sumber data beragam (big data). Sehingga, dengan menggunakan PySpark dalam pembuatan sistem penskoran otomatis pada soal *essay* dapat menghasilkan model yang akurat dan efisien dalam menilai jawaban *essay* otomatis.

Dengan demikian, terdapat penilaian otomatis atas jawaban-jawaban *essay* tersebut tanpa intervensi manual secara signifikan. Untuk *link* laporan dan kode *script* dapat diakses https://tinyurl.com/06RTM-5. Terdapat beberapa tahapan atau proses untuk melakukan pembuatan prediksi skor menggunakan ALS, di antaranya yakni.

**1. Mengimport Modul Dan Membuat Session**

Merupakan proses untuk meberikan akses ke fungsi dan pembuatan *session* yang diperlukan untuk manipulasi data dan pengembangan model.

**Mengimport Modul Dan Membuat Session**

```
#mengimport modul yang dibutuhkan
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

#membuat session
appName = "Automated Scoring System ALS"
spark = SparkSession \
    .builder \
    .appName(appName) \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

*proses import modul dan membuat session*

Pada kode *script* diatas, dilakukan penggunaan *library* PySpark yang sebelumnya telah di *install* menggunakan perintah "*!pip install pyspark*". Dengan menggunakan modul *SparkSession* untuk membuat *session* bernama "Automated Scoring System ALS" untuk menginisiasi lingkungan Spark dan *Col* untuk mengakses kolom dari *dataframe*. Kode tersebut menyiapkan lingkungan kerja Spark dengan konfigurasi yang diperlukan dalam pengolahan data pembuatan sistem penskrosan otomatis soal *essay* menggunakan algoritma ALS.

## 2. Membaca Dataset

Merupakan proses untuk memahami data (sturuktur dan karakteristik dataset).

**Membaca Dataset**

```
import pandas as pd

#dikarenakan pyspark tidak mendukung pembacaan langsung dari format file excel
#baca file excel menggunakan pandas
data_pandas = pd.read_excel("training_data_essay.xlsx")

#buat dataframe spark dari dataframe pandas
data = spark.createDataFrame(data_pandas)
```

```
#membaca dataset
data.show(10)
```

```
+---+------------+--------------------+----+-------------+
|npm|nama_peserta|             jawaban|soal|skor_per_soal|
+---+------------+--------------------+----+-------------+
|  0|       Admin|Tidak, Hanya memb...|   1|        100.0|
|  0|       Admin|Biaya dihitung be...|   2|        100.0|
|  0|       Admin|Hak cipta adalah ...|   3|        100.0|
|  0|       Admin|Dijelaskan kepada...|   4|        100.0|
|  0|       Admin|1. Melindungi dan...|   5|        100.0|
|  0|       Admin|Ruang Komputer, P...|   6|        100.0|
|  0|       Admin|Aturlah posisi pe...|   7|        100.0|
|  0|       Admin|Posisi Kepala dan...|   8|        100.0|
|  0|       Admin|1. Kecocokan soft...|   9|        100.0|
|  0|       Admin|1. Fokus dan expo...|  10|        100.0|
+---+------------+--------------------+----+-------------+
only showing top 10 rows
```

*proses load dan show dataset*

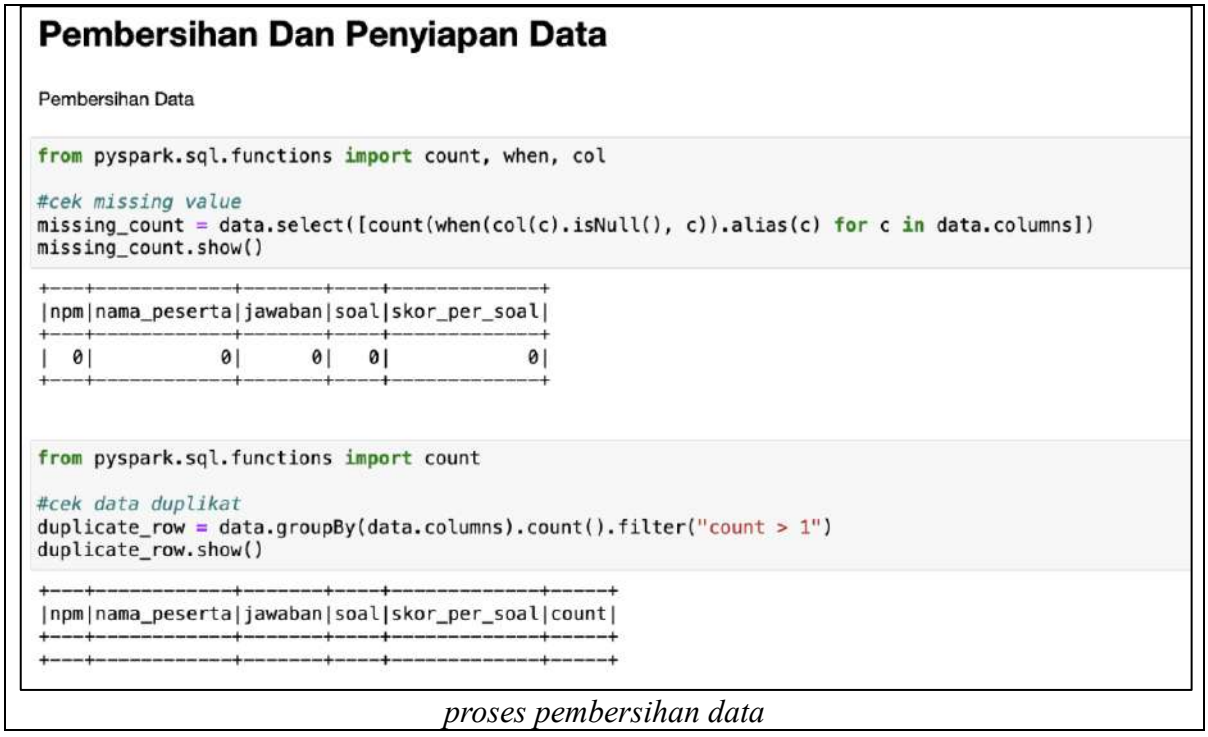
Pada kode *script* di atas, dilakukan proses membaca dan menampilkan dataset yang akan digunakan untuk membangun model sistem penskoran otomatis soal *essay*. Dimana untuk melakukan *load* data tersebut dibagi menjadi dua tahapan, yakni melakukan pembacaan dataset meenggunakan *library* pandas terlebih dahulu. Hal ini dikarenakan, pada PySpark tidak mendukung pembacaan *file* dengan format *.xlsx*.

Kemudian, dilakukan konversi menjadi *dataframe* PySpark, disimpan dalam variabel "data", dan ditampilkan 10 baris teratas. Dimana nantinya, data ini akan dipergunakan untuk membuat dan membangun model.

**3. Pembersihan Dan Penyiapan Data**

Merupakan proses untuk pengecekan *missing value* dan data *duplicate*. Dan, mengubah kolom "jawaban" yang memiliki tipe data *string* ke numerik untuk pembangunan model ALS.

**Pembersihan Data**




*proses pembersihan data*

Pada kode *script* di atas, dilakukan proses pembersihan data untuk mengecek *missing value* (nilai yang hilang) disemua kolom dataset dengan menggunakan *function select* dan *count* disetiap kolom dataset. Diperoleh hasil, bahwa tidak terdapat *missing value* disetiap kolom dalam dataset. Selanjutnya, dilakukan pemeriksaan duplikat dalam dataset menggunakan *function groupBy* dan *count* dengan *filtering "count > 1"* untuk menampilkan data duplikat.

Diperoleh hasil, bahwa tidak terdapat data duplikat dalam dataset yang disimpan dalam variabel “data”.

**Penyiapan Data**

Penyiapan Data

```
#memilih data yang ingin ditampilkan
#kolom "npm", "jawaban", "soal", dan mengubah "skor_per_soal" menjadi "true score"
data = data.select("npm", "jawaban", "soal", col("skor_per_soal").cast("Int").alias("true score"))

data.show(10)
```

```
+---+--------------------+----+----------+
|npm|             jawaban|soal|true score|
+---+--------------------+----+----------+
|  0|Tidak, Hanya memb...|   1|       100|
|  0|Biaya dihitung be...|   2|       100|
|  0|Hak cipta adalah ...|   3|       100|
|  0|Dijelaskan kepada...|   4|       100|
|  0|1. Melindungi dan...|   5|       100|
|  0|Ruang Komputer, P...|   6|       100|
|  0|Aturlah posisi pe...|   7|       100|
|  0|Posisi Kepala dan...|   8|       100|
|  0|1. Kecocokan soft...|   9|       100|
|  0|1. Fokus dan expo...|  10|       100|
+---+--------------------+----+----------+
only showing top 10 rows
```

*proses pengubahan nama kolom*

Pada kode *script* di atas, dilakukan pemilihan kolom tertentu dari dataset yang tersimpan dalam *dataframe* variabel data. Dengan memilih kolom “npm”, “jawaban”, “soal” dan mengubah “skor_per_soal” menjadi “true score”. Dimana “true score ini bertujuan untuk menjadi skor label aktual.

```
#pemrosesan data untuk mengubah "jawaban" menjadi vektor fitur
#agar dapat dipahami oleh model ALS
#tokenizing jawaban
from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover

tokenizer = Tokenizer(inputCol = "jawaban", outputCol = "feature1")
tokenized = tokenizer.transform(data)

tokenized.show(10)
```

```
+---+--------------------+----+----------+--------------------+
|npm|             jawaban|soal|true score|            feature1|
+---+--------------------+----+----------+--------------------+
|  0|Tidak, Hanya memb...|   1|       100|[tidak,, hanya, m...|
|  0|Biaya dihitung be...|   2|       100|[biaya, dihitung,...|
|  0|Hak cipta adalah ...|   3|       100|[hak, cipta, adal...|
|  0|Dijelaskan kepada...|   4|       100|[dijelaskan, kepa...|
|  0|1. Melindungi dan...|   5|       100|[1., melindungi, ...|
|  0|Ruang Komputer, P...|   6|       100|[ruang, komputer,...|
|  0|Aturlah posisi pe...|   7|       100|[aturlah, posisi,...|
|  0|Posisi Kepala dan...|   8|       100|[posisi, kepala, ...|
|  0|1. Kecocokan soft...|   9|       100|[1., kecocokan, s...|
|  0|1. Fokus dan expo...|  10|       100|[1., fokus, dan, ...|
+---+--------------------+----+----------+--------------------+
only showing top 10 rows
```

*proses pemrosesan data teks menjadi tokenizer*

Pada kode *script* di atas, dilakukan proses pengubahan kolom “jawaban” agar dapat dimengerti oleh model ALS. Dengan menggunakan modul HashingTF untuk mengubah kumpulan teks menjadi vector fitur dengan menggunakan fungsi hash untuk menghitung nilai hash dari setiap kata dalam teksi. Tokenizer untuk membagi teks menjadi token atau kata-kata individual untuk dianalisis lebih lanjut. StopWordRemover untuk menghapus kata-kata umum yang tidak informatif (*stopwords*).

Diperoleh hasil pengubahan teks jawaban menjadi token dan disimpan dalam kolom baru bernama “feature1” dan *dataframe* disimpan dalam variabel “tokenized”.

```
#menghapus kata2 yang tidak diinginkan dari teks
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

locale = sc._jvm.java.util.Locale
locale.setDefault(locale.forLanguageTag("en-US"))

swr = StopWordsRemover(inputCol=tokenizer.getOutputCol(), outputCol = "feature2", locale = "en")
tokenized = swr.transform(tokenized)

tokenized.show(10)
```

```
+---+--------------------+----+----------+--------------------+--------------------+
|npm|             jawaban|soal|true score|            feature1|            feature2|
+---+--------------------+----+----------+--------------------+--------------------+
|  0|Tidak, Hanya memb...|   1|       100|[tidak,, hanya, m...|[tidak,, hanya, m...|
|  0|Biaya dihitung be...|   2|       100|[biaya, dihitung,...|[biaya, dihitung,...|
|  0|Hak cipta adalah ...|   3|       100|[hak, cipta, adal...|[hak, cipta, adal...|
|  0|Dijelaskan kepada...|   4|       100|[dijelaskan, kepa...|[dijelaskan, kepa...|
|  0|1. Melindungi dan...|   5|       100|[1., melindungi, ...|[1., melindungi, ...|
|  0|Ruang Komputer, P...|   6|       100|[ruang, komputer,...|[ruang, komputer,...|
|  0|Aturlah posisi pe...|   7|       100|[aturlah, posisi,...|[aturlah, posisi,...|
|  0|Posisi Kepala dan...|   8|       100|[posisi, kepala, ...|[posisi, kepala, ...|
|  0|1. Kecocokan soft...|   9|       100|[1., kecocokan, s...|[1., kecocokan, s...|
|  0|1. Fokus dan expo...|  10|       100|[1., fokus, dan, ...|[1., fokus, dan, ...|
+---+--------------------+----+----------+--------------------+--------------------+
only showing top 10 rows
```

*poses penghapusan kata-kata yang tidak diinginkan*

Pada kode *script* di atas, dilakukan proses pembersihan teks dari kata-kata yang tidak di inginkan, yang merupakan pemrosesan bahasa alami (NLP). Bertujuan untuk membersihkan teks dari kata umum yang tidak relevan dan dapat meningkatkan kualitas analisis dan pemodelan yang dilakukan pada teks, seperti berfokus pada kata kunci dalam menilai *essay*, meningkatkan akurasi serta relevansi skor yang dihasilkan.

Dengan menggunakan objek *StopWordsRemover* untuk menghapus kata umum yang tidak diinginkan dan menyimoannya dalam kolom baru bernama “feature2”.

```
#mengubah feature jawaban menjadi vektor
from pyspark.ml.feature import VectorAssembler

#mengubah featire 2 menjadi angka
hashTF = HashingTF(inputCol=swr.getOutputCol(), outputCol = "feature3")
tokenized = hashTF.transform(tokenized)

tokenized.show(10)
```

```
+---+--------------------+----+----------+--------------------+--------------------+--------------------+
|npm|             jawaban|soal|true score|            feature1|            feature2|            feature3|
+---+--------------------+----+----------+--------------------+--------------------+--------------------+
|  0|Tidak, Hanya memb...|   1|       100|[tidak,, hanya, m...|[tidak,, hanya, m...|(262144,[22138,79...|
|  0|Biaya dihitung be...|   2|       100|[biaya, dihitung,...|[biaya, dihitung,...|(262144,[18111,56...|
|  0|Hak cipta adalah ...|   3|       100|[hak, cipta, adal...|[hak, cipta, adal...|(262144,[462,1515...|
|  0|Dijelaskan kepada...|   4|       100|[dijelaskan, kepa...|[dijelaskan, kepa...|(262144,[1532,686...|
|  0|1. Melindungi dan...|   5|       100|[1., melindungi, ...|[1., melindungi, ...|(262144,[10768,10...|
|  0|Ruang Komputer, P...|   6|       100|[ruang, komputer,...|[ruang, komputer,...|(262144,[78139,11...|
|  0|Aturlah posisi pe...|   7|       100|[aturlah, posisi,...|[aturlah, posisi,...|(262144,[1515,225...|
|  0|Posisi Kepala dan...|   8|       100|[posisi, kepala, ...|[posisi, kepala, ...|(262144,[1532,225...|
|  0|1. Kecocokan soft...|   9|       100|[1., kecocokan, s...|[1., kecocokan, s...|(262144,[6780,731...|
|  0|1. Fokus dan expo...|  10|       100|[1., fokus, dan, ...|[1., fokus, dan, ...|(262144,[2437,359...|
+---+--------------------+----+----------+--------------------+--------------------+--------------------+
only showing top 10 rows
```

*proses mengubah fitur jawaban menjadi vektor*

Pada Pada kode *script* di atas, dilakukan, pengubahan jawaban yang sebelumnya telah di *tokenizing* dan dibersihkan menjadi vektor fitur dengan menggunakan teknik *hasing term frequency* (HashingTF). Dilakukan inisiasi kolom *input* yakni "feature2" dengan kolom *output* "feature3" dengan menggunakan *function transform()*. Dan diperoleh hasil kolom baru bernama "feature3" yang berisi nilai vektor fitur dalam bentuk *double* (..,[..]...).

```
#mengubah feature jawaban yang telah ditokenized menjadi vektor tunggal dan memilih kolom yang ingin ditampilkan
from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType
from pyspark.ml.linalg import Vectors

#fungsi UDF untuk mengekstrak nilai tunggal dari vektor
extract_value = udf(lambda v: float(v[0]), DoubleType())

#membuat kolom baru dengan nilai tunggal dari vektor
tokenized = tokenized.withColumn("features", extract_value("feature3"))

selected_features = ["npm", "soal", "jawaban", "features", "true score"]
data = tokenized.select(selected_features)

data.show(10)
```

```
+---+----+--------------------+--------+----------+
|npm|soal|             jawaban|features|true score|
+---+----+--------------------+--------+----------+
|  0|   1|Tidak, Hanya memb...|     0.0|       100|
|  0|   2|Biaya dihitung be...|     0.0|       100|
|  0|   3|Hak cipta adalah ...|     0.0|       100|
|  0|   4|Dijelaskan kepada...|     0.0|       100|
|  0|   5|1. Melindungi dan...|     0.0|       100|
|  0|   6|Ruang Komputer, P...|     0.0|       100|
|  0|   7|Aturlah posisi pe...|     0.0|       100|
|  0|   8|Posisi Kepala dan...|     0.0|       100|
|  0|   9|1. Kecocokan soft...|     0.0|       100|
|  0|  10|1. Fokus dan expo...|     0.0|       100|
+---+----+--------------------+--------+----------+
only showing top 10 rows
```

*proses mengubah vektor double menjadi vektor tunggal*

Pada kode *script* di atas, dilakukan proses pengubahan jawaban yang telah diubah kedalam vektor dengan *output vektor double* menjadi vektor tunggal. Hal ini dikarenakan, untuk membangun dan membuat model ALS diperlukan kolom dengan numerik tunggal (angka tunggal). Dengan menggunakan udf (*user defined function*) untuk kustomisasi *datframe* yang berisi nilai tunggal dari vektor. DoubleType untuk menentukan tipe data *output*.

Dengan menggunakan kolom "feature3" sebagai *input* dan kolom *"features*" sebagai *output*. Kemudian, dipilih kolom yang akan ditampilkan saat pembangunan dan hasil model dan dinggap lebih relevan, yakni "npm", "soal", "jawaban", "features", dan "true score".

**4. Pembagian Data Training Dan Data Testing**

Merupakan proses untuk membagi dataset menjadi dua subset bagian, yakni data *training* dan data *testing* yang membantu untuk membangun dan mengevaluasi model ALS yang akan dibuat.

**Pembagian Data Training Dan Data Testing**

```
#membagi data dalam dataset menjadi data training 70% dan data testing 30%
split_data = data.randomSplit([0.7, 0.3])
training_data = split_data[0]
testing_data = split_data[1]

#jumlah data training dan data testing
training_row = training_data.count()
testing_row = testing_data.count()

print("Jumlah Data Training : ", training_row, " Dan Jumlah Data Testing : ", testing_row)
```

```
Jumlah Data Training :  84  Dan Jumlah Data Testing :  36
```

*proses pembagian data training dan data testing*

Pada kode *script* di atas, dilakukan proses untuk membagi dataset menjadi data *training* dengan proporsi 70% dan data *testing* dengan proporsi 30%. Dengan menggunakan *function randomSplit()*. Dengan jumlah total data sebesar 120 data, diperoleh data *training* sebesar 84 dan simpan dalam variabel "training_data" serta data *testing* sebesar 36 dan disimpan dalam variabel "testing_data".

```
#menampilkan data testing
training_data.show(84)
```

```
+----------+----+--------------------+--------+----------+
|       npm|soal|             jawaban|features|true score|
+----------+----+--------------------+--------+----------+
|         0|   1|Tidak, Hanya memb...|     0.0|       100|
|         0|   2|Biaya dihitung be...|     0.0|       100|
|         0|   4|Dijelaskan kepada...|     0.0|       100|
|         0|   5|1. Melindungi dan...|     0.0|       100|
|         0|   8|Posisi Kepala dan...|     0.0|       100|
|         0|   9|1. Kecocokan soft...|     0.0|       100|
|         0|  10|1. Fokus dan expo...|     0.0|       100|
|         0|  11|1. Peralatan yang...|     0.0|       100|
|1121020033|   1|tidak, cuma mengi...|     0.0|        52|
|1121020033|   2|biaya dihitung be...|     0.0|        42|
|1121020033|   3|hak membuat merup...|     0.0|        42|
```

*data training*

```
#menampilkan data training
testing_data.show(36)
```

```
+----------+----+--------------------+--------+----------+
|       npm|soal|             jawaban|features|true score|
+----------+----+--------------------+--------+----------+
|         0|   3|Hak cipta adalah ...|     0.0|       100|
|         0|   6|Ruang Komputer, P...|     0.0|       100|
|         0|   7|Aturlah posisi pe...|     0.0|       100|
|         0|  12|1. Dibuat grafik ...|     0.0|       100|
|1121020033|   7|aturlah posisi fi...|     0.0|        57|
|1121020033|   9|1.kesesuaian apli...|     0.0|        54|
|1121020033|  12|metode digital ar...|     0.0|        26|
|1220020029|   5|melindungi dan me...|     0.0|        74|
|1220020018|   6|ruang komputer, p...|     0.0|       100|
|1220020029|   9|kecocokan softwar...|     0.0|        84|
|1220020029|  12|dibuat grafik yan...|     0.0|        81|
|1220020018|  12|dibuat grafik yan...|     0.0|        86|
```

*data testing*

**5. Pembangunan Model Alternating Least Square (ALS)**

Merupakan proses pembangunan model *alternating least square* (ALS) dapat berperan dalam sistem penalian otomatis dengan mengoptimalkan proses penilaian jawaban *essay*. Dengan merekomendasikan jawaban *essay* berdasarkan kemiripan degan jawaban pada data *training.*

**Pembangunan Model ALS (Alternating Least Square)**

```
#membangun model ALS
from pyspark.ml.feature import StringIndexer
from pyspark.ml.recommendation import ALS

#karena semua kolom sudah bertipe numerik
#dan memenuhi syarat pembangunan model ALS, maka dibangun model ALS
als = ALS(maxIter = 10, regParam = 0.01, userCol = "soal", itemCol = "features",
          ratingCol = "true score")

#melatih model als
model = als.fit(training_data)
```

*proses pembangunan model ALS*

Pada kode *script* di atas, dilakukan proses pembangunan model ALS dengan *library* ALS. Dengan inisiasi jumlah iterasi sebesar 10, userCol “soal”, dan itemCol “features”. Kolom “features” merupakan kolom jawaban yang telah diubah menjadi vektor tunggal dengan menggunakan data *training* untuk membangun model. *Argumen* tersebut memiliki arti bahwa akan dilakukan prediksi skor jawaban *essay* berdasarkan kode soal.

Hal ini dikarenakan, setiap kode soal memiliki jawaban yang berbeda.

**6. Pengujian Model Berdasarkan Model Alternating Least Squre (ALS) Menggunakan Data Testing**

Merupakan proses untuk menerapkan model yang telah dihasilkan sebelumnya ke dalam data *testing*. Bertujuan untuk mengetahui hasil prediksi skor yang dihasilkan dari model tersebut ke dalam data *testing*.

**Pengujian Model Berdasarkan Model ALS (Alternating Least Square) Menggunakan Data Testing**

```
#lakukan prediksi dan penampilan pada data pengujian
from pyspark.sql.functions import round

predictions_testing_data = model.transform(testing_data)
predictions_testing_data = predictions_testing_data.withColumn("prediction",
                                                    round(predictions_testing_data["prediction"], 2))

predictions_testing_data.show(36)
```

*proses pengujian model ALS*

Pada kode *script* di atas, dilakukan proses memprediksi menggunakan model ALS yang telah dibuat dengan data *training* untuk dilatih menggunakan data *testing* sebesar 36 data. Dengan menggunakan *model.transform()* untuk melakukan prediksi pada data *testing* yang tersimpan dalam variabel "testing_data". Dan, hasil prediksi skornya disimpan dalam variabel "predictions_testing_test" dengan pembulatan dua angka dibelakang koma. Untuk hasil prediksi skor dapat direpresentasikan sebagai berikut.

| npm | soal | jawaban | features | true score | prediction |
|---|---|---|---|---|---|
| 0 | 12 | 1. Dibuat grafik ... | 0.0 | 100 | 71.33 |
| 0 | 6 | Ruang Komputer, P... | 0.0 | 100 | 100.0 |
| 0 | 3 | Hak cipta adalah ... | 0.0 | 100 | 78.2 |
| 0 | 7 | Aturlah posisi pe... | 0.0 | 100 | 88.0 |
| 1121020033 | 12 | metode digital ar... | 0.0 | 26 | 71.33 |
| 1220020029 | 5 | melindungi dan me... | 0.0 | 74 | 76.62 |
| 1121020033 | 9 | 1.kesesuaian apli... | 0.0 | 54 | 80.29 |
| 1121020033 | 7 | aturlah posisi fi... | 0.0 | 57 | 88.0 |
| 1220020029 | 12 | dibuat grafik yan... | 0.0 | 81 | 71.33 |
| 1220020018 | 6 | ruang komputer, p... | 0.0 | 100 | 100.0 |
| 1220020029 | 9 | kecocokan softwar... | 0.0 | 84 | 80.29 |
| 1220020018 | 12 | dibuat grafik yan... | 0.0 | 86 | 71.33 |
| 1220020023 | 12 | dibuat grafik yan... | 0.0 | 91 | 71.33 |
| 1220020023 | 1 | tidak, hanya memb... | 0.0 | 100 | 94.0 |
| 1220020023 | 3 | hak cipta adalah ... | 0.0 | 91 | 78.2 |
| 1120020017 | 12 | dibuat grafik yan... | 0.0 | 86 | 71.33 |
| 1120020017 | 1 | tidak, hanya memb... | 0.0 | 100 | 94.0 |
| 1120020017 | 6 | posisi tubuh, pos... | 0.0 | 90 | 100.0 |
| 1120020017 | 3 | emperbanyak cipta... | 0.0 | 47 | 78.2 |
| 1120020017 | 10 | fokus dan exposur... | 0.0 | 85 | 85.14 |
| 1120020017 | 2 | biaya dihitung be... | 0.0 | 84 | 83.56 |
| 1121020032 | 12 | dibuat grafik yan... | 0.0 | 86 | 71.33 |
| 1121020024 | 6 | ruang komputer, p... | 0.0 | 100 | 100.0 |
| 1121020032 | 6 | ruang komputer, p... | 0.0 | 100 | 100.0 |
| 1121020024 | 3 | hak cipta adalah ... | 0.0 | 83 | 78.2 |
| 1121020032 | 5 | melindungi dan me... | 0.0 | 74 | 76.62 |
| 1121020032 | 4 | dijelaskan kepada... | 0.0 | 72 | 68.12 |
| 1121020024 | 8 | posisi kepala dan... | 0.0 | 100 | 82.5 |
| 1121020036 | 8 | posisi kepala dan... | 0.0 | 82 | 82.5 |
| 1121020024 | 10 | fokus dan exposur... | 0.0 | 50 | 85.14 |
| 1121020035 | 6 | posisi tubuh | 0.0 | 67 | 100.0 |
| 1121020035 | 3 | hak cipta adalah ... | 0.0 | 83 | 78.2 |
| 1121020035 | 9 | kecocokan softwar... | 0.0 | 65 | 80.29 |
| 1121020035 | 4 | dijelaskan kepada... | 0.0 | 78 | 68.12 |
| 1121020035 | 7 | aturlah posisi pe... | 0.0 | 59 | 88.0 |
| 1121020035 | 10 | okus dan exposure... | 0.0 | 48 | 85.14 |

*hasil prediksi skor data testing*

Dari hasil prediksi tersebut, dapat dilihat bahwa terdapat beberapa kesamaan antara hasil skor prediksi dengan *true score* (skor aktual), namun juga terdapat beberapa perbedaan yang signifikan. Dengan contoh kesamaan, seperti pada baris ke-2, ke-10, ke-23, dan ke-24. Terdapat pula beberapa skor prediksi yang memiliki perbedaan signifikan dengan *true score*, seperti pada baris ke-5, ke-30, ke-31, dan ke-36. Untuk hasil yang lainnya, masih mirip dengan *true score*-nya.

Dalam melakukan analisis lebih lanjut, dapat dilakukan evaluasi kinerja model yang akan dilakukan pada proses akurasi model RMSE dan akurasi data.

**7. Akurasi Model**

Merupakan proses untuk mengetahui seberapa baik model ALS yang dihasilkan dalam memprediksi skor jawaban *essay* dengan menggunakan RMSE dan akurasi.

**Akurasi Model**

```
from pyspark.ml.evaluation import RegressionEvaluator

#perhitungan RMSE
evaluator = RegressionEvaluator(metricName = "rmse", labelCol = "true score", predictionCol = "prediction")
rmse = evaluator.evaluate(predictions_testing_data)

#menampilkan hasil
print("Root Mean Square Error (RMSE) : ", rmse)
```

```
Root Mean Square Error (RMSE) :  18.896240372702476
```

```
correct_predictions = predictions_testing_data.filter(predictions_testing_data["prediction"] ==
                                                      predictions_testing_data["true score"])

total_data = predictions_testing_data.count()
correct_count = correct_predictions.count()  #menghitung jumlah prediksi yang benar

accuracy = correct_count / total_data  #menghitung akurasi

print("Prediksi Benar : ", correct_count,
      "Total Data : ", total_data,
      "Akurasi : ", accuracy)
```

```
Prediksi Benar :  4 Total Data :  36 Akurasi :  0.1111111111111111
```

*proses perhitungan RMSE dan akurasi*

Pada kode script di atas, dilakukan dua proses, yakni perhitungan RMSE (Root Mean Square Error) dan akurasi. Untuk perhitungan RMSE dipergunakan RegressionEvaluator untuk mengevaluasi model dengan menentukan evaluasi RMSE dengan nilai sebenarnya "true score" dengan nilai prediksi "prediction" yang akan dievaluasi. Diperoleh nilai RMSE sebesar 18,8962 yang memiliki artian terdapat kesalahan rata-rata prediksi sekitar angka tersebut.

Dengan kata lain, prediksi model diharapkan memiliki kesalahan sekitar 18,8962 dalam memperkirakan skor jawaba soal *essay*. Untuk perhitungan akurasi dilakukan *filtering* dengan hanya prediksi yang sama dengan skor sebenarnya yang dipertahankan. Kemudian dilakukan

perhitungan akurasi dengan rumus *prediksi yang benar/total data testing*. Diperoleh hasil bahwa terdapat 4 prediksi yang benar dengan akurasi sekitar 0,1111.

Hal tersebut menujukkan adanya akurasi yang rendah. Sehingga, model perlu disempurnakan dan di sesuaikan lebih lanjut.

**8. Pengujian Model Berdasarkan Model Alternating Least Square (ALS) Menggunakan Data Uji Baru Dan Akurasi Modelnya**

Untuk memenuhi penugasan dengan melakukan pengujian menggunakan data uji soal baru dan mengetahui seberapa baik model dalam melakukan prediksi skor. Dilakukan pengujian model menggunakan data uji beru dengan langkah-langkah sebagai berikut.

**Data Uji Soal Baru**

```
import pandas as pd
import random

# Data uji baru yang ingin dibuat
data_uji = {
    "npm": ["1121020086"] * 12,  #semua npm adalah '06'
    "soal": list(range(1, 13)),  #kode soal dari 1 hingga 12
    "jawaban": [
        "Satu software sudah cukup karena sesuai dengan keahlian yang dibutuhkan.",
        "Perhitungan biaya didasarkan pada waktu pengerjaan dan tingkat kesulitan proyek.",
        "Hak cipta merupakan hak eksklusif bagi pencipta atau penerima hak untuk mengumumkan atau memperbanyak karya
        "Penting untuk menjelaskan kepada klien bahwa ukuran huruf yang terlalu besar atau terlalu kecil dapat memen
        "Langkah-langkah untuk melindungi dan menjaga keamanan di tempat kerja mencakup berbagai hal, seperti hak ke
        "Ruang komputer harus diatur sedemikian rupa agar nyaman digunakan, termasuk posisi tubuh dan posisi kompute
        "Untuk mengurangi ketegangan otot dan pikiran, penting untuk melakukan istirahat yang cukup dan gerakan tubu
        "Untuk memastikan kenyamanan saat bekerja, penting untuk menjaga posisi kepala, leher, punggung, pundak, len
        "Saat memilih perangkat lunak kamera, perlu dipertimbangkan kesesuaian dengan perangkat keras, resolusi kame
        "Fokus dan eksposur kamera digital harus diperhatikan agar gambar yang dihasilkan sesuai dengan kebutuhan, t
        "Saat mentransfer gambar, perlu diidentifikasi peralatan dan materi yang diperlukan, serta memastikan perala
        "Desain grafis harus memperhatikan prinsip desain visual, teknik digital artwork, dan penggunaan software ya
    ],
    "true score": [random.uniform(60, 100) for _ in range(12)]  # Skor acak antara 60 dan 100
}

#membuat DataFrame dari data uji
data_uji_baru = pd.DataFrame(data_uji)

# Menampilkan DataFrame
data_uji_baru
```

| | npm | soal | jawaban | true score |
|---|---|---|---|---|
| 0 | 1121020006 | 1 | Satu software sudah cukup karena sesuai dengan... | 70.752525 |
| 1 | 1121020006 | 2 | Perhitungan biaya didasarkan pada waktu penger... | 60.919572 |
| 2 | 1121020006 | 3 | Hak cipta merupakan hak eksklusif bagi pencipt... | 71.858525 |
| 3 | 1121020006 | 4 | Penting untuk menjelaskan kepada klien bahwa u... | 73.820298 |
| 4 | 1121020006 | 5 | Langkah-langkah untuk melindungi dan menjaga k... | 92.810806 |
| 5 | 1121020006 | 6 | Ruang komputer harus diatur sedemikian rupa ag... | 80.650687 |
| 6 | 1121020006 | 7 | Untuk mengurangi ketegangan otot dan pikiran, ... | 80.319357 |
| 7 | 1121020006 | 8 | Untuk memastikan kenyamanan saat bekerja, pent... | 86.776008 |
| 8 | 1121020006 | 9 | Saat memilih perangkat lunak kamera, perlu dip... | 97.714766 |
| 9 | 1121020006 | 10 | Fokus dan eksposur kamera digital harus diperh... | 69.376626 |
| 10 | 1121020006 | 11 | Saat mentransfer gambar, perlu diidentifikasi ... | 84.792481 |
| 11 | 1121020006 | 12 | Desain grafis harus memperhatikan prinsip desa... | 71.382934 |

*proses membuat data uji baru*

Dilakukan pembuatan data uji baru dengan npm “1121010200006” dengan kode soal “1-12” dan jawaban yang telah ditentukan, serta *true score* (skor aktual) di*random* dalam rentang 60 sampai dengan 100. Data tersebut disimpan dalam *dataframe* dengan variabel “data_uji_baru”.

```
from pyspark.ml.feature import Tokenizer, HashingTF, IDF
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType

#membuat sesi spark untuk mengubah data uji baru menjadi vektor
spark = SparkSession.builder \
    .appName("Data Uji Baru") \
    .getOrCreate()

#membuat DataFrame PySpark dari data uji
data_uji_spark = spark.createDataFrame(data_uji_baru)

#tokenisasi jawaban
tokenizer = Tokenizer(inputCol="jawaban", outputCol="words")
wordsData = tokenizer.transform(data_uji_spark)

#menggunakan HashingTF untuk mengubah kata-kata menjadi fitur vektor
hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=10000)
featurizedData = hashingTF.transform(wordsData)

#menghitung IDF untuk mendapatkan vektor akhir
idf = IDF(inputCol="rawFeatures", outputCol="jawaban_vektor")
idfModel = idf.fit(featurizedData)
data_vectorized = idfModel.transform(featurizedData)

#pemilihan kolom yang diperlukan untuk model ALS
data_selected = data_vectorized.select("npm", "soal", "jawaban", "true score", "jawaban_vektor")

#fungsi UDF untuk mengekstrak nilai tunggal dari vektor
extract_value = udf(lambda v: float(v[0]), DoubleType())

#membuat kolom baru dengan nilai tunggal dari vektor
data_selected = data_selected.withColumn("features", extract_value("jawaban_vektor"))

#menampilkan DataFrame yang sudah diubah ke dalam vektor
data_final = data_selected.select("npm", "soal", "jawaban", "features", "true score")
data_final.show()
```

```
+----------+----+--------------------+--------+------------------+
|       npm|soal|             jawaban|features|       true score|
+----------+----+--------------------+--------+------------------+
|1121020006|   1|Satu software sud...|     0.0| 70.7525245899488|
|1121020006|   2|Perhitungan biaya...|     0.0|60.919571857939 61|
|1121020006|   3|Hak cipta merupak...|     0.0|71.85852543131415|
|1121020006|   4|Penting untuk men...|     0.0|73.82029818135985|
|1121020006|   5|Langkah-langkah u...|     0.0| 92.8108063809619|
|1121020006|   6|Ruang komputer ha...|     0.0|80.65068661981215|
|1121020006|   7|Untuk mengurangi ...|     0.0|80.31935657019821|
|1121020006|   8|Untuk memastikan ...|     0.0|86.77600833946627|
|1121020006|   9|Saat memilih pera...|     0.0|97.71476565028138|
|1121020006|  10|Fokus dan eksposu...|     0.0|69.37662586918414|
|1121020006|  11|Saat mentransfer ...|     0.0|84.79248130763396|
|1121020006|  12|Desain grafis har...|     0.0|71.39293374117395|
+----------+----+--------------------+--------+------------------+
```

*proses pengubahan jawaban menjadi vector tunggal*

Kemudian dilakukan perubahan kolom “jawaban”dengan tipe *string* pada data uji baru tersebut menjadi vektor fitur tunggal yang dapat digunakan dalam model ALS. Menggunakan langkah seperti sebelumnya, yakni tokenisasi jawaban, penggunaan HashingTF untuk mengonversi kata-kata menjadi fitur vektor, perhitungan IDF untuk mendapatkan vektor akhir, dan pemilihan kolom “npm”, “soal”, jawaban”, “features”, dan “true score” yang akan ditampilkan.

```
#lakukan prediksi dan penampilan pada data pengujian
from pyspark.sql.functions import round

predictions_testing_data_new = model.transform(data_final)
predictions_testing_data_new = predictions_testing_data_new.withColumn("prediction", round(predictions_testing_data_
                                                                                            ["prediction"], 2))

predictions_testing_data_new.show()
```

```
+----------+----+--------------------+--------+------------------+----------+
|       npm|soal|             jawaban|features|       true score|prediction|
+----------+----+--------------------+--------+------------------+----------+
|1121020006|   1|Satu software sud...|     0.0| 70.7525245899488|      94.0|
|1121020006|   3|Hak cipta merupak...|     0.0|71.85852543131415|      78.2|
|1121020006|   2|Perhitungan biaya...|     0.0|60.91957185793961|     83.56|
|1121020006|   4|Penting untuk men...|     0.0|73.82029818135986|     68.12|
|1121020006|   6|Ruang komputer ha...|     0.0|80.65068661981215|     100.0|
|1121020006|   5|Langkah-langkah u...|     0.0| 92.8108063809619|     76.62|
|1121020006|   7|Untuk mengurangi ...|     0.0|80.31935657019821|      88.0|
|1121020006|   9|Saat memilih pera...|     0.0|97.71476565028138|     80.29|
|1121020006|   8|Untuk memastikan ...|     0.0|86.77600833946627|      82.5|
|1121020006|  10|Fokus dan eksposu...|     0.0|69.37662586918414|     85.14|
|1121020006|  12|Desain grafis har...|     0.0|71.39293374117395|     71.33|
|1121020006|  11|Saat mentransfer ...|     0.0|84.79248130763396|      82.9|
+----------+----+--------------------+--------+------------------+----------+
```

*proses pembuatan hasil skor prediksi*

Selanjutnya, dilakukan proses penggunaan model yang telah dibuat sebelumnya (pada tahap pembangunan model ALS) dengan output pembulatan 2 angka dibelakang koma. Diperoleh hasil prediksi skor jawaban *essay* masih terdapat beberapa perbedaan yang signifikan antara "true score" dan prediction.

**Akurasi Model Data Uji Soal Baru**

```
from pyspark.ml.evaluation import RegressionEvaluator

#perhitungan RMSE
evaluator = RegressionEvaluator(metricName = "rmse", labelCol = "true score", predictionCol = "prediction")
rmse = evaluator.evaluate(predictions_testing_data_new)

#menampilkan hasil
print("Root Mean Square Error (RMSE) : ", rmse)
```

```
Root Mean Square Error (RMSE) :  14.128128002543683
```

```
correct_predictions = predictions_testing_data_new.filter(predictions_testing_data_new["prediction"] ==
                                                          predictions_testing_data_new["true score"])

total_data = predictions_testing_data_new.count()
correct_count = correct_predictions.count()  #menghitung jumlah prediksi yang benar

accuracy = correct_count / total_data  #menghitung akurasi

print("Prediksi Benar : ", correct_count,
      "Total Data : ", total_data,
      "Akurasi : ", accuracy)
```

```
Prediksi Benar :  0 Total Data :  12 Akurasi :  0.0
```

*proses perhitungan akurasi model*

Dilakukan perhitungan RMSE dan akurasi model dari hasil prediksi data uji baru. Diperoleh nilai RMSE sekitar 14,1281 dan nilai prediksi benar sebesar 0 dengan akurasi 0,0. Dengan akurasi tersebut, dapat diartikan bahwa tidak terdapat prediksi yang benar dari model terhadap data uji yang diberikan.

**Automated Scoring Sytem Menggunakan Hash**

Hash merupakan fungsi matematis yang mengambil *input* data dan menghasilkan *output* untuk berukuran tetap yang disebut nilai hash. Pada pembuatan sistem otomatis penskoran jawaban *essay*, Hash dipergunakan untuk mengubah kolom "jawaban" dengan tipe *string* menjadi numerik dalam bentuk bilangkan bulat ataupun heksadesimel. Proses Hash dapat membantu mengubah data teks atau *string* dalam kolom "jawaban" *essay* menjadi representasi numerik yang lebih sederhana dan mudah untuk diproses. Mengenai pembuata sistem otomatis penskoran menggunakan Hash akan dijelaskan lebih lanjut sebagai berikut.

**Automated Scoring System Menggunakan PySpark Dengan Hash**

```
from pyspark.sql import SparkSession
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.sql.functions import hash
import pandas as pd

#inisialisasi SparkSession
spark = SparkSession.builder \
    .appName("Automated Scoring System ALS (Hash)") \
    .getOrCreate()

#baca file excel menggunakan pandas dan buat dataframe Spark
data = spark.createDataFrame(pd.read_excel("training_data_essay.xlsx"))

#ubah jawaban menjadi vektor hash
hashed_data = data.withColumn("vektor hash", hash("jawaban")).select("npm", "soal", "vektor hash",
                                                                      col("skor_per_soal").alias("true score"))

#membagi data menjadi data latih dan data uji
(training_data, testing_data) = hashed_data.randomSplit([0.7, 0.3])
print("Jumlah Data Training:", training_data.count())
print("Jumlah Data Training:", testing_data.count())

#membuat dan membangun model ALS
model = ALS(maxIter=10, regParam=0.1, userCol="soal", itemCol="vektor hash", ratingCol="true score").fit(
    training_data)

#memprediksi skor pada data uji
predictions = model.transform(testing_data)

#evaluasi kinerja model dengan RMSE
rmse = RegressionEvaluator(metricName="rmse", labelCol="true score", predictionCol="prediction").evaluate(
    predictions)

#menampilkan hasil hash
hashed_data.select("npm", "soal", "vektor hash", "true score").show(truncate=False)
```

*proses pembuatan session, penggunaan hash, dan pembuatan model*

Pada kode *script* di atas, dilakukan proses implementasi sistem rekomendasi menggunakan metode *alternating least square* (ALS) dalam lingkungan Spark menggunakan PySpark. Dengan menggunakan langkah-langkah, melakukan inisialisasi *spark session* dengan nama "Automated Scoring System ALS (Hash)". Lalu dilakukan, *load* dataset bernama "training_data_essay.xlsx" yang diubah dalam variabel data.

Lalu, dilakukan transformasi data dengan mengubah kolom "jawaban" menjadi vektor atau nilai hash dengan menggunakan *function hash()* untuk mengonversi nilai dalam kolom "jawaban" menjadi representasi nilai hash. Kemudian dilakukan pengubahan nama kolom "skor_per_soal" menjadi "true score" dan memilih kolom yang ingin ditampilkan untuk membentuk *dataframe* baru yang disimpan dalam variabel "hashed_data".

Dilakukan pembagian data dengan menggunakan *randomSplit()* dengan proporsi 70% data *training* sebesar 84 dan data *testing* sebesar 86. Kemudian dilakukan pembuatan model ALS dengan jumlah iterasi maksimum sebesar 10 menggunakan "data_training". Bertujuan untuk memprediksi skor jawaban *essay* pada data uji. Lalu, dilakukan evaluasi kinerja model menggunakan RMSE. Dengan data hasil transformasi dapat direpresentasikan sebagai berikut.

```
Jumlah Data Training: 84
Jumlah Data Training: 36
+-----------+----+-----------+----------+
|npm        |soal|vektor hash|true score|
+-----------+----+-----------+----------+
|0          |1   |-2059296905|100.0     |
|0          |2   |1183180174 |100.0     |
|0          |3   |1232762403 |100.0     |
|0          |4   |-2035408785|100.0     |
|0          |5   |1588395990 |100.0     |
|0          |6   |339970513  |100.0     |
|0          |7   |50850002   |100.0     |
|0          |8   |-945877996 |100.0     |
|0          |9   |1576366224 |100.0     |
|0          |10  |-1905649442|100.0     |
|0          |11  |550139146  |100.0     |
|0          |12  |1727767227 |100.0     |
|1121020033 |1   |1947733435 |52.7      |
|1121020033 |2   |-1139863335|42.86     |
|1121020033 |3   |122676417  |42.16     |
|1121020033 |4   |-1054163002|27.19     |
|1121020033 |5   |1990940339 |44.14     |
|1121020033 |6   |1770907636 |100.0     |
|1121020033 |7   |-463479969 |57.68     |
|1121020033 |8   |-412537011 |45.71     |
+-----------+----+-----------+----------+
only showing top 20 rows
```

*proses penampilan jumlah data testing, data traning, dan pengubahan jawaban menjadi vektor hash*

Kemudian, untuk prediksi skor data *testing* menggunakan model yang telah dibuat dengan format dua angka dibelakang koma dapat direpresentasikan sebagai berikut.

```
#menampilkan hasil prediksi pada data testing
from pyspark.sql.functions import hash, col, format_number

predictions.select("npm", "soal", "vektor hash", "true score",
                   format_number("prediction", 2).alias("prediction")).show(36)
```

```
+----------+----+-----------+----------+----------+
|       npm|soal|vektor hash|true score|prediction|
+----------+----+-----------+----------+----------+
|         0|   1|-2059296905|     100.0|       NaN|
|         0|   5| 1588395990|     100.0|       NaN|
|         0|   9| 1576366224|     100.0|       NaN|
|         0|   8| -945877996|     100.0|       NaN|
|         0|   7|   50850002|     100.0|       NaN|
|         0|  11|  550139146|     100.0|       NaN|
|         0|   2| 1183180174|     100.0|       NaN|
|1121020033|   2|-1139863335|     42.86|       NaN|
|1121020033|   6| 1770907636|     100.0|    100.00|
|1220020029|   6| 1770907636|     100.0|    100.00|
|1220020029|   3|  770340049|     91.71|     91.71|
|1121020033|   5| 1990940339|     44.14|       NaN|
|1220020018|   5|-1932865057|     74.73|     74.73|
|1220020018|   9|-1019902387|     99.31|       NaN|
|1220020029|   9|-1092404005|     84.88|     84.87|
|1220020029|  10|-1672280317|     100.0|       NaN|
|1220020029|  11| -271105210|     77.63|       NaN|
|1220020018|   2| 1176853507|     100.0|    100.00|
|1220020023|  12|-1335018086|     91.17|       NaN|
|1220020023|   3|  770340049|     91.71|     91.71|
|1220020023|   5|  -69276039|     90.27|       NaN|
|1220020023|   9|  447829639|     65.89|     65.89|
|1220020023|   7|-1392782412|     86.22|     86.21|
|1220020023|  10|-1208312224|     90.14|       NaN|
|1220020023|   2| 1176853507|     100.0|    100.00|
|1120020017|   8| 1044171719|     89.17|     89.17|
|1120020017|  10| 1988679646|     85.75|       NaN|
|1120020017|  11|-1838887780|     80.09|     80.09|
|1121020024|  12| 1086045397|     64.29|       NaN|
|1121020036|   1| -256638840|     100.0|    100.00|
|1121020036|   9|-1083923656|     84.88|       NaN|
|1121020024|   7| 1123192925|     100.0|       NaN|
|1121020024|  11| 1713927558|     98.35|       NaN|
|1121020036|  12| -902409772|     86.53|     86.53|
|1121020035|   1| -256638840|     100.0|    100.00|
|1121020035|  11|-1994407618|     78.41|       NaN|
+----------+----+-----------+----------+----------+
```

*proses penampilan jumlah prediksi*

Dapat diperlihatkan bahawa terdapat beberapa baris di kolom “prediction” yang memiliki nilai “nan”. Hal ini dikarenakan kesulitan dalam melakukan ekstrapolasi dari data latih ke data uji dan keterbatasan model dalam menangani pola yang ada dalam data uji. Sehingga, akan ditampilkan RMSE bernilai *nan*.

```
#akurasi data
#rmse
print("Root Mean Squared Error (RMSE) pada data uji:", rmse)
```

```
Root Mean Squared Error (RMSE) pada data uji: nan
```

*proses penampilan rmse*

Untuk memeroleh nilai RMSE dilakukan penghapusan baris yang memiliki nilai *nan* pada kolom “prediction”. Untuk menjaga konsistensi data, mendukung analisis yang lebih akurat, dan menghindari bias. Dengan menggunakan *argument na.drop()* ditampilkan hasil sebagai berikut.

```
#menghapus baris yang memiliki nilai prediksi nan agar bisa dihitung rmse
predictions_without_nan = predictions.na.drop(subset=["prediction"])
predictions_without_nan.select("npm", "soal", "vektor hash", "true score",
                               format_number("prediction", 2).alias("prediction")).show()
```

```
+----------+----+-----------+----------+----------+
|       npm|soal|vektor hash|true score|prediction|
+----------+----+-----------+----------+----------+
|1121020033|   6| 1770907636|     100.0|    100.00|
|1220020029|   6| 1770907636|     100.0|    100.00|
|1220020029|   3|  770340049|     91.71|     91.71|
|1220020018|   5|-1932865057|     74.73|     74.73|
|1220020029|   9|-1092404005|     84.88|     84.87|
|1220020018|   2| 1176853507|     100.0|    100.00|
|1220020023|   3|  770340049|     91.71|     91.71|
|1220020023|   9|  447829639|     65.89|     65.89|
|1220020023|   7|-1392782412|     86.22|     86.21|
|1220020023|   2| 1176853507|     100.0|    100.00|
|1120020017|   8| 1044171719|     89.17|     89.17|
|1120020017|  11|-1838887780|     80.09|     80.09|
|1121020036|   1| -256638840|     100.0|    100.00|
|1121020036|  12| -902409772|     86.53|     86.53|
|1121020035|   1| -256638840|     100.0|    100.00|
+----------+----+-----------+----------+----------+
```

```
#rmse
evaluator = RegressionEvaluator(metricName="rmse", labelCol="true score", predictionCol="prediction")
rmse = evaluator.evaluate(predictions_without_nan)

print("Root Mean Squared Error (RMSE) Setelah Menghapus NaN : ", rmse)
```

```
Root Mean Squared Error (RMSE) Setelah Menghapus NaN :  0.0039636533191873425
```

*proses penampilan hasil prediksi setelah nilai nan dihapus dan rmse*

Setelah dilakukan penghapusan baris yang memiliki nilai *nan* ditampilkan bahwa hanya terdapat 15 baris yang memiliki nilai lengkap. Awalnya, terdapat 36 data *testing*, namun kurang lebih terdapat 11 baris prediksi skor yang memiliki nilai *nan*. Dan, diperoleh nilai RMSE sebesar

0,004 yang berarti model memiliki tingkat kesalahan yang sangat kecil dalam memprediksi nilai sebenarnya dari data *testing*.

**Kesimpulan Automate Scoring Sytem Menggunakan Tokenizing Dan Hash :**

**Automated Scoring System Menggunakan Tokenizing (ALS)**

Dilakukan pembuatan model menggunakan *tokenizing* untuk mengubah teks jawaban menjadi vektor fitur yang kemudian akan digunakan dalam model ALS. Diperoleh hasil pengujian adanya perbedaan signifikan antara prediksi skor dan skor aktual, serta akurasi yang rendah. Memiliki kelebihan mampu memroses teks jawaban dengan lebih detail dengan mempertimbangkan kata-kata individual, memberikan representasi vektor fitur yang lebih informatif.

Dengan kekurangan adanya proses *tokenizing* yang kompleks dan model cenderung sensitif terhadap kata kunci dalam jawaban.

**Automated Scoring System Menggunakan Hash (ALS)**

Dilakukan pembuatan model menggunakan Hash untuk mengubah teks jawaban menjadi nilai numerik yang kemudian akan digunakan dalam model ALS. Diperoleh hasil pengujian adanya nilai RMSE yang sangat kecil dan model yang memiliki tingkat kesalahan sangat rendah dalam meprediksi skor jawaban *essay*. Memiliki kelebihan adanya proses yang cepat dan sederhana. Dengan kekurangan adanya representasi fitur yang kurang informatif dibanding *tokenizing*.

Pendekatan Hash lebih direkomendasikan dalam meprediksi skor dibanding Tokenizing dibuktikan dengan nilai RMSE yang rendah. Namun, perlu diingat untuk memeriksa konsistensi hasil prediksi.