LAPORAN

RENCANA TUGAS MANDIRI (RTM) Ke-5

MATA KULIAH BIG DATA

"MEMBUAT AUTOMATED SCORING SYSTEM MENGGUNAKAN PYSPARK"

DISUSUN OLEH:

Fadlila Agustina (21083010050)

DOSEN PENGAMPU:

Kartika Maulida Hindrayani, S.Kom., M.Kom. (NIP. 199209092022032009)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2023

A. Deskripsi Tugas

Tugas ini bertujuan untuk membuat kode *script PySpark* untuk melakukan sistem penskoran secara otomatis pada soal *essay*. Dataset yang digunakan sebagai data latih diambil dari hasil ujian siswa/mahasiswa dengan sejumlah soal tertentu. Algoritma yang digunakan untuk prediksi menggunakan Alternating Least Square. *Output* dari program berupa akurasi dan pengujian menggunakan data uji soal baru. Mahasiswa juga diminta untuk mendokumentasikan langkah demi langkah prosesnya

B. Penjelasan Dataset

Dataset grading esai adalah kumpulan data yang digunakan untuk melatih dan menguji model pembelajaran mesin dalam melakukan penilaian otomatis terhadap jawaban esai yang dibuat oleh siswa. Dataset ini terdiri dari beberapa atribut, seperti:

- NPM (Nomor Pokok Mahasiswa) : Kode unik yang diberikan kepada setiap mahasiswa di sebuah perguruan tinggi.
- Nama Peserta : Nama mahasiswa yang menulis jawaban esai.
- Jawaban : Teks jawaban esai yang ditulis oleh mahasiswa.
- Skor per soal : Skor yang diberikan pada jawaban esai untuk setiap soal yang diberikan.

Dataset essay grading dapat digunakan sebagai pelatihan bagi model pembelajaran mesin untuk melakukan penilaian otomatis terhadap jawaban esai. Fitur-fitur pada dataset, seperti teks jawaban dan skor per soal, digunakan dalam proses pelatihan model agar dapat memprediksi skor yang seharusnya diberikan pada sebuah jawaban esai berdasarkan teks yang ditulis. Dataset ini tersedia secara publik dan sering digunakan oleh para peneliti atau praktisi di bidang pemrosesan bahasa alami (*Natural Language Processing*) dan pembelajaran mesin untuk mengembangkan model penilaian otomatis yang lebih baik.

C. Automated Scoring System

Sistem penilaian otomatis adalah sebuah program komputer yang melakukan evaluasi secara otomatis. Sistem ini biasanya digunakan dalam bidang pendidikan untuk menilai jawaban tes siswa atau keterampilan dalam suatu bidang. Penilaian otomatis sering digunakan dalam menilai jawaban tes siswa atau karya tulis seperti esai, makalah, atau tugas lainnya. Sistem ini menggunakan algoritma yang telah diprogram sebelumnya untuk mengevaluasi karya siswa berdasarkan kriteria tertentu, seperti tata bahasa, struktur, isi, dan kejelasan argumen.

D. Langkah-langkah

```
[1] pip install pyspark

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting pyspark
      Downloading pyspark-3.4.0.tar.gz (310.8 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 310.8/310.8 MB 2.9 MB/s eta 0:00:00
      Preparing metadata (setup.py) ... done
    Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
    Building wheels for collected packages: pyspark
      Building wheel for pyspark (setup.py) ... done
      Created wheel for pyspark: filename=pyspark-3.4.0-py2.py3-none-any.whl size=311317130 sha256=453fd2655a937d0ae0372cd056f3e67524d3f5cf95b20caa04e204a
      Stored in directory: /root/.cache/pip/wheels/7b/1b/4b/3363a1d04368e7ff0d408e57ff57966fcdf00583774e761327
    Successfully built pyspark
    Installing collected packages: pyspark
    Successfully installed pyspark-3.4.0
```

Perintah "**pip install pyspark**" adalah perintah yang digunakan untuk menginstal paket PySpark menggunakan pip, yang merupakan manajer paket Python. PySpark adalah modul

Python yang digunakan untuk mengakses dan mengolah data menggunakan Apache Spark, sebuah platform pemrosesan data terdistribusi.

Setelah menjalankan perintah "**pip install pyspark**", pip akan mengunduh dan menginstal paket PySpark serta dependensinya. Setelah instalasi selesai, Anda akan dapat menggunakan PySpark dalam kode Python Anda.

PySpark memungkinkan Anda untuk memproses data secara distribusi menggunakan paradigma pemrograman terdistribusi. Ini memungkinkan Anda untuk memanfaatkan kecepatan dan kapasitas pemrosesan yang tinggi, sehingga cocok untuk analisis data besar atau tugas pemrosesan data yang kompleks.

Setelah menginstal PySpark, Anda dapat mengimpor modulnya dalam kode Python Anda menggunakan pernyataan import pyspark. Dengan demikian, Anda dapat memanfaatkan fungsionalitas PySpark untuk melakukan pemrosesan data terdistribusi menggunakan Apache Spark.

```
[40] # Import modul
     from pyspark.sql import SparkSession
     from pyspark.sql.types import *
     from pyspark.sql.functions import *
     from pyspark.ml.classification import LogisticRegression
     from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover

     # Buat session
     appName = "Automated scoring system menggunakan PySpark"
     spark = SparkSession \
       .builder \
       .appName(appName) \
       .config("spark.some.config.option", "some-value") \
       .getOrCreate()
```

Selanjutnya, kode membuat sesi Spark dengan nama aplikasi "**Automated scoring system menggunakan PySpark**" menggunakan SparkSession. Sesi Spark ini digunakan sebagai titik masuk untuk berinteraksi dengan Apache Spark.

Setelah sesi Spark dibuat, Anda dapat menggunakan modul dan fungsi-fungsi PySpark tersebut untuk memanipulasi dan menganalisis data, serta mengembangkan model pembelajaran mesin. Beberapa modul yang diimpor, seperti HashingTF, Tokenizer, dan StopWordsRemover, menyediakan fitur-fitur pemrosesan teks yang berguna dalam analisis teks dan pemrosesan bahasa alami. Sedangkan LogisticRegression adalah salah satu algoritma klasifikasi yang dapat digunakan dalam pengembangan sistem penilaian otomatis.

```
[41] import os
     os.environ["PYSPARK_SUBMIT_ARGS"] = "--master local[*] pyspark-shell"
```

Selanjutnya, **os.environ**["PYSPARK_SUBMIT_ARGS"] = "--master local[*] pyspark-shell" adalah perintah yang digunakan untuk mengatur variabel lingkungan PYSPARK_SUBMIT_ARGS pada objek **os.environ**. Nilai yang ditetapkan, yaitu "--master local[*] pyspark-shell", digunakan untuk mengkonfigurasi Apache Spark dengan mode lokal. Argumen --master local[*] menunjukkan bahwa Spark akan berjalan pada mode lokal dengan menggunakan semua core yang tersedia pada mesin saat ini. Sementara itu, pyspark-shell menunjukkan bahwa lingkungan Spark shell PySpark akan digunakan.

Dengan mengatur variabel lingkungan ini, saat kita menjalankan kode PySpark, sesi Spark akan berjalan dalam mode lokal dengan menggunakan semua core yang tersedia pada mesin saat ini, dan lingkungan Spark shell PySpark akan aktif.

```
[42] import pandas as pd
```

Pandas adalah sebuah pustaka Python yang populer digunakan untuk manipulasi dan analisis data. Dengan menggunakan pandas, kita dapat memanipulasi data dalam bentuk tabel (DataFrame) dan melakukan berbagai operasi seperti pemfilteran, penggabungan, pengindeksan, dan perhitungan statistik.

```
[43] essay = spark.read.csv("/content/training_data_essay.csv", inferSchema=True, header=True)
     essay.show(truncate=False, n=5 )

+---+------------+--------------------------------------------------------------------------------------------------------------
|npm|nama_peserta|jawaban
+---+------------+--------------------------------------------------------------------------------------------------------------
|0  |Admin       |Tidak, Hanya membutuhkan satu karena satu software sesuai dengan keahlian
|0  |Admin       |Biaya dihitung berdasarkan waktu pengerjaan dan tingkat kesulitan
|0  |Admin       |Hak cipta adalah hak eksklusif bagi pencipta atau penerima hak untuk mengumumkan atau memperbanyak ciptaannya atau memberi izin untuk
|0  |Admin       |Dijelaskan kepada klien jika huruf terlalu besar maka kita tidak dapat memasukkan cukup banyak kata pada setiap baris agar pembaca le
|0  |Admin       |1. Melindungi dan menjamin pekerja atas hak keselamatannya 2. Memlihara sumber produksi agar dapat digunakan secara aman dan efisien
+---+------------+--------------------------------------------------------------------------------------------------------------
only showing top 5 rows
```

Pada kode di atas, kita menggunakan **spark.read.csv()** untuk membaca file CSV dengan nama training_data_essay.csv. Parameter inferSchema=True digunakan untuk menginfer skema data secara otomatis berdasarkan isi file CSV. Sedangkan **header=True** menandakan bahwa file CSV memiliki baris header yang berisi nama kolom.

Setelah membaca file CSV, kita menggunakan metode **show()** pada objek DataFrame essay untuk menampilkan 5 baris pertama dari data tersebut. Parameter **truncate=False** digunakan agar seluruh konten dari setiap kolom dapat ditampilkan secara penuh.

```
[44] #pilih data hanya dari kolom "SentimentText" dan kolom "Sentiment".
     #pilih nilai di kolom "Sentiment" ke tipe integer dan mengganti nya dengan "label"
     data = essay.select('npm', 'jawaban', 'soal', 'skor_per_soal')
     data.show(5)
```

```
+---+--------------------+----+-------------+
|npm|             jawaban|soal|skor_per_soal|
+---+--------------------+----+-------------+
|  0|Tidak, Hanya memb...|   1|        100.0|
|  0|Biaya dihitung be...|   2|        100.0|
|  0|Hak cipta adalah ...|   3|        100.0|
|  0|Dijelaskan kepada...|   4|        100.0|
|  0|1. Melindungi dan...|   5|        100.0|
+---+--------------------+----+-------------+
only showing top 5 rows
```

Dalam kode di atas, kita menggunakan metode **select()** pada objek DataFrame essay untuk memilih kolom-kolom yang ingin ditampilkan. Kita memilih kolom "npm", "jawaban", "soal", dan "skor_per_soal" dengan menyertakan nama kolom tersebut sebagai argumen dalam metode **select()**.

Setelah memilih kolom yang diinginkan, kita menggunakan metode **show(5)** untuk menampilkan 5 baris pertama dari data yang telah dipilih.

```
[45] from pyspark.sql.functions import hash, abs
     data = essay.withColumn("HashValue", hash("jawaban"))
     data.show()
```

```
+----------+------------+--------------------+----+------------+-----------+
|       npm|nama_peserta|             jawaban|soal|skor_per_soal|  HashValue|
+----------+------------+--------------------+----+------------+-----------+
|         0|       Admin|Tidak, Hanya memb...|   1|       100.0|-2059296905|
|         0|       Admin|Biaya dihitung be...|   2|       100.0| 1183180174|
|         0|       Admin|Hak cipta adalah ...|   3|       100.0| 1232762403|
|         0|       Admin|Dijelaskan kepada...|   4|       100.0|-2035408785|
|         0|       Admin|1. Melindungi dan...|   5|       100.0| 1588395990|
|         0|       Admin|Ruang Komputer, P...|   6|       100.0|  339970513|
|         0|       Admin|Aturlah posisi pe...|   7|       100.0|   50850002|
|         0|       Admin|Posisi Kepala dan...|   8|       100.0| -945877996|
|         0|       Admin|1. Kecocokan soft...|   9|       100.0| 1576366224|
|         0|       Admin|1. Fokus dan expo...|  10|       100.0|-1905649442|
|         0|       Admin|1. Peralatan yang...|  11|       100.0|  550139146|
|         0|       Admin|1. Dibuat grafik ...|  12|       100.0| 1727767227|
|1121020033|          AP|tidak, cuma mengi...|   1|        52.7| 1947733435|
|1121020033|          AP|biaya dihitung be...|   2|       42.86|-1139863335|
|1121020033|          AP|hak membuat merup...|   3|       42.16|  122676417|
|1121020033|          AP|dipaparkan pada k...|   4|       27.19|-1054163002|
|1121020033|          AP|1. mencegah serta...|   5|       44.14| 1990940339|
|1121020033|          AP|ruang komputer, p...|   6|       100.0| 1770907636|
|1121020033|          AP|aturlah posisi fi...|   7|       57.68| -463479969|
|1121020033|          AP|posisi kepala ser...|   8|       45.71| -412537011|
+----------+------------+--------------------+----+------------+-----------+
only showing top 20 rows
```

Dalam kode di atas, kita menggunakan from **pyspark.sql.functions** import hash, abs untuk mengimpor fungsi hash dan abs dari modul **pyspark.sql.functions**.

Selanjutnya, kita menggunakan metode **withColumn()** pada objek DataFrame essay untuk menambahkan kolom baru dengan nama "HashValue". Di dalam metode **withColumn()**, kita menggunakan fungsi hash("jawaban") untuk menghasilkan nilai hash dari nilai dalam kolom "jawaban". **Fungsi hash()** digunakan untuk menghasilkan nilai hash dari suatu nilai.

Terakhir, kita menggunakan metode **show()** untuk menampilkan DataFrame yang telah diperbarui dengan penambahan kolom "HashValue".

```
[46] dt = data.select("soal", "skor_per_soal", "HashValue")
```

Dalam kode di atas, kita menggunakan metode **select()** pada objek DataFrame data untuk memilih kolom "soal", "skor_per_soal", dan "HashValue". Kita menyertakan nama kolom-kolom tersebut sebagai argumen dalam metode **select().**

Selanjutnya, kita menyimpan hasil pemilihan kolom tersebut dalam objek DataFrame baru yang diberi nama dt.

```
[47] #bagi data jadi 30% dan 70%
     splits = data.randomSplit([0.8, 0.2])
     train = splits[0].withColumnRenamed("skor_per_soal", "label")
     test = splits[1].withColumnRenamed("skor_per_soal", "trueLabel")
```

Dalam kode di atas, kita menggunakan metode **randomSplit()** pada objek DataFrame data untuk membagi data menjadi dua bagian dengan proporsi 80% dan 20%. Argumen [0.8, 0.2] menunjukkan proporsi masing-masing bagian.

Hasil dari pemisahan tersebut disimpan dalam objek splits, di mana elemen pertama (splits[0]) akan berisi 80% data sebagai data pelatihan, dan elemen kedua (splits[1]) akan berisi 20% data sebagai data pengujian.

Selanjutnya, kita menggunakan metode **withColumnRenamed()** untuk mengganti nama kolom "skor_per_soal" pada objek DataFrame train menjadi "label", dan pada objek DataFrame test menjadi "trueLabel". Ini dilakukan untuk memberikan nama yang lebih deskriptif pada kolom yang akan digunakan.

```
[48] #hitung jumlah data training dan testing
     train_rows = train.count()
     test_rows = test.count()
     print ("Jumlah baris data training:", train_rows,
            ", jumlah baris data testing:", test_rows)
```

```
Jumlah baris data training: 93 , jumlah baris data testing: 27
```

Dalam kode di atas, kita menggunakan metode **count()** pada objek DataFrame train dan test untuk menghitung jumlah baris (data) dalam masing-masing DataFrame tersebut.

Hasil perhitungan jumlah baris tersebut disimpan dalam variabel **train_rows** untuk data pelatihan (train) dan **test_rows** untuk data pengujian (test).

Selanjutnya, kita menggunakan perintah **print()** untuk menampilkan jumlah baris data training dan testing dengan pesan yang sesuai.

```
[49] train.show()
```

Perintah **train.show()** digunakan untuk menampilkan isi dari DataFrame train.

```
+----------+------------+--------------------+----+-----+-----------+
|       npm|nama_peserta|             jawaban|soal|label|  HashValue|
+----------+------------+--------------------+----+-----+-----------+
|         0|       Admin|1. Dibuat grafik ...|  12|100.0| 1727767227|
|         0|       Admin|1. Fokus dan expo...|  10|100.0|-1905649442|
|         0|       Admin|1. Kecocokan soft...|   9|100.0| 1576366224|
|         0|       Admin|1. Melindungi dan...|   5|100.0| 1588395990|
|         0|       Admin|Aturlah posisi pe...|   7|100.0|   50850002|
|         0|       Admin|Biaya dihitung be...|   2|100.0| 1183180174|
|         0|       Admin|Dijelaskan kepada...|   4|100.0|-2035408785|
|         0|       Admin|Hak cipta adalah ...|   3|100.0| 1232762403|
|         0|       Admin|Posisi Kepala dan...|   8|100.0| -945877996|
|1120020017|         RDW|aturlah posisi pe...|   7|86.22|-1392782412|
|1120020017|         RDW|dijelaskan kepada...|   4|72.06| -683553012|
|1120020017|         RDW|kecocokan softwar...|   9|65.89|-1544668284|
|1120020017|         RDW|memlihara sumber ...|   5| 88.3|   40672765|
|1120020017|         RDW|peralatan yang di...|  11|80.09|-1838887780|
|1120020017|         RDW|posisi kepala dan...|   8|89.17| 1044171719|
|1120020017|         RDW|posisi tubuh, pos...|   6|90.37|-1702735646|
|1120020017|         RDW|tidak, hanya memb...|   1|100.0| -256638840|
|1121020024|         IDP|aturlah posisi pe...|   7|100.0| 1123192925|
|1121020024|         IDP|biaya dihitung be...|   2|100.0| 1176853507|
|1121020024|         IDP|dibuat grafik yan...|  12|64.29| 1086045397|
+----------+------------+--------------------+----+-----+-----------+
only showing top 20 rows
```

Output tersebut merupakan tampilan dari DataFrame train yang menampilkan beberapa baris pertama dari data pelatihan setelah beberapa proses pemrosesan dan modifikasi. Setiap baris dalam output tersebut mewakili satu baris data dalam DataFrame train, dengan kolom-kolom tersebut.

Dengan menampilkan beberapa baris pertama dari data pelatihan, output tersebut memberikan gambaran tentang struktur dan isi dari DataFrame train, termasuk nilai-nilai dalam setiap kolom untuk setiap baris data yang ditampilkan.

```
[67] #mendefinisikan model
     from pyspark.ml.recommendation import ALS
     from pyspark.ml.evaluation import RegressionEvaluator
     als = ALS(maxIter=5, regParam=0.01, userCol='HashValue', itemCol='soal', ratingCol='label')
```

Dalam kode di atas, kita menggunakan modul pyspark.ml.recommendation untuk mengimpor kelas ALS yang digunakan untuk membangun model Collaborative Filtering. Kami juga mengimpor kelas RegressionEvaluator dari modul **pyspark.ml.evaluation**.

Selanjutnya, kita mendefinisikan objek als dengan menggunakan konstruktor ALS. Dalam konstruktor tersebut, kita mengatur beberapa parameter, seperti maxIter yang menentukan jumlah iterasi maksimum, regParam yang mengontrol parameter regularisasi, userCol yang menunjukkan kolom untuk identitas pengguna (dalam hal ini, menggunakan kolom "HashValue"), itemCol yang menunjukkan kolom untuk identitas item (dalam hal ini, menggunakan kolom "soal"), dan ratingCol yang menunjukkan kolom yang berisi nilai rating atau label.

```
[68] als = ALS(maxIter=5, regParam=0.01, userCol="HashValue",
               itemCol="soal", ratingCol="label")
     #training model dengan fungsi ".fit()"
     model = als.fit(train)
     print("Model telah selesai ditraining!")

     Model telah selesai ditraining!
```

Dalam kode di atas, kita menggunakan objek ALS yang telah didefinisikan sebelumnya untuk membangun model Collaborative Filtering. Objek ALS ini memiliki beberapa parameter yang telah ditentukan sebelumnya, seperti maxIter (jumlah iterasi maksimum), regParam (parameter regularisasi), userCol (kolom yang menyimpan identitas pengguna), itemCol (kolom yang menyimpan identitas item), dan ratingCol (kolom yang menyimpan nilai rating).

Selanjutnya, kita menggunakan metode **.fit()** pada objek ALS untuk melatih (meng-training) model Collaborative Filtering menggunakan data pelatihan (train). Proses pelatihan ini akan menggunakan metode ALS untuk mempelajari pola dan hubungan antara pengguna, item, dan nilai rating dalam data pelatihan.

Setelah pelatihan selesai, kita mencetak pesan "Model telah selesai ditraining!" untuk menandakan bahwa pelatihan model telah berhasil dilakukan.

```
[69] prediction = model.transform(test)
     prediction.show()
```

npm	nama_peserta	jawaban	soal	trueLabel	HashValue	prediction
1120020017	RDW	dibuat grafik yan...	12	86.53	-902409772	86.52996
0	Admin	Tidak, Hanya memb...	1	100.0	-2059296905	NaN
1121020024	IDP	tidak, hanya memb...	1	100.0	-256638840	99.99991
0	Admin	Ruang Komputer, P...	6	100.0	339970513	NaN
1220020018	AKM	ruang komputer, p...	6	100.0	1770907636	99.99995
1121020024	IDP	hak cipta adalah ...	3	83.43	-1876419705	83.429955
1120020017	RDW	emperbanyak cipta...	3	47.67	944450734	NaN
1121020036	DAR	kecocokan softwar...	9	84.88	-1083923656	NaN
1121020033	AP	dipaparkan pada k...	4	27.19	-1054163002	NaN
1121020024	IDP	dijelaskan kepada...	4	57.94	-97531001	NaN
1121020036	DAR	posisi kepala dan...	8	82.78	732095970	NaN
1121020036	DAR	aturlah posisi pe...	7	86.22	-1392782412	86.21998
1220020023	DRP	aturlah posisi pe...	7	86.22	-1392782412	86.21998
1220020018	AKM	fokus dan exposur...	10	94.51	503500283	NaN
1120020017	RDW	fokus dan exposur...	10	85.75	1988679646	NaN
1121020033	AP	1. perlengkapan y...	11	41.99	-2137389145	NaN
1121020032	ZA	peralatan yang di...	11	83.22	-275485461	NaN
0	Admin	1. Peralatan yang...	11	100.0	550139146	NaN
1220020023	DRP	peralatan yang di...	11	100.0	779050498	NaN
1220020018	AKM	peralatan yang di...	11	92.65	2048025308	NaN

only showing top 20 rows

Dalam kode di atas, kita menggunakan model Collaborative Filtering yang telah dilatih sebelumnya untuk melakukan prediksi menggunakan data pengujian (test). Kita menggunakan metode **.transform()** pada model untuk melakukan transformasi terhadap data pengujian.

Hasil prediksi tersebut disimpan dalam objek prediction, yang merupakan DataFrame baru. Kita menggunakan metode **.show()** untuk menampilkan beberapa baris pertama dari DataFrame prediction, sehingga dapat melihat hasil prediksi yang telah dihasilkan.

```
[70] from pyspark.ml.evaluation import RegressionEvaluator

     evaluator = RegressionEvaluator(
         labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
     rmse = evaluator.evaluate(prediction)
     print ("Root Mean square Error (RMSE):", rmse)
```

```
Root Mean square Error (RMSE): nan
```

Dalam kode di atas, kita menggunakan modul pyspark.ml.evaluation untuk mengimpor kelas RegressionEvaluator. Kemudian, kita mendefinisikan objek evaluator menggunakan konstruktor RegressionEvaluator. Pada konstruktor tersebut, kita mengatur beberapa parameter, seperti labelCol yang menunjukkan kolom yang berisi nilai label yang sebenarnya (dalam hal ini, "trueLabel"), dan predictionCol yang menunjukkan kolom yang berisi nilai prediksi (dalam hal ini, "prediction").

Selanjutnya, kita menggunakan metode **.evaluate()** pada objek evaluator untuk menghitung metrik evaluasi regresi, yaitu RMSE (Root Mean Square Error), dengan membandingkan nilai label yang sebenarnya dengan nilai prediksi yang dihasilkan dari DataFrame prediction.

Hasil RMSE disimpan dalam variabel rmse, dan kemudian dicetak menggunakan perintah **print()** untuk menampilkan nilai RMSE yang telah dihitung.

```
[71] prediction.count()
     a = prediction.count()
     print("jumlah baris sebelum di hapus data kosong: ", a)
     cleanPred = prediction.dropna(how="any", subset=["prediction"])
     b = cleanPred.count()
     print("jumlah baris setelah di hapus data kosong: ", b)
     print("jumlah baris data kosong: ", a-b)
```

```
jumlah baris sebelum di hapus data kosong:  24
jumlah baris setelah di hapus data kosong:  9
jumlah baris data kosong:  15
```

Dalam kode di atas, kita menggunakan metode **.count()** pada objek prediction untuk menghitung jumlah baris dalam DataFrame prediction. Hasil perhitungan tersebut disimpan dalam variabel a dan kemudian dicetak untuk menampilkan jumlah baris sebelum menghapus data kosong.

Selanjutnya, kita menggunakan metode **.dropna()** pada objek prediction untuk menghapus baris yang memiliki nilai kosong (NaN) dalam kolom "prediction". Parameter how="any" menunjukkan bahwa baris akan dihapus jika terdapat setidaknya satu nilai kosong dalam kolom yang ditentukan (dalam hal ini, "prediction"). Parameter subset=["prediction"] digunakan untuk menentukan kolom yang akan diperiksa untuk nilai kosong.

Setelah menghapus data kosong, kita menggunakan metode **.count()** kembali pada DataFrame cleanPred (DataFrame yang telah dihapus data kosong) untuk menghitung jumlah baris yang tersisa setelah penghapusan data kosong. Hasil perhitungan tersebut disimpan dalam variabel b dan dicetak untuk menampilkan jumlah baris setelah menghapus data kosong.

Terakhir, kita menggunakan perintah **print()** untuk mencetak selisih antara jumlah baris sebelum dan setelah penghapusan data kosong, yang mengindikasikan jumlah baris data kosong yang telah dihapus.

```
[72] #evaluasi mse
     evaluator = RegressionEvaluator(metricName="mse", labelCol="trueLabel", predictionCol="prediction")
     mse = evaluator.evaluate(prediction)
     print("Mean Squares Error = " + str(mse))

     Mean Squares Error = nan
```

Dalam kode di atas, kita menggunakan modul **pyspark.ml.evaluation** untuk mengimpor kelas RegressionEvaluator. Kemudian, kita mendefinisikan objek evaluator menggunakan konstruktor RegressionEvaluator. Pada konstruktor tersebut, kita mengatur beberapa parameter, seperti metricName yang diatur sebagai "mse" untuk menghitung Mean Squared Error (MSE), labelCol yang menunjukkan kolom yang berisi nilai label yang sebenarnya (dalam hal ini, "trueLabel"), dan predictionCol yang menunjukkan kolom yang berisi nilai prediksi (dalam hal ini, "prediction").

Selanjutnya, kita menggunakan metode **.evaluate()** pada objek evaluator untuk menghitung MSE antara nilai prediksi yang dihasilkan dari model dengan nilai label yang sebenarnya pada data pengujian. Hasil MSE disimpan dalam variabel mse, dan kemudian dicetak menggunakan perintah **print()** untuk menampilkan nilai MSE yang telah dihitung.