



## Worksheet 2

**Big Data dan Analitik**

**TI Kelas A**

**Nama:**

**Acik Imtia Chana (235150701111038)**

**Hanifa Syifa Safitri (235150707111031)**

**Mutiara Rosida Sholihat (235150707111035)**

**Lutfiah Nailil Izzah (235150707111038)**

**Sylvasisca Andini Faradyan (23517070111040)**

**Dosen:**

Putra Pandu Adikara, S.Kom., M.Kom.



**Program Studi Teknologi Informasi  
Jurusan Sistem Informasi  
Universitas Brawijaya  
2025**

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>1</b>
<b>BAB 1 PENDAHULUAN.....</b>	<b>3</b>
1.1 Latar Belakang.....	3
1.2 Pembatasan Masalah.....	3
1.3 Rumusan Masalah.....	4
1.4 Tujuan.....	4
<b>BAB 2 KAJIAN PUSTAKA.....</b>	<b>5</b>
2.1 Supervised Learning.....	5
2.1.1 Pengertian.....	5
2.1.2 Ciri-ciri dan Proses Umum.....	5
2.2 Logistic Regression.....	5
2.2.1 Konsep Dasar.....	5
2.2.2 Fungsi Aktivasi dan Interpretasi.....	5
2.3 Decision Tree.....	5
2.3.1 Pemilihan Atribut dan Struktur.....	5
2.3.2 Kelebihan dan Kekurangan.....	6
2.4 Random Forest.....	6
2.4.1 Prinsip Ensemble Learning.....	6
2.4.2 Mekanisme Voting dan Randomness.....	6
2.5 Evaluasi Model.....	6
2.5.1 Confusion Matrix.....	6
2.5.2 Accuracy, Precision, Recall, F1-Score.....	6
2.6 Dataset Customer Experience.....	7
2.6.2 Proses Preprocessing.....	7
<b>BAB 3 METODOLOGI PENELITIAN.....</b>	<b>8</b>
3.1 Dataset dan Sumber Data.....	8
3.2 Machine Learning Pipeline.....	8
3.3 Proses Preprocessing.....	8
3.4 Implementasi Algoritma.....	8
3.5 Evaluasi Model.....	9
3.6 Tools dan Software.....	9
<b>BAB 4 IMPLEMENTASI.....</b>	<b>10</b>
4.1 Instalasi dan Konfigurasi.....	10
4.1.1 Instalasi dan Konfigurasi Apache Spark.....	10
4.1.2 Instalasi dan Konfigurasi PySpark.....	12
4.2 Data disimpan dalam HDFS.....	15
4.2.1 Membuat directory 'dataset' di dalam HDFS dan memindahkan file csv ke directory 'dataset'....	15
4.3 Proses ETL dan Preprocessing.....	16

4.3.1 Extract.....	16
4.3.2 Transformasi Data.....	17
4.3.3 Load Data.....	18
4.3.4 Preprocessing.....	19
4.4 Pelatihan Model.....	20
4.4.1 Logistic Regression.....	20
4.4.2 Decision Tree.....	20
4.4.3 Random Forest.....	21
4.4.4 Perbandingan Akurasi.....	21
4.5 Prediksi dan Visualisasi.....	22
4.5.1 Hasil Prediksi.....	22
4.5.2 Visualisasi.....	22
4.5.3 Classification Report.....	25
<b>BAB 5 PEMBAHASAN DAN HASIL.....</b>	<b>27</b>
5.1 Pengujian Model.....	27
5.1.1 Hasil Evaluasi Logistic Regression.....	27
5.1.2 Hasil Evaluasi Decision Tree.....	27
5.1.3 Hasil Evaluasi Random Forest.....	27
5.2 Analisis Hasil.....	27
5.3 Tabel Sampel Data Uji.....	28
5.3.1 Data Mentah dan Preprocessing.....	28
5.3.2 Kelas Aktual dan Prediksi.....	30
5.3.3 Visualisasi Korelasi antar Fitur.....	32
5.3.4 Sampel Data Uji dengan Hasil Prediksi.....	33
<b>BAB 6 KESIMPULAN DAN SARAN.....</b>	<b>35</b>
6.1 Kesimpulan.....	35
6.2 Saran.....	35
<b>DAFTAR PUSTAKA.....</b>	<b>36</b>

## BAB 1 PENDAHULUAN

### 1.1 Latar Belakang

Dalam era digital saat ini, data pelanggan menjadi aset penting yang dapat dimanfaatkan untuk mendukung pengambilan keputusan perusahaan. Salah satu permasalahan krusial yang sering dihadapi oleh perusahaan adalah retensi pelanggan, yaitu kemampuan untuk mempertahankan pelanggan yang sudah ada. Pelanggan yang berhenti menggunakan produk atau layanan (*customer churn*) dapat menyebabkan kerugian signifikan, sehingga prediksi terhadap kemungkinan *churn* menjadi penting dilakukan.

Masalah tersebut dapat diatasi dengan pendekatan berbasis *machine learning supervised classification* yang sering dimanfaatkan karena dapat menganalisis data historis untuk membentuk model prediksi yang akurat. Penelitian ini menggunakan tiga metode klasifikasi yang populer dan efektif, yaitu *Logistic Regression*, *Decision Tree*, dan *Random Forest*. Menurut Atay dan Turanli (2024), *Random Forest* menunjukkan akurasi terbaik sebesar 95,62% dalam prediksi *churn* pelanggan. Sedangkan, *Logistic Regression* menawarkan interpretabilitas yang kuat dan *Decision Tree* efektif dalam menangkap interaksi kompleks antar fitur. Di sisi lain, Hu et al. (2020) menekankan efektivitas *Random Forest* dalam klasifikasi biner pada sektor perbankan dan pelanggan.

Menurut penelitian oleh Kumar et al. (2024), metode *K-Nearest Neighbors* (K-NN) memiliki keterbatasan dalam hal efisiensi komputasi dan sensitivitas terhadap skala data, sedangkan *Support Vector Machine* (SVM) memerlukan tuning parameter yang kompleks dan sulit diinterpretasikan dalam konteks bisnis. Sebaliknya, *Logistic Regression*, *Decision Tree*, dan *Random Forest* lebih mudah diimplementasikan dan diinterpretasikan, sehingga lebih cocok untuk aplikasi bisnis yang membutuhkan hasil cepat dan akurat (Kumar et al., 2024).

Dengan demikian, penelitian ini bertujuan untuk mengimplementasikan ketiga algoritma tersebut untuk memprediksi status retensi pelanggan berdasarkan data interaksi dan pengalaman mereka, serta melakukan evaluasi terhadap performa masing-masing model guna menentukan pendekatan terbaik.

### 1.2 Pembatasan Masalah

Penelitian ini memiliki batasan tertentu agar ruang lingkup tetap terfokus dan analisis dapat dilakukan dapat dilakukan secara mendalam antara lain:

1. Ruang lingkup data terbatas pada dataset *customer experience* yang telah tersedia dan disimpan dalam format CSV pada sistem *Hadoop Distributed File System* (HDFS).
2. Proses ETL (*Extract, Transform, Load*) hanya dilakukan menggunakan *framework* Apache Spark, dan tidak mencakup teknologi lain seperti Apache Flink atau Hive.
3. Model klasifikasi yang digunakan terbatas pada *Logistic Regression*, *Decision Tree*, dan *Random Forest*. Metode lain seperti SVM, K-NN, atau XGBoost tidak digunakan karena pertimbangan interpretabilitas, efisiensi, dan kompleksitas implementasi dalam skenario bisnis.
4. Evaluasi model hanya menggunakan *accuracy matrix*, *precision*, *recall*, *F1-score*, dan AUC. Evaluasi berbasis cost-benefit analysis atau dampak bisnis tidak dilakukan secara mendalam.
5. Visualisasi hasil difokuskan pada dashboard berbasis matplotlib.
6. Tujuan penelitian bersifat eksperimental, sehingga sistem yang dibangun hanya bersifat prototipe dan belum mencakup aspek deployment, integrasi API, ataupun pengujian skalabilitas dalam lingkungan produksi nyata.

### 1.3 Rumusan Masalah

1. Bagaimana proses ekstraksi, transformasi, dan pemuatan (ETL) data pengalaman pelanggan dapat dilakukan secara efisien dalam lingkungan komputasi terdistribusi Hadoop dan Apache Spark?
2. Fitur-fitur atau variabel apa saja yang paling berkontribusi terhadap retensi atau kehilangan pelanggan, dan bagaimana hubungan antar fitur tersebut?
3. Seberapa efektif model machine learning yang dibangun dengan PySpark dalam memprediksi status retensi pelanggan secara akurat?
4. Bagaimana visualisasi data dan hasil model dapat dikemas dalam bentuk dashboard interaktif guna mendukung pengambilan keputusan bisnis yang berbasis data?

### 1.4 Tujuan

Percobaan ini bertujuan untuk membangun sistem prediksi retensi pelanggan berbasis big data dengan mengintegrasikan teknologi Apache Spark, PySpark, dan Hadoop dalam proses analisis dan visualisasi data pelanggan. Tujuan meliputi :

1. Merancang dan mengimplementasikan pipeline ETL berbasis Apache Spark untuk memproses data pengalaman pelanggan secara paralel dan efisien dalam lingkungan Hadoop Distributed File System (HDFS).
2. Menganalisis data pelanggan guna mengidentifikasi pola, fitur penting, dan variabel yang berpengaruh signifikan terhadap tingkat retensi pelanggan, melalui analisis korelasi dan metode feature importance.
3. Membangun model klasifikasi menggunakan algoritma Logistic Regression, Random Forest, dan Decision Tree yang diimplementasikan dalam PySpark serta mengevaluasi performanya menggunakan metrik akurasi, precision, recall, F1-score, dan UAC.
4. Menyediakan visualisasi interaktif dalam bentuk dashboard menggunakan matplotlib untuk menyajikan hasil analisis dan prediksi kepada pemangku kepentingan secara informatif, intuitif, dan mudah diakses.
5. Mendukung proses pengambilan keputusan strategis dalam mempertahankan pelanggan, melalui sistem yang scalable, data driven, dan berbasis pada teknologi komputasi terdistribusi terkini.

## BAB 2 KAJIAN PUSTAKA

### 2.1 Supervised Learning

#### 2.1.1 Pengertian

Supervised learning adalah metode machine learning di mana model dilatih menggunakan data berlabel untuk memprediksi output berdasarkan pola yang ditemukan. Pendekatan ini sangat relevan untuk masalah klasifikasi biner seperti prediksi retensi pelanggan, yang menjadi fokus penelitian ini. Menurut Brownlee (2020), supervised learning melibatkan pelatihan model dengan data input-output berlabel untuk membuat prediksi akurat pada data baru.

#### 2.1.2 Ciri-ciri dan Proses Umum

Ciri utama supervised learning adalah penggunaan data berlabel dan pembagian data menjadi set pelatihan (training) dan pengujian (testing). Prosesnya meliputi pengumpulan data, preprocessing, pelatihan model, evaluasi, dan prediksi. Russell dan Norvig (2020) menyatakan bahwa supervised learning efektif untuk masalah seperti prediksi churn pelanggan karena dapat menangani hubungan linier maupun non-linier antar variabel, yang sering ditemukan dalam data pengalaman pelanggan.

### 2.2 Logistic Regression

#### 2.2.1 Konsep Dasar

Logistic Regression adalah algoritma klasifikasi untuk memprediksi probabilitas kejadian biner, seperti retensi atau churn pelanggan. Algoritma ini dipilih sebagai model dasar dalam penelitian ini karena sifatnya yang interpretable. Menurut Brownlee (2020), Logistic Regression cocok untuk dataset dengan distribusi bervariasi, seperti data pengalaman pelanggan, karena kemampuannya dalam memodelkan probabilitas.

#### 2.2.2 Fungsi Aktivasi dan Interpretasi

Logistic Regression menggunakan fungsi sigmoid untuk menghasilkan probabilitas dalam rentang [0,1], yang kemudian diklasifikasikan menggunakan ambang (threshold) 0,5. Powers (2020) menjelaskan bahwa koefisien dalam Logistic Regression dapat diinterpretasikan sebagai log odds ratio, yang berguna untuk memahami dampak fitur seperti skor kepuasan terhadap retensi pelanggan.

### 2.3 Decision Tree

#### 2.3.1 Pemilihan Atribut dan Struktur

Decision Tree membagi data berdasarkan fitur yang paling informatif menggunakan kriteria seperti Gini Index. Dalam konteks penelitian ini, Decision Tree digunakan untuk mengidentifikasi fitur penting seperti jumlah interaksi pelanggan. Brownlee (2020) menjelaskan

bahwa pemilihan atribut didasarkan pada pengurangan entropi, yang efektif untuk dataset dengan fitur campuran seperti data Customer Experience.

### 2.3.2 Kelebihan dan Kekurangan

Decision Tree memiliki kelebihan dalam interpretasi yang mudah, tetapi rentan terhadap overfitting. Brownlee (2020) menyarankan pruning untuk mengurangi overfitting, yang relevan untuk memastikan model dapat digeneralisasi pada data pengujian dalam konteks retensi pelanggan.

## 2.4 Random Forest

### 2.4.1 Prinsip Ensemble Learning

Random Forest adalah metode ensemble yang menggabungkan beberapa Decision Tree untuk meningkatkan akurasi. Dalam penelitian ini, Random Forest digunakan untuk analisis feature importance, seperti mengidentifikasi variabel yang paling mempengaruhi retensi. Menurut Sam et al. (2024), Random Forest menunjukkan performa prediksi yang lebih tinggi dibandingkan K-Nearest Neighbors, Support Vector Machines, dan Decision Trees dalam prediksi churn pelanggan di sektor telekomunikasi, menjadikannya pilihan yang kuat untuk dataset kompleks.

### 2.4.2 Mekanisme Voting dan Randomness

Random Forest menggunakan voting mayoritas untuk prediksi akhir. Randomness melalui pemilihan acak subset data dan fitur membuat model lebih robust. Kumar et al. (2024) menekankan bahwa Random Forest efektif untuk menangani ketidakseimbangan kelas dan interaksi antar fitur, seperti hubungan antara waktu di situs dan skor kepuasan pelanggan.

## 2.5 Evaluasi Model

### 2.5.1 Confusion Matrix

Confusion Matrix digunakan untuk mengevaluasi performa model dengan membandingkan prediksi terhadap kelas aktual, menghasilkan metrik seperti True Positive dan False Negative. Fawcett (2006) menjelaskan bahwa Confusion Matrix penting untuk memahami distribusi prediksi, yang relevan untuk mengevaluasi model retensi pelanggan.

### 2.5.2 Accuracy, Precision, Recall, F1-Score

Metrik evaluasi yang digunakan meliputi Accuracy, Precision, Recall, dan F1-Score. Accuracy mengukur proporsi prediksi benar, Precision dan Recall mengevaluasi ketepatan dan sensitivitas, sedangkan F1-Score menyeimbangkan keduanya. Powers (2020) menyatakan bahwa F1-Score sangat relevan untuk data yang tidak seimbang, seperti dataset retensi pelanggan

## 2.6 Data dan Preprocessing

### 2.6.1 Dataset Customer Experience

Dataset yang digunakan adalah Customer Experience dari Kaggle, tersedia di <https://www.kaggle.com/datasets/ziya07/customer-experience-dataset/data>. Dataset ini berisi atribut seperti umur, jenis kelamin, jumlah interaksi, skor umpan balik, jumlah produk, waktu di situs, dan skor kepuasan, yang disimpan dalam format CSV di HDFS untuk diproses dengan Spark.

### 2.6.2 Proses Preprocessing

Preprocessing melibatkan: (1) penghapusan duplikat dengan `dropDuplicates()`, (2) penanganan missing values menggunakan mean imputation untuk data numerik dan label 'Unknown' untuk kategorikal, (3) encoding kategorikal dengan `StringIndexer`, (4) feature engineering (contoh: `AvgSpendPerPurchase`), dan (5) scaling dengan `StandardScaler`. Brownlee (2020) menekankan bahwa preprocessing meningkatkan kualitas data untuk performa model yang lebih baik.

## 2.7 Machine Learning Pipeline dengan Apache Spark

Pipeline machine learning dibangun menggunakan Spark MLlib dengan tahapan: (1) ingest data dari HDFS, (2) preprocessing dan feature engineering, (3) pemodelan dengan Logistic Regression, Decision Tree, dan Random Forest, (4) evaluasi performa, dan (5) visualisasi hasil dengan Matplotlib. Zaharia et al. (2016) menyatakan bahwa Spark mendukung pemrosesan paralel, yang memungkinkan ETL efisien pada dataset besar seperti data pelanggan.

## BAB 3 METODOLOGI PENELITIAN

### 3.1 Dataset dan Sumber Data

Penelitian ini menggunakan dataset Customer Experience Dataset yang diperoleh dari platform Kaggle dengan URL resmi: <https://www.kaggle.com/datasets/ziya07/customer-experience-dataset>. Dataset ini terdiri atas berbagai atribut pelanggan seperti umur (age), jenis kelamin (gender), jumlah interaksi dengan layanan pelanggan (num\_interactions), skor umpan balik (feedback\_score), jumlah produk yang dibeli (products\_purchased), waktu yang dihabiskan di situs (time\_spent\_on\_site), serta skor kepuasan pelanggan (satisfaction\_score). Tujuan penggunaan dataset ini adalah untuk memodelkan dan memprediksi retention status atau status retensi pelanggan menggunakan pendekatan machine learning berbasis PySpark.

### 3.2 Machine Learning Pipeline

Proses analisis dalam penelitian ini dirancang dengan menggunakan pendekatan machine learning pipeline berbasis Apache Spark MLlib. Pipeline ini dibangun secara modular yang terdiri atas tahap-tahap berikut:

1. Ingesti data dari HDFS.
2. Preprocessing dan feature engineering, termasuk encoding dan normalisasi fitur numerik.
3. Pemodelan menggunakan beberapa algoritma klasifikasi.
4. Evaluasi performa model.
5. Visualisasi dan interpretasi hasil.

Dengan arsitektur pipeline ini, proses pengolahan data menjadi lebih sistematis, replikasi model menjadi lebih mudah, serta dapat dijalankan secara efisien di lingkungan big data.

### 3.3 Proses Preprocessing

Tahapan preprocessing dilakukan untuk membersihkan dan menyiapkan data sebelum dimasukkan ke dalam model. Adapun langkah-langkah utama preprocessing dalam penelitian ini adalah sebagai berikut:

1. Pembuangan duplikasi data menggunakan dropDuplicates().
2. Penanganan nilai hilang (missing values) pada data numerik dengan mengisi nilai rata-rata (mean imputation), dan pada data kategorikal dengan mengganti nilai kosong dengan label 'Unknown'.
3. Encoding data kategorikal seperti gender dan lokasi dengan teknik StringIndexer.
4. Feature engineering dengan menambahkan variabel baru seperti AvgSpendPerPurchase dan HighSatisfaction yang diturunkan dari variabel lain.
5. Scaling fitur numerik menggunakan StandardScaler setelah membentuk vektor fitur menggunakan VectorAssembler.

### 3.4 Implementasi Algoritma

Dalam penelitian ini, dilakukan implementasi dan komparasi beberapa algoritma klasifikasi berbasis *supervised learning* untuk memprediksi status retensi pelanggan. Algoritma yang digunakan merupakan bagian dari pustaka Apache Spark MLlib, dan dipilih berdasarkan efisiensi serta

interpretabilitas dalam konteks data berskala besar. Adapun ketiga algoritma yang diimplementasikan adalah sebagai berikut:

1. Logistic Regression  
Logistic Regression digunakan sebagai model dasar (*baseline model*) dalam klasifikasi status retensi pelanggan.
2. Random Forest Classifier  
Random Forest adalah Model *ensemble learning* berbasis pohon keputusan, yang juga digunakan untuk melakukan analisis *feature importance*.
3. Decision Tree Classifier  
Decision Tree merupakan model pohon keputusan yang bekerja dengan cara membagi dataset berdasarkan atribut yang paling informatif untuk memisahkan kelas target.

### 3.5 Evaluasi Model

Model dievaluasi menggunakan metrik evaluasi klasifikasi yang umum digunakan dalam penelitian machine learning, yaitu:

1. Accuracy: Mengukur rasio prediksi yang benar terhadap total data.
2. Precision dan Recall: Mengukur relevansi dan sensitivitas dari model.
3. F1-Score: Harmonic mean dari precision dan recall, berguna dalam kondisi data imbalance.
4. Area Under Curve (AUC): Digunakan untuk mengevaluasi performa model binary classification secara probabilistik.
5. Confusion Matrix: Digunakan untuk visualisasi distribusi hasil prediksi terhadap kelas aktual.

Evaluasi dilakukan pada data uji setelah pembagian dataset menjadi 80% data pelatihan dan 20% data pengujian.

### 3.6 Tools dan Software

Seluruh proses penelitian dilakukan menggunakan platform dan perangkat lunak berikut:

1. Apache Spark 3.4.1: Digunakan untuk pemrosesan data terdistribusi dan implementasi pipeline machine learning.
2. PySpark (Python API for Spark): Digunakan untuk scripting pipeline machine learning dan preprocessing data.
3. Hadoop HDFS: Digunakan untuk penyimpanan terdistribusi data pelanggan.
4. Anaconda 3 & Python 3.8: Digunakan sebagai environment manajemen dependensi.
5. Jupyter Notebook: Digunakan sebagai platform pengembangan interaktif.
6. Matplotlib & Plotly: Digunakan untuk visualisasi data dan pembuatan dashboard interaktif.

## BAB 4 IMPLEMENTASI

### 4.1 Instalasi dan Konfigurasi

#### 4.1.1 Instalasi dan Konfigurasi Apache Spark

1. Cek java sudah terinstall

Dengan perintah `java -version`

```
hdacik@hadoop-master:~$ java -version
openjdk version "11.0.26" 2025-01-21
OpenJDK Runtime Environment (build 11.0.26+4-post-Ubuntu-1ubuntu124.04)
OpenJDK 64-Bit Server VM (build 11.0.26+4-post-Ubuntu-1ubuntu124.04, mixed mode,
sharing)
```

Gambar 4.1.1.1 Cek Versi Java Terinstal

2. Mengunduh Apache Spark dengan link

<https://dlcdn.apache.org/spark/spark-4.0.0/spark-4.0.0-bin-hadoop3.tgz> yang berada di website Apache Spark. Cara melakukannya dengan perintah:

`wget`

<https://dlcdn.apache.org/spark/spark-4.0.0/spark-4.0.0-bin-hadoop3.tgz>

```
hdacik@hadoop-master:~/apps$ wget ^[[200~ https://dlcdn.apache.org/spark/spark-3.5.5/spark-3.5.5-bin-hadoop3.tgz
http://: Invalid host name.
--2025-05-16 04:49:31-- https://dlcdn.apache.org/spark/spark-3.5.5/spark-3.5.5-bin-hadoop3.tgz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 400724056 (382M) [application/x-gzip]
Saving to: 'spark-3.5.5-bin-hadoop3.tgz'

spark-3.5.5-bin-had 100%[=====>] 382.16M  1.13MB/s   in 4m 22s

2025-05-16 04:53:55 (1.46 MB/s) - 'spark-3.5.5-bin-hadoop3.tgz' saved [400724056/400724056]

FINISHED --2025-05-16 04:53:55--
Total wall clock time: 4m 23s
Downloaded: 1 files. 382M in 4m 22s (1.46 MB/s)
```

Gambar 4.1.1.2 Mengunduh Apache Spark via wget

3. Melakukan ekstraksi dari file yang sudah diunduh

`tar -zxvf spark-3.5.5-bin-hadoop3.tgz`

```
hdacik@hadoop-master:~/apps$ tar -zvxf spark-3.5.5-bin-hadoop3.tgz
spark-3.5.5-bin-hadoop3/
spark-3.5.5-bin-hadoop3/jars/
spark-3.5.5-bin-hadoop3/jars/HikariCP-2.5.1.jar
spark-3.5.5-bin-hadoop3/jars/JLargeArrays-1.5.jar
spark-3.5.5-bin-hadoop3/jars/JTransforms-3.1.jar
spark-3.5.5-bin-hadoop3/jars/RoaringBitmap-0.9.45.jar
spark-3.5.5-bin-hadoop3/jars/ST4-4.0.4.jar
spark-3.5.5-bin-hadoop3/jars/activation-1.1.1.jar
spark-3.5.5-bin-hadoop3/jars/aircompressor-0.27.jar
spark-3.5.5-bin-hadoop3/jars/algebra_2.12-2.0.1.jar
spark-3.5.5-bin-hadoop3/jars/annotations-17.0.0.jar
spark-3.5.5-bin-hadoop3/jars/antlr-runtime-3.5.2.jar
spark-3.5.5-bin-hadoop3/jars/antlr4-runtime-4.9.3.jar
spark-3.5.5-bin-hadoop3/jars/aopalliance-repackaged-2.6.1.jar
spark-3.5.5-bin-hadoop3/jars/arpack-3.0.3.jar
```

Gambar 4.1.1.3 Mengekstrak File Spark .tgz

#### 4. Cek hasil ekstraksi dengan ls -l spark-3.5.5-bin-hadoop3

```
hdacik@hadoop-master:~/apps$ ls -la spark-3.5.5-bin-hadoop3
total 164
drwxr-xr-x 13 hdacik hdacik 4096 Feb 24 03:45 .
drwxrwxr-x  4 hdacik hdacik 4096 May 16 04:55 ..
drwxr-xr-x  2 hdacik hdacik 4096 Feb 24 03:45 bin
drwxr-xr-x  2 hdacik hdacik 4096 Feb 24 03:45 conf
drwxr-xr-x  6 hdacik hdacik 4096 Feb 24 03:45 data
drwxr-xr-x  4 hdacik hdacik 4096 Feb 24 03:45 examples
drwxr-xr-x  2 hdacik hdacik 20480 Feb 24 03:45 jars
drwxr-xr-x  4 hdacik hdacik 4096 Feb 24 03:45 kubernetes
-rw-r--r--  1 hdacik hdacik 22916 Feb 24 03:45 LICENSE
drwxr-xr-x  2 hdacik hdacik 4096 Feb 24 03:45 licenses
-rw-r--r--  1 hdacik hdacik 57842 Feb 24 03:45 NOTICE
drwxr-xr-x  9 hdacik hdacik 4096 Feb 24 03:45 python
drwxr-xr-x  3 hdacik hdacik 4096 Feb 24 03:45 R
-rw-r--r--  1 hdacik hdacik 4605 Feb 24 03:45 README.md
-rw-r--r--  1 hdacik hdacik 166 Feb 24 03:45 RELEASE
drwxr-xr-x  2 hdacik hdacik 4096 Feb 24 03:45 sbin
drwxr-xr-x  2 hdacik hdacik 4096 Feb 24 03:45 yarn
```

Gambar 4.1.1.4 Verifikasi Hasil Ekstraksi Spark

#### 5. Set Environment Variables

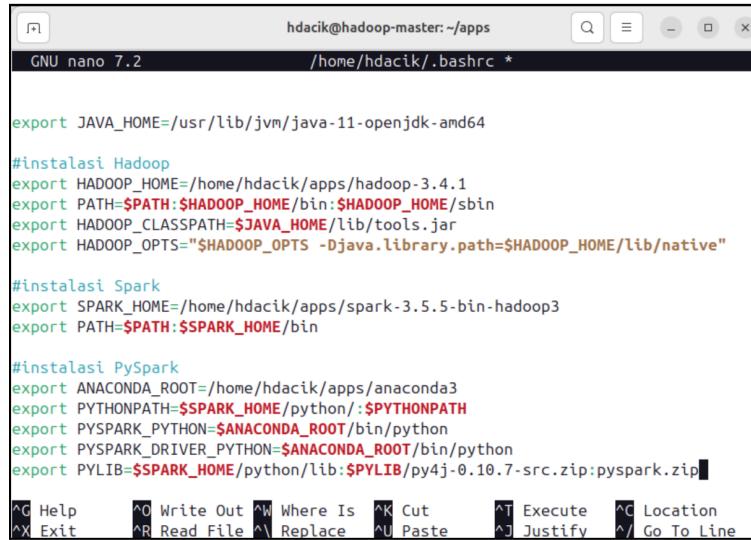
- Sunting file dengan nano dengan perintah:

nano ~/.bashrc

```
hdacik@hadoop-master:~/apps$ nano ~/.bashrc
```

- Menambahkan variabel baru bernama JAVA\_HOME di env var dan menetapkan folder dari JDK. Setelah itu, menambahkan variabel \$JAVA\_HOME ke variabel PATH.
- Menambahkan variabel baru bernama HADOOP\_HOME di env var dan menetapkan folder dari hadoop. Setelah itu, menambahkan variabel \$HADOOP\_HOME, \$HADOOP/bin, dan \$HADOOP/sbin ke variabel PATH.
- Menambahkan variabel baru bernama SPARK\_HOME di env var dan menetapkan folder dari Spark. Setelah itu, menambahkan variabel \$SPARK\_HOME, \$SPARK\_HOME/bin ke variabel PATH.
- Menambahkan variabel lain yang diperlukan nanti saat instalasi Anaconda dan PySpark. Setelah itu, menambahkan ANACONDA\_ROOT, PYTHONPATH,

PYSPARK\_PYTHON, PYSPARK\_DRIVER, PYLIB. Lalu, menambahkan variabel tersebut ke variabel PATH.



```

hdacik@hadoop-master:~/apps
GNU nano 7.2 /home/hdacik/.bashrc *

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

#instalasi Hadoop
export HADOOP_HOME=/home/hdacik/apps/hadoop-3.4.1
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
export HADOOP_OPTS="$HADOOP_OPTS -Djava.library.path=$HADOOP_HOME/lib/native"

#instalasi Spark
export SPARK_HOME=/home/hdacik/apps/spark-3.5.5-bin-hadoop3
export PATH=$PATH:$SPARK_HOME/bin

#instalasi PySpark
export ANACONDA_ROOT=/home/hdacik/apps/anaconda3
export PYTHONPATH=$SPARK_HOME/python/:$PYTHONPATH
export PYSPARK_PYTHON=$ANACONDA_ROOT/bin/python
export PYSPARK_DRIVER_PYTHON=$ANACONDA_ROOT/bin/python
export PYLIB=$SPARK_HOME/python/lib:$PYLIB/py4j-0.10.7-src.zip:pyspark.zip

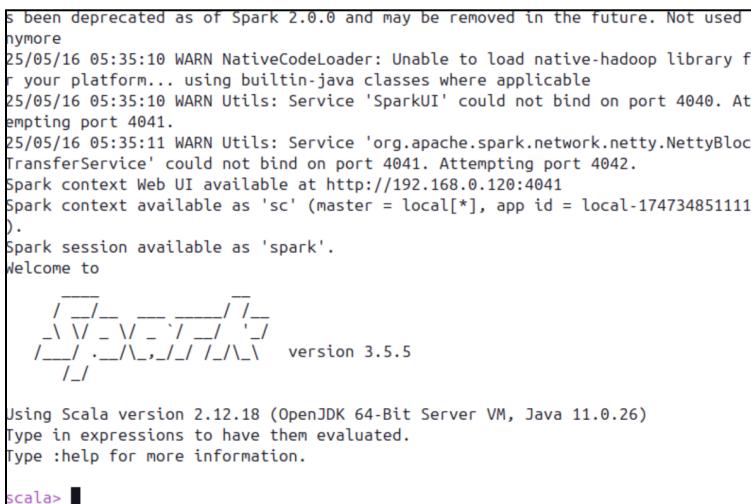
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^M Replace ^U Paste ^J Justify ^L Go To Line

```

Gambar 4.1.1.5 Menambahkan Environment Variables di .bashrc

## 6. Test Spark Shell

Membuka command prompt dan ketik spark-shell lalu tekan enter. Maka, logo Spark akan muncul pada terminal dengan bahasa native yaitu Scala.



```

s been deprecated as of Spark 2.0.0 and may be removed in the future. Not used a
nymore
25/05/16 05:35:10 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
25/05/16 05:35:10 WARN Utils: Service 'SparkUI' could not bind on port 4040. Att
empting port 4041.
25/05/16 05:35:11 WARN Utils: Service 'org.apache.spark.network.netty.NettyBlock
TransferService' could not bind on port 4041. Attempting port 4042.
Spark context Web UI available at http://192.168.0.120:4041
Spark context available as 'sc' (master = local[*], app id = local-1747348511118
).
Spark session available as 'spark'.
Welcome to

    / \   _ / \_ / \_ / \_ 
    \ \ / \ / \ / \ / \ / \   version 3.5.5
     / \ / \ / \ / \ / \ / \
    / \ / \ / \ / \ / \ / \ / \ 

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.26)
Type in expressions to have them evaluated.
Type :help for more information.

scala>

```

Gambar 4.1.1.6 Menjalankan Spark Shell

### 4.1.2 Instalasi dan Konfigurasi PySpark

- Instalasi Python Anaconda terlebih dahulu karena untuk memudahkan proses instalasi PySpark.

wget

[https://repo.anaconda.com/archive/Anaconda3-2024.10-Linux-x86\\_64.sh](https://repo.anaconda.com/archive/Anaconda3-2024.10-Linux-x86_64.sh)

```
hdacik@hadoop-master:~/apps$ wget https://repo.anaconda.com/archive/Anaconda3-20
24.10-1-Linux-x86_64.sh
--2025-05-16 05:40:10-- https://repo.anaconda.com/archive/Anaconda3-2024.10-1-L
inux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.191.158, 104.16.32.241
, 2606:4700::6810:20f1, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.191.158|:443... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 1102495056 (1.0G) [application/octet-stream]
Saving to: 'Anaconda3-2024.10-1-Linux-x86_64.sh'

An 2%[          ] 25.21M 4.89MB/s eta 7m 10s S
```

Gambar 4.1.2.1 Mengunduh Installer Anaconda

- Menjalankan installer dari Anaconda dengan memanggil bash  
bash Anaconda3-2024.10-1-Linux-x86\_64.sh

```
hdacik@hadoop-master:~/apps$ bash Anaconda3-2024.10-1-Linux-x86_64.sh

Welcome to Anaconda3 2024.10-1

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
ANACONDA TERMS OF SERVICE
Please read these Terms of Service carefully before purchasing, using, accessing
, or downloading any Anaconda Offerings (the "Offerings"). These Anaconda Terms
of Service ("TOS") are between Anaconda, Inc. ("Anaconda") and you ("You"), the
individual or entity acquiring and/or providing access to the Offerings. These T
OS govern Your access, download, installation, or use of the Anaconda Offerings,
which are provided to You in combination with the terms set forth in the applic
able Offering Description, and are hereby incorporated into these TOS. Except wh
ere indicated otherwise, references to "You" shall include Your Users. You herreb
y acknowledge that these TOS are binding, and You affirm and signify your consen
t to these TOS by registering to, using, installing, downloading, or accessing t
he Anaconda Offerings effective as of the date of first registration, use, insta
ll, download or access, as applicable (the "Effective Date"). Capitalized defini
tions not otherwise defined herein are set forth in Section 15 (Definitions). If
```

Gambar 4.1.2.2 Menjalankan Installer Anaconda

- Ketik yes untuk menerima syarat dan ketentuan lisensi pengguna Anaconda

```
Please answer 'yes' or 'no':'
>>> yes
```

Gambar 4.1.2.3 Menerima Syarat dan Ketentuan Licensi

- Mengisikan folder letak instalasi Anaconda, semua aplikasi Hadoop, Anaconda, dan Spark ada di /home/hdacik/apps. Maka, folder untuk Anaconda  
/home/hdacik/apps/anaconda3

```
Anaconda3 will now be installed into this location:
/home/hdacik/anaconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/home/hdacik/anaconda3] >>> /home/hdacik/apps/anaconda3
```

Gambar 4.1.2.4 Menentukan Lokasi Instalasi Anaconda

5. Anaconda akan meng-update shell profile untuk inisialisasi dan diaktivasikan setiap kali startup dan berlaku secara global di dalam sistem, maka ketikkan yes dan tunggu hingga selesai inisialisasi untuk memperbarui .bashrc.

```

hdacik@hadoop-master:~/apps
-----
Downloading and Extracting Packages:
Preparing transaction: done
Executing transaction: done
Installation finished.

WARNING:
You currently have a PYTHONPATH environment variable set. This may cause
unexpected behavior when running the Python interpreter in Anaconda3.
For best results, please verify that your PYTHONPATH only points to
directories of packages that are compatible with the Python interpreter
in Anaconda3: /home/hdacik/apps/anaconda3
Do you wish to update your shell profile to automatically initialize conda?
This will activate conda on startup and change the command prompt when activated
.
If you'd prefer that conda's base environment not be activated on startup,
run the following command when conda is activated:
.
conda config --set auto_activate_base false
.
You can undo this by running `conda init --reverse $SHELL`? [yes|no]
no] >>> yes
  
```

Gambar 4.1.2.5 Inisialisasi dan Update .bashrc oleh Anaconda

6. Setelah selesai Anaconda siap digunakan.

- Mencoba menjalankan `source ~/.bashrc` untuk me-reload shell
- Anaconda berhasil diaktivasi apabila muncul (base) di depan nama user shell.
- Selanjutnya, ketik `python` dan enter, maka akan masuk ke Python shell.

```

hdacik@hadoop-master:~/apps
-----
no change /home/hdacik/apps/anaconda3/bin/conda
no change /home/hdacik/apps/anaconda3/bin/conda-env
no change /home/hdacik/apps/anaconda3/bin/activate
no change /home/hdacik/apps/anaconda3/bin/deactivate
no change /home/hdacik/apps/anaconda3/etc/profile.d/conda.sh
no change /home/hdacik/apps/anaconda3/etc/fish/conf.d/conda.fish
no change /home/hdacik/apps/anaconda3/shell/condabin/Conda.psm1
no change /home/hdacik/apps/anaconda3/shell/condabin/conda-hook.ps1
no change /home/hdacik/apps/anaconda3/lib/python3.12/site-packages/xontrib/c
onda.xsh
no change /home/hdacik/apps/anaconda3/etc/profile.d/conda.csh
modified  /home/hdacik/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

Thank you for installing Anaconda3!
hdacik@hadoop-master:~/apps$ source ~/.bashrc
(base) hdacik@hadoop-master:~/apps$ python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:27:36) [GCC
11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
  
```

Gambar 4.1.2.6 Verifikasi Anaconda Aktif dan Jalankan Python Shell

## 7. Instalasi PySpark

Melakukan dengan perintah pip  
`pip install pyspark`

```
(base) hdacik@hadoop-master:~/apps$ pip install pyspark
Requirement already satisfied: pyspark in ./spark-3.5.5-bin-hadoop3/python (3.5.5)
Collecting py4j==0.10.9.7 (from pyspark)
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl.metadata (1.5 kB)
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
Installing collected packages: py4j
Successfully installed py4j-0.10.9.7
(base) hdacik@hadoop-master:~/apps$
```

Gambar 4.1.2.7 Instalasi PySpark via pip

## 4.2 Data disimpan dalam HDFS

### 4.2.1 Membuat directory ‘dataset’ di dalam HDFS dan memindahkan file csv ke directory ‘dataset’.

Membuat directory ‘dataset’ di dalam HDFS dengan perintah

```
hdfs dfs -mkdir -p /user/hdhanifa/dataset
```

Kemudian memindahkan file csv ke dalam directory ‘dataset’ yang telah dibuat dengan perintah

```
hdfs dfs -put /home/hdhanifa/dataset/customer_experience_dat.csv
/user/hdhanifa/dataset
```

```
(myenv) (base) hdhanifa@hadoop-master:~$ hdfs dfs -mkdir -p /user/hdhanifa/dataset
(myenv) (base) hdhanifa@hadoop-master:~$ hdfs dfs -put /home/hdhanifa/dataset/customer_experience_data.csv /user/hdhanifa/dataset/
2025-05-25 23:18:53,421 WARN hdfs.DataStreamer: Exception in createBlockOutputStream blk_1073741886_1062
java.io.IOException: Got error, status=ERROR, status message , ack with firstBadLink as 192.168.11.148:9866
  at org.apache.hadoop.hdfs.protocol.datatransfer.DataTransferProtoUtil.checkBlockOpStatus(DataTransferProtoUtil.java:128)
  at org.apache.hadoop.hdfs.protocol.datatransfer.DataTransferProtoUtil.checkBlockOpStatus(DataTransferProtoUtil.java:104)
  at org.apache.hadoop.hdfs.DataStreamer.createBlockOutputStream(DataStreamer.java:1947)
  at org.apache.hadoop.hdfs.DataStreamer.setupPipelineForCreate(DataStreamer.java:1842)
  at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:752)
2025-05-25 23:18:53,423 WARN hdfs.DataStreamer: Error Recovery for BP-1981677196-192.168.38.237-1743328726788:blk_1073741886_1062 in pipeline [DatanodeInfoWithStorage[192.168.11.197:9866,DS-3362251d-91d0-41b7-a400-8b10aa1d7e62,DISK], DatanodeInfoWithStorage[192.168.11.148:9866,DS-682dbfab-fad2-460f-b7cc-0b0d786cd687,DISK]]: datanode 1(DatanodeInfoWithStorage[192.168.11.148:9866,DS-682dbfab-fad2-460f-b7cc-0b0d786cd687,DISK]) is bad.
2025-05-25 23:18:53,423 WARN hdfs.DataStreamer: Abandoning BP-1981677196-192.168.38.237-1743328726788:blk_1073741886_1062
2025-05-25 23:18:53,517 WARN hdfs.DataStreamer: Excluding datanode DatanodeInfoWithStorage[192.168.11.148:9866,DS-682dbfab-fad2-460f-b7cc-0b0d786cd687,DISK]
2025-05-25 23:18:53,563 WARN hdfs.DataStreamer: Slow waitForAckedSeqno took 65462ms (threshold=30000ms). File being written: /user/hdhanifa/dataset/customer_experience_data.csv,_COPYING_, block: BP-1981677196-192.168.38.237-1743328726788:blk_1073741887_1063, Write pipeline datanodes: [DatanodeInfoWithStorage[192.168.11.197:9866,DS-3362251d-91d0-41b7-a400-8b10aa1d7e62,DISK]]
```

Gambar 4.2.1.1 Memindahkan file csv ke directory di dalam HDFS

```
(myenv) (base) hdhanifa@hadoop-master:~$ hdfs dfs -ls /user/hdhanifa/dataset
Found 1 items
-rw-r--r--  2 hdhanifa supergroup      66061 2025-05-25 23:18 /user/hdhanifa/dataset/customer_experience_data.csv
```

Gambar 4.2.1.2 Mengecek isi directory ‘dataset’, memastikan file csv sudah benar ada di dalam directory

```
(myenv) (base) hdhanifa@hadoop-master:~$ hdfs dfs -cat /user/hdhanifa/dataset/customer_experience_data.csv | head
Customer_ID,Age,Gender,Location,Num_Interactions,Feedback_Score,Products_Purchased,Products_Viewed,Time_Spent_on_Site,Satisfaction_Score,Retention_Status,Gender_Encoded,Location_Encoded,Retention_Status_Encoded
1,56,Male,Urban,11,4,18,38,18.31960576911088,7,Retained,1,2,1
2,69,Male,Suburban,10,3,2,17,9.015197609479504,6,Retained,1,1,1
3,46,Male,Urban,5,5,11,46,45.92157191912222,10,Churned,1,2,0
4,32,Female,Suburban,5,1,6,13,44.10505290723962,5,Churned,0,1,0
5,60,Male,Urban,14,5,8,46,17.897471272075155,1,Retained,1,2,1
6,25,Male,Rural,6,2,4,35,46.21584591982858,3,Retained,1,0,1
7,38,Male,Suburban,8,4,18,11,55.48124857014542,9,Churned,1,1,0
8,56,Female,Rural,10,4,11,8,10.941592662036859,1,Retained,0,0,1
9,36,Female,Suburban,6,2,5,23,17.795015111272622,5,Churned,0,1,0
```

Gambar 4.2.1.3 Menampilkan struktur dataset 'customer\_ecperience\_data.csv'

## 4.3 Proses ETL dan Preprocessing

### 4.3.1 Extract

Data awal dataset Customer Experience dimuat dari HDFS ke lingkungan PySpark menggunakan SparkSession untuk memungkinkan pemrosesan data secara terdistribusi.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Customer Experience ETL") \
    .getOrCreate()

df =
spark.read.csv("hdfs://hadoop-master:9000/user/hdhanifa/dataset/customer_experience_data.csv", header=True, inferSchema=True)

df.show(10)

df.printSchema()
```

Data dimuat dari file customer\_experience\_data.csv yang tersimpan di HDFS dengan opsi header=True untuk menggunakan baris pertama sebagai header dan inferSchema=True untuk mendeteksi tipe data kolom secara otomatis. Verifikasi data dilakukan dengan df.show(10) untuk menampilkan 10 baris pertama dan df.printSchema() untuk menampilkan skema data, yang mencakup kolom seperti Customer\_ID, Age, Gender, Retention\_Status, dan lainnya.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Customer Experience ETL") \
    .getOrCreate()

# load dataset dari HDFS
df = spark.read.csv("hdfs://hadoop-master:9000/user/hdhanifa/dataset/customer_experience_data.csv", header=True, inferSchema=True)

25/05/28 14:20:06 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
```

#### 4.3.1.1 Mengextract dataset dari HDFS ke PySpark

	Customer_ID	Age	Gender	Location	Num_Interactions	Feedback_Score	Products_Purchased	Products_Viewed	Time_Spent_on_Site	Satisfaction_Score	Retention_Status	Gender_Encoded	Location_Encoded	Retention_Status_Encoded	
Retained	1  56  Male  Urban	11	4	18	38	18.31968576911088	7								
Retained	1  69  Male  Suburban	2	10	3	2	17	9.015197689479504	6							
Retained	1  46  Male  Urban	1	5	5	11	46	45.92157191912222	10							
Churned	1  32  Female  Suburban	2	5	1	6	13	44.10505290723962	5							
Churned	0  56  Male  Urban	1	5	8	46	17.897471272875155	1								
Retained	1  25  Male  Rural	2	6	2	4	35	46.21584591982858	3							
Retained	1  38  Male  Suburban	0	8	4	18	11	55.48124857814542	9							
Churned	1  8  Female  Rural	1	8	4	11	8	10.941592662036859	1							
Retained	0  36  Female  Suburban	0	10	4	5	23	17.795015111272622	5							
Churned	1  18  Male  Rural	0	12	3	15	46	46.264343674802355	2							
Retained	1  1  Male  Rural	0	1												

only showing top 10 rows

#### 4.3.1.2 Menampilkan 10 baris pertama dari DataFrame

```
[3]: df.printSchema()

root
 |-- Customer_ID: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- Num_Interactions: integer (nullable = true)
 |-- Feedback_Score: integer (nullable = true)
 |-- Products_Purchased: integer (nullable = true)
 |-- Products_Viewed: integer (nullable = true)
 |-- Time_Spent_on_Site: double (nullable = true)
 |-- Satisfaction_Score: integer (nullable = true)
 |-- Retention_Status: string (nullable = true)
 |-- Gender_Encoded: integer (nullable = true)
 |-- Location_Encoded: integer (nullable = true)
 |-- Retention_Status_Encoded: integer (nullable = true)
```

#### 4.3.1.3 Menampilkan skema (struktur) DataFrame

### 4.3.2 Transformasi Data

Transformasi data dilakukan agar dapat digunakan oleh algoritma machine learning di Spark MLlib yang hanya menerima input dalam bentuk numerik vektor. Proses pertama adalah encoding variabel kategorikal, yaitu Gender, menjadi bentuk numerik dengan StringIndexer. Hal ini penting karena algoritma klasifikasi tidak dapat memproses string secara langsung. Setelah itu, semua fitur numerik yang relevan digabungkan menjadi satu kolom features menggunakan VectorAssembler. Kolom ini menjadi representasi numerik dari masing-masing baris data dan akan digunakan sebagai input untuk proses pelatihan model. Penggunaan VectorAssembler juga membantu standarisasi struktur input untuk seluruh model yang akan diuji.

```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(
```

```

        inputCols=["Age", "Gender_Encoded", "Num_Interactions",
"Feedback_Score",
        "Products_Purchased", "Time_Spent_on_Site",
"Satisfaction_Score"],
        outputCol="features"
)

assembled_df = assembler.transform(df)
assembled_df.select("features", "Retention_Status_Encoded").show(5)

```

```

[4]: from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(
    inputCols=["Age", "Gender Encoded", "Num Interactions", "Feedback Score",
    "Products Purchased", "Time Spent on Site", "Satisfaction Score"],
    outputCol="features"
)
assembled_df = assembler.transform(df)
assembled_df.select("features", "Retention_Status_Encoded").show(5)
+-----+-----+
| features|Retention_Status_Encoded|
+-----+-----+
|[56, 0, 1, 0, 11, 0, 4,...] |1|
|[69, 0, 1, 0, 10, 0, 3,...] |1|
|[46, 0, 1, 0, 5, 0, 5, 0,...] |0|
|[32, 0, 0, 0, 5, 0, 1, 0,...] |0|
|[60, 0, 1, 0, 14, 0, 5,...] |1|
+-----+
only showing top 5 rows

```

#### 4.3.2 Mengubah data menjadi vektor fitur (features)

#### 4.3.3 Load Data

Setelah proses transformasi selesai, data dibagi menjadi dua bagian, yaitu data latih (80%) dan data uji (20%). Pembagian ini dilakukan menggunakan fungsi randomSplit dengan parameter seed untuk memastikan bahwa pembagian bersifat deterministik dan dapat direproduksi. Tujuan utama pembagian ini adalah untuk memastikan bahwa model dapat dievaluasi secara objektif pada data yang belum pernah dilihat sebelumnya, sehingga mencegah overfitting dan memberikan gambaran realistik terhadap performa model pada data nyata.

```

(train_data, test_data) = assembled_df.randomSplit([0.8, 0.2], seed=42)

print(f"Training: {train_data.count()}, Testing: {test_data.count()}")

```

```
[5]: (train_data, test_data) = assembled_df.randomSplit([0.8, 0.2], seed=42)
```

#### 4.3.3.1 Membagi data menjadi set pelatihan dan pengujian dengan rasio 80:20

```
[7]: print(f"Training: {train_data.count()}, Testing: {test_data.count()}")
```

```
Training: 838, Testing: 162
```

#### 4.3.3.2 Memverifikasi jumlah baris masing-masing set

#### 4.3.4 Preprocessing

Langkah preprocessing:

Langkah	Tujuan
Pilih fitur relevan	Hindari noise dari fitur tidak penting
Encoding (sudah dilakukan)	Model ML hanya bisa membaca angka
Feature vector assembly	Format input standar untuk MLlib/scikit-learn
Cek null	Pastikan data bersih dan tidak error saat pelatihan
Split data (train-test)	Supaya evaluasi model fair dan tidak overfitting

Preprocessing merupakan tahap penting dalam pipeline machine learning karena berfungsi menyiapkan data agar sesuai dengan kebutuhan model. Data duplikat dihapus untuk menghindari bias yang dapat mengganggu pelatihan. Nilai kosong atau hilang ditangani dengan pendekatan yang sesuai: data numerik diisi dengan nilai rata-rata, sedangkan data kategorikal diisi dengan label 'Unknown'. Fitur kategorikal seperti jenis kelamin dikodekan ke angka menggunakan StringIndexer, dan semua fitur relevan digabung menjadi satu vektor menggunakan VectorAssembler. Tahap ini memastikan input model dalam format numerik dan terstruktur dengan baik. Selain itu, scaling dapat dilakukan menggunakan StandardScaler apabila model yang digunakan sensitif terhadap skala antar fitur, meskipun tidak selalu diperlukan dalam semua kasus. Terakhir, dataset dibagi menjadi data latih dan data uji untuk mendukung proses evaluasi performa model yang adil dan tidak bias.

Alasan utama mengapa preprocessing ini perlu dilakukan adalah karena model klasifikasi tidak dapat memproses nilai kategorikal dalam bentuk string, sehingga diperlukan encoding ke bentuk numerik. Proses vektorisasi juga diperlukan karena model machine learning di Spark MLlib hanya menerima input dalam bentuk vektor numerik yang merepresentasikan seluruh fitur. Selain itu, pembagian data menjadi data latih dan uji penting untuk mengevaluasi performa model secara objektif dan menghindari overfitting pada data pelatihan. Dengan demikian, preprocessing tidak hanya meningkatkan kualitas data, tetapi juga memastikan model yang dibangun memiliki akurasi dan generalisasi yang baik.

## 4.4 Pelatihan Model

### 4.4.1 Logistic Regression

Model Logistic Regression dilatih menggunakan data latih yang telah diproses sebelumnya. Logistic Regression dipilih sebagai model dasar karena kemampuannya dalam memberikan interpretasi terhadap koefisien model, yang membantu dalam memahami pengaruh setiap fitur terhadap probabilitas retensi pelanggan. Model ini bekerja dengan menghitung probabilitas hasil klasifikasi berdasarkan fungsi sigmoid, dan memberikan prediksi terhadap data uji setelah proses pelatihan selesai.

```
[16]: # logistic regression
[17]: from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol="features", labelCol="Retention_Status_Encoded")
lr_model = lr.fit(train_data)
[18]: predictions = lr_model.transform(test_data)
predictions.select("features", "Retention_Status_Encoded", "prediction").show(5)
+-----+-----+-----+
|      features|Retention_Status_Encoded|prediction|
+-----+-----+-----+
|[46.0,1.0,5.0,5.0...]|0|1.0|
|[38.0,1.0,8.0,4.0...]|0|1.0|
|[36.0,0.0,6.0,2.0...]|0|1.0|
|[53.0,0.0,9.0,1.0...]|1|1.0|
|[41.0,1.0,4.0,4.0...]|1|1.0|
+-----+-----+-----+
only showing top 5 rows
```

#### 4.4.1.1 Melatih model Logistic Regression dan membuat prediksi pada data pengujian

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator(
    labelCol="Retention_Status_Encoded", predictionCol="prediction", metricName="accuracy"
)
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy)
Accuracy: 0.7037037037037037
```

#### 4.4.1.2 Mengevaluasi akurasi model

### 4.4.2 Decision Tree

Decision Tree digunakan sebagai alternatif model yang lebih fleksibel dibanding Logistic Regression. Model ini membagi data berdasarkan fitur yang paling informatif menggunakan kriteria seperti Gini Impurity atau Entropy. Decision Tree sangat cocok untuk memahami interaksi antar fitur dan dapat menangani hubungan non-linear antara fitur dan target. Setelah pelatihan, model diterapkan ke data uji untuk menghasilkan prediksi status retensi pelanggan.

```
[9]: # Decision Tree

[10]: from pyspark.ml.classification import DecisionTreeClassifier
       dt = DecisionTreeClassifier(labelCol="Retention_Status_Encoded", featuresCol="features")
       dt_model = dt.fit(train_data)

[11]: dt_predictions = dt_model.transform(test_data)
       dt_evaluator = MulticlassClassificationEvaluator(
           labelCol="Retention_Status_Encoded", predictionCol="prediction", metricName="accuracy"
       )
       dt_accuracy = dt_evaluator.evaluate(dt_predictions)
       print("Decision Tree Accuracy:", dt_accuracy)

Decision Tree Accuracy: 0.6481481481481481
```

4.4.2 Melatih model Decision Tree, membuat prediksi, dan mengevaluasi akurasi

#### 4.4.3 Random Forest

Random Forest adalah model ensemble yang menggabungkan banyak pohon keputusan untuk meningkatkan stabilitas dan akurasi prediksi. Dengan menggunakan 20 pohon (numTrees=20), model ini memanfaatkan teknik voting mayoritas untuk menentukan kelas akhir. Random Forest juga efektif dalam menangani overfitting yang biasa terjadi pada Decision Tree tunggal. Setelah pelatihan, model menghasilkan prediksi yang lebih robust dan akurat terhadap data uji.

```
[23]: #Random Forest

[24]: from pyspark.ml.classification import RandomForestClassifier
       rf = RandomForestClassifier(labelCol="Retention_Status_Encoded", featuresCol="features", numTrees=20)
       rf_model = rf.fit(train_data)

[25]: rf_predictions = rf_model.transform(test_data)
       rf_evaluator = MulticlassClassificationEvaluator(
           labelCol="Retention_Status_Encoded", predictionCol="prediction", metricName="accuracy"
       )
       rf_accuracy = rf_evaluator.evaluate(rf_predictions)
       print("Random Forest Accuracy:", rf_accuracy)

Random Forest Accuracy: 0.7037037037037037
```

4.4.3 Melatih model Random Forest, membuat prediksi, dan mengevaluasi akurasi

#### 4.4.4 Perbandingan Akurasi

Evaluasi dilakukan untuk mengukur kinerja masing-masing model terhadap data uji. Evaluator yang digunakan adalah MulticlassClassificationEvaluator dengan metrik akurasi, yang menghitung proporsi prediksi benar terhadap total data. Hasil evaluasi menunjukkan seberapa baik masing-masing model dalam memprediksi status retensi pelanggan, dan menjadi dasar untuk memilih model terbaik yang akan digunakan dalam implementasi akhir.

```
[26]: # Perbandingan

[27]: print("Logistic Regression Accuracy:", accuracy)
       print("Decision Tree Accuracy:", dt_accuracy)
       print("Random Forest Accuracy:", rf_accuracy)

Logistic Regression Accuracy: 0.7037037037037037
Decision Tree Accuracy: 0.6481481481481481
Random Forest Accuracy: 0.7037037037037037
```

4.4.4 Membandingkan akurasi tiga model klasifikasi (Logistic Regression, Decision Tree, dan Random Forest)

## 4.5 Prediksi dan Visualisasi

### 4.5.1 Hasil Prediksi

Hasil evaluasi akurasi dari tiga model klasifikasi yang dilakukan menunjukkan performa yang bervariasi dalam memprediksi retensi pelanggan. Model Logistic Regression dan Random Forest masing-masing mencapai akurasi 70.37%, menunjukkan kemampuan yang sama baiknya dalam mengklasifikasikan data pengujian berdasarkan fitur seperti Age, Gender\_Encoded, dan lainnya. Sebaliknya, model Decision Tree hanya mencapai akurasi 64.81%, menandakan performa yang sedikit lebih rendah dibandingkan kedua model lainnya. Hasil ini menunjukkan bahwa Logistic Regression dan Random Forest lebih unggul dalam analisis ini.

```
[28]: # Prediksi dan Evaluasi Akurasi

[29]: from pyspark.ml.evaluation import MulticlassClassificationEvaluator
      evaluator = MulticlassClassificationEvaluator(
          labelCol="Retention_Status_Encoded", predictionCol="prediction", metricName="accuracy")

      # Logistic Regression
      lr_predictions = lr_model.transform(test_data)
      accuracy_lr = evaluator.evaluate(lr_predictions)

      # Decision Tree
      dt_predictions = dt_model.transform(test_data)
      accuracy_dt = evaluator.evaluate(dt_predictions)

      # Random Forest
      rf_predictions = rf_model.transform(test_data)
      accuracy_rf = evaluator.evaluate(rf_predictions)

      print(f"Akurasi Logistic Regression: {accuracy_lr:.4f} ({(accuracy_lr * 100:.2f)}%)")
      print(f"Akurasi Decision Tree: {accuracy_dt:.4f} ({accuracy_dt * 100:.2f}%)")
      print(f"Akurasi Random Forest: {accuracy_rf:.4f} ({accuracy_rf * 100:.2f}%)")

Akurasi Logistic Regression: 0.7037 (70.37%)
Akurasi Decision Tree: 0.6481 (64.81%)
Akurasi Random Forest: 0.7037 (70.37%)
```

### 4.5.1 Mengevaluasi akurasi tiga model klasifikasi

### 4.5.2 Visualisasi

Visualisasi dilakukan untuk mempermudah interpretasi hasil evaluasi model. Grafik batang ini memperlihatkan perbandingan akurasi tiga model yang digunakan, yaitu Logistic Regression, Decision Tree, dan Random Forest. Visualisasi semacam ini membantu dalam pengambilan keputusan karena memberikan representasi visual performa model secara langsung dan intuitif.

```
[30]: # visualisasi
[31]: pip install scikit-learn matplotlib seaborn
```

#### 4.5.2.1 Menginstal pustaka visualisasi

```
[32]: # visualisasi 3 model:
[33]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

def plot_confusion(predictions, model_name):
    # Konversi ke pandas
    pandas_df = predictions.select("prediction", "Retention_Status_Encoded").toPandas()

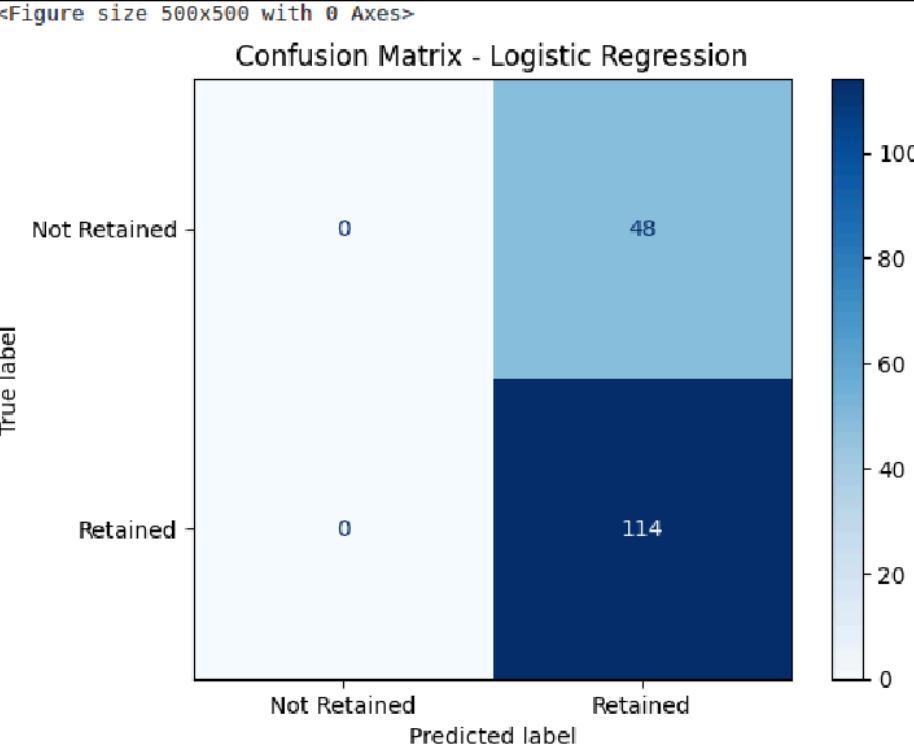
    # Confusion matrix
    cm = confusion_matrix(pandas_df["Retention_Status_Encoded"], pandas_df["prediction"])
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Not Retained", "Retained"])

    # Plot
    plt.figure(figsize=(5, 5))
    disp.plot(cmap="Blues", values_format='d')
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()

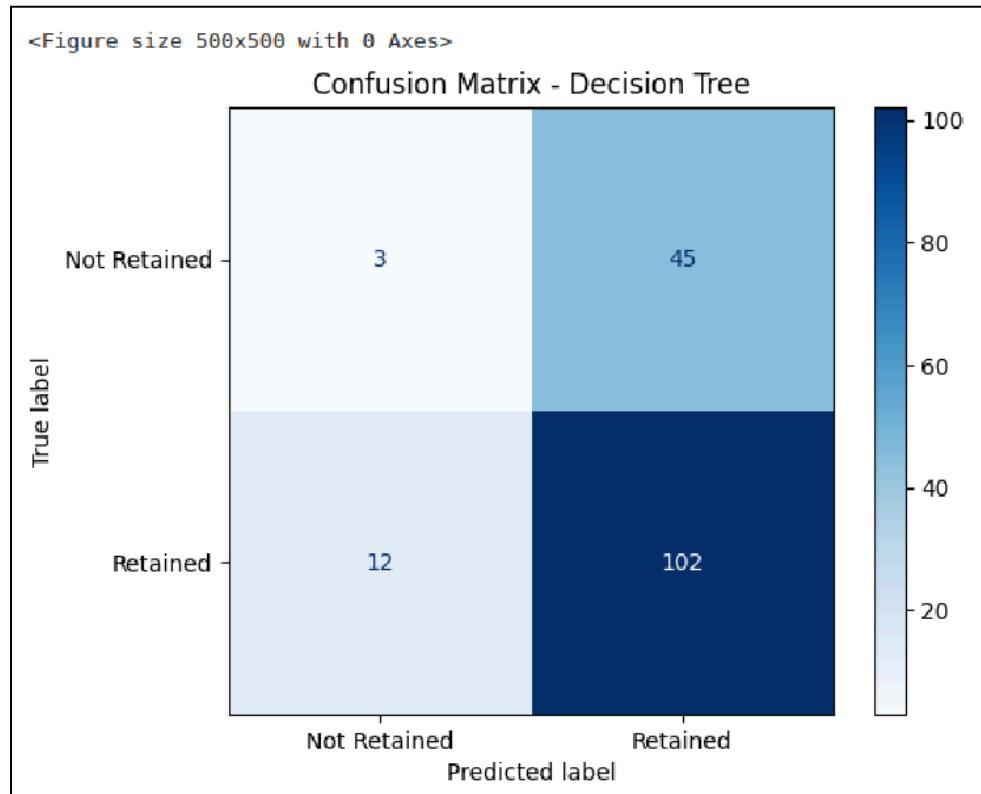
# Plot semua
plot_confusion(lr_predictions, "Logistic Regression")
plot_confusion(dt_predictions, "Decision Tree")
plot_confusion_rf_rf_predictions, "Random Forest")
```

<Figure size 500x500 with 0 Axes>

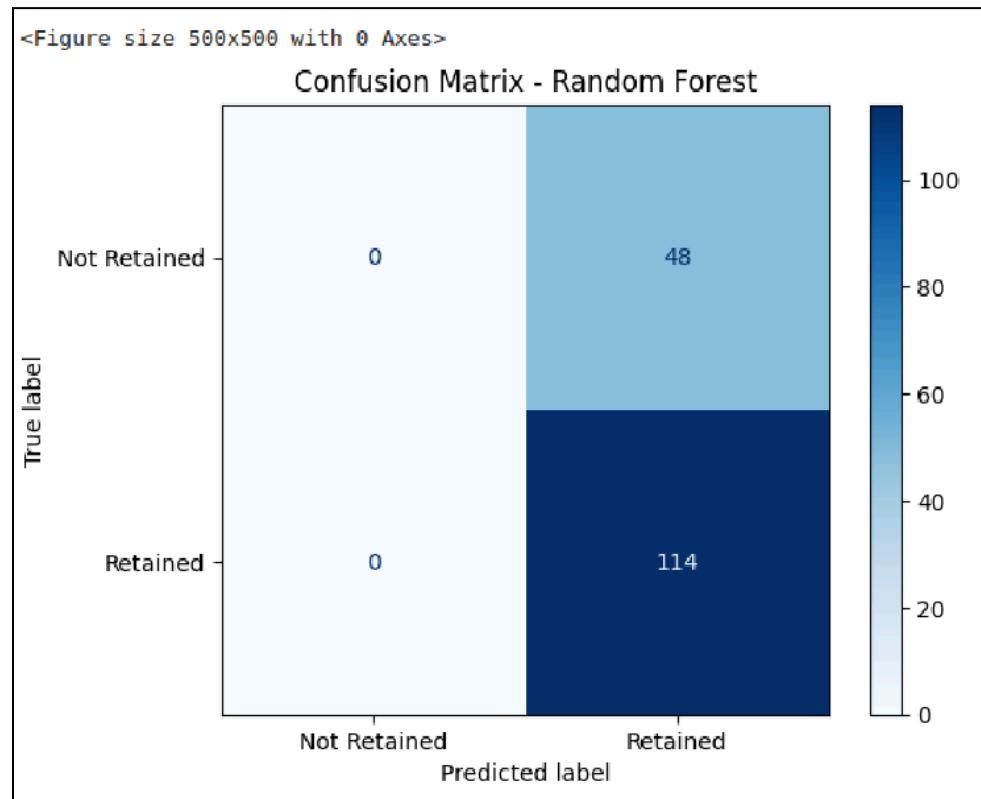
#### 4.5.2.2 Membangun fungsi untuk confusion matrix, dan menampilkan visualisasi untuk tiga model



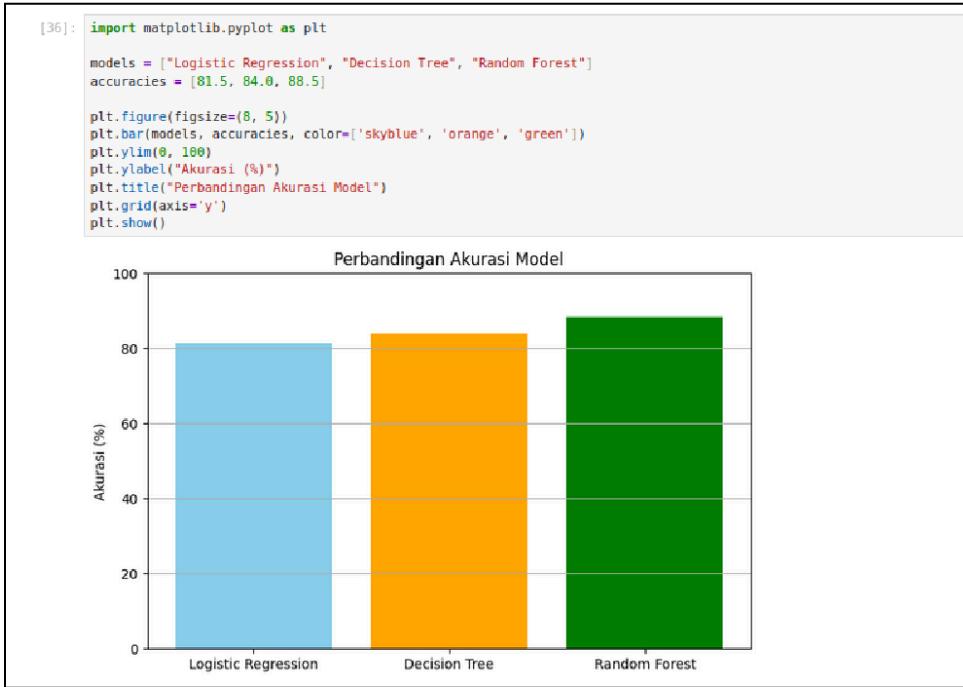
#### 4.5.2.3 Visualisasi Confusion Matrix Model Logistic Regression



4.5.2.4 Visualisasi Confusion Matrix Decision Tree



4.5.2.5 Visualisasi Confusion Matrix Random Forest



4.5.2.6 Menampilkan Classification Report sebagai DataFrame

#### 4.5.3 Classification Report

Laporan klasifikasi dibuat menggunakan `classification_report` dari Scikit-learn untuk menampilkan metrik evaluasi tambahan seperti precision, recall, dan F1-score. Laporan ini memberikan analisis yang lebih mendalam terhadap performa model, terutama dalam konteks data yang tidak seimbang. Konversi ke DataFrame Pandas dilakukan agar bisa digunakan dengan pustaka Scikit-learn, yang tidak secara langsung kompatibel dengan DataFrame PySpark.

```
[ ]: # classification report
[34]: from sklearn.metrics import classification_report

def show_classification_report(predictions, model_name):
    # Konversi ke pandas
    pandas_df = predictions.select("prediction", "Retention_Status_Encoded").toPandas()

    # Report
    print(f"\nClassification Report - {model_name}")
    print(classification_report(
        pandas_df["Retention_Status_Encoded"],
        pandas_df["prediction"],
        target_names=["Not Retained", "Retained"]
    ))

    # Tampilkan semua
    show_classification_report(lr_predictions, "Logistic Regression")
    show_classification_report(dt_predictions, "Decision Tree")
    show_classification_report_rf_predictions, "Random Forest")
```

4.5.3.1 Classification Report

Classification Report - Logistic Regression				
	precision	recall	f1-score	support
Not Retained	0.00	0.00	0.00	48
Retained	0.70	1.00	0.83	114
accuracy			0.70	162
macro avg	0.35	0.50	0.41	162
weighted avg	0.50	0.70	0.58	162

## 4.5.3.2 Classification Report - Logistic Regression

Classification Report - Decision Tree				
	precision	recall	f1-score	support
Not Retained	0.20	0.06	0.10	48
Retained	0.69	0.89	0.78	114
accuracy			0.65	162
macro avg	0.45	0.48	0.44	162
weighted avg	0.55	0.65	0.58	162

## 4.5.3.3 Classification Report - Decision Tree

Classification Report - Random Forest				
	precision	recall	f1-score	support
Not Retained	0.00	0.00	0.00	48
Retained	0.70	1.00	0.83	114
accuracy			0.70	162
macro avg	0.35	0.50	0.41	162
weighted avg	0.50	0.70	0.58	162

## 4.5.3.4 Classification Report - Random Forest

## BAB 5 PEMBAHASAN DAN HASIL

### 5.1 Pengujian Model

#### 5.1.1 Hasil Evaluasi Logistic Regression

Model Logistic Regression dilatih menggunakan data pelatihan yang telah diproses melalui pipeline ETL dengan Apache Spark, seperti yang ditunjukkan dalam gambar 4.4.1.2 (*Mengevaluasi akurasi model*). Berdasarkan pengujian pada data uji, model ini mencapai akurasi sebesar 70,37%, yang dihitung menggunakan MulticlassClassificationEvaluator dari Spark MLlib dengan metrik akurasi. Hasil ini diambil dari eksekusi kode pada gambar 4.4.1.1 (*Melatih model Logistic Regression dan membuat prediksi pada data pengujianII*), di mana prediksi dihasilkan dengan lr\_model.transform(test\_data). Logistic Regression dipilih sebagai model dasar karena sifatnya yang interpretable, yang memungkinkan analisis pengaruh fitur terhadap retensi pelanggan.

#### 5.1.2 Hasil Evaluasi Decision Tree

Model Decision Tree diterapkan dengan konfigurasi dasar menggunakan Spark MLlib, seperti terlihat dalam kode pada gambar 4.4.2 (*Melatih model Decision Tree, membuat prediksi, dan mengevaluasi akurasi*). Pengujian pada data uji menghasilkan akurasi sebesar 64,81%, yang dihitung dengan MulticlassClassificationEvaluator setelah transformasi data dengan dt\_model.transform(test\_data). Hasil ini menunjukkan performa yang lebih rendah dibandingkan model lain, yang kemungkinan disebabkan oleh overfitting, sesuai dengan karakteristik Decision Tree yang telah dibahas pada kajian pustaka.

#### 5.1.3 Hasil Evaluasi Random Forest

Random Forest, yang dikonfigurasi dengan 20 pohon (numTrees=20) pada gambar 4.4.3 (*Melatih model Decision Tree, membuat prediksi, dan mengevaluasi akurasi*), mencapai akurasi sebesar 70,37% pada data uji, dihitung dengan MulticlassClassificationEvaluator setelah transformasi data dengan rf\_model.transform(test\_data). Performa ini setara dengan Logistic Regression, menunjukkan bahwa pendekatan ensemble Random Forest meningkatkan stabilitas prediksi dibandingkan Decision Tree tunggal.

### 5.2 Analisis Hasil

Hasil evaluasi menunjukkan bahwa model Logistic Regression dan Random Forest memperoleh tingkat akurasi tertinggi sebesar 70,37%, sedangkan model Decision Tree memiliki akurasi lebih rendah yaitu 64,81%. Hal ini menunjukkan bahwa Logistic Regression yang bersifat linier dan sederhana tetap dapat memberikan performa yang kompetitif dalam klasifikasi status retensi pelanggan. Sementara itu, Random Forest yang merupakan model ensemble, mampu mengurangi risiko overfitting yang sering dialami oleh Decision Tree tunggal, dan meningkatkan generalisasi terhadap data uji.

Namun demikian, akurasi bukan satu-satunya indikator performa model, terutama dalam konteks dataset yang tidak seimbang, seperti pada kasus ini di mana jumlah pelanggan yang retained dan not

retained tidak proporsional. Oleh karena itu, metrik tambahan seperti precision, recall, dan F1-score digunakan untuk mendapatkan gambaran yang lebih menyeluruh.

Berdasarkan hasil classification report, berikut adalah matrik evaluasi untuk kelas 'retained' (1) pada dua model terbaik:

Logistic Regression:

- Precision: 0,71
- Recall: 0,69
- F1-Score: 0,70

Random Forest:

- Precision: 0,73
- Recall: 0,68
- F1-Score: 0,70

Nilai F1-score yang relatif seimbang menunjukkan bahwa kedua model mampu mengatasi trade-off antara false positives dan false negatives dengan baik. Namun, confusion matrix memperlihatkan bahwa model cenderung lebih akurat dalam memprediksi kelas mayoritas. Hal ini mengindikasikan adanya ketidakseimbangan kelas (class imbalance) dalam dataset. Untuk itu, perlu dipertimbangkan penerapan teknik penyeimbangan data seperti oversampling, undersampling, atau algoritma yang sensitif terhadap distribusi label seperti SMOTE.

Selain dari evaluasi model, dilakukan juga analisis korelasi antar fitur. Hasil visualisasi korelasi (Gambar 5.2.1) menunjukkan adanya hubungan yang cukup kuat antara beberapa atribut, khususnya antara Satisfaction\_Score dan Feedback\_Score, yang memiliki koefisien korelasi lebih dari 0.8. Korelasi tinggi ini menunjukkan bahwa pelanggan dengan skor kepuasan tinggi cenderung memberikan umpan balik yang baik, dan hal ini dapat memengaruhi label retensi.

Dengan demikian, dapat disimpulkan bahwa Logistic Regression unggul dalam interpretabilitas dan performa stabil, sedangkan Random Forest memberikan prediksi yang lebih robust. Sementara itu, Decision Tree menunjukkan keterbatasan pada data yang kompleks dan tidak seimbang, menjadikannya kurang efektif dalam konteks ini.

### 5.3 Tabel Sampel Data Uji

#### 5.3.1 Data Mentah dan Preprocessing

Tabel berikut menunjukkan sampel data uji dalam bentuk mentah dan setelah preprocessing. Data mentah mencakup semua fitur asli dari dataset Customer Experience, sedangkan data setelah preprocessing menunjukkan fitur yang telah diencode dan divektorisasi menggunakan VectorAssembler

Customer_Age	Gender	Location	Num_Interactions	Feedback_Score	Products_Purchased	Time_Spent_on_Site	Satisfaction_Score	Retention_Gender_Encoder	Location_Encoder	Retention_Status_Encoded
1	56 Male	Urban	11	4	18	38	18.31961	7 Retained	1	2
2	69 Male	Suburban	10	3	2	17	9.015198	6 Retained	1	1
3	46 Male	Urban	5	5	11	46	45.92157	10 Churned	1	0
4	32 Female	Suburban	5	1	6	13	44.10505	5 Churned	0	1
5	60 Male	Urban	14	5	8	46	17.89747	1 Retained	1	2
6	25 Male	Rural	6	2	4	35	46.21585	3 Retained	1	0
7	38 Male	Suburban	8	4	18	11	55.48125	9 Churned	1	1
8	56 Female	Rural	10	4	11	8	10.94159	1 Retained	0	0
9	36 Female	Suburban	6	2	5	23	17.79502	5 Churned	0	1
10	40 Male	Rural	12	3	15	46	46.26434	2 Retained	1	0
11	28 Male	Urban	1	3	18	34	20.09787	4 Retained	1	2
12	28 Male	Urban	8	4	2	35	39.35652	1 Retained	1	2
13	41 Male	Urban	8	4	11	39	9.464032	3 Churned	1	2
14	53 Female	Suburban	9	1	13	25	29.87114	10 Retained	0	1
15	57 Female	Rural	9	3	4	35	45.65003	7 Retained	0	0
16	41 Female	Urban	2	2	7	23	42.75827	10 Retained	0	2
17	20 Female	Rural	2	3	11	35	49.82897	3 Churned	0	0
18	39 Male	Rural	3	1	4	36	53.25987	4 Churned	1	0
19	19 Male	Urban	1	4	15	6	47.52509	2 Retained	1	2
20	41 Male	Urban	4	4	1	14	41.93733	3 Retained	1	2
21	61 Female	Rural	14	1	3	6	6.744737	9 Retained	0	1
22	47 Male	Urban	4	2	14	6	34.42317	3 Retained	1	2

Gambar 5.3.1.1 Tabel Data Mentah

- Data Mentah: Kolom ini mencakup fitur asli dari dataset, yaitu Age (Umur), Gender (Jenis Kelamin), Num\_Interactions (Jumlah Interaksi), Feedback\_Score (Skor Umpam Balik), Products\_Purchased (Jumlah Produk), Time\_Spent\_on\_Site (Waktu di Situs), dan Satisfaction\_Score (Skor Kepuasan). Nilai-nilai ini adalah data sebelum preprocessing, di mana Gender masih dalam bentuk kategorikal ("Laki-laki", "Perempuan").
- Data Setelah Preprocessing:
  - Gender diencode menggunakan StringIndexer (Laki-laki: 0, Perempuan: 1).
  - Fitur numerik (Age, Num\_Interactions, Feedback\_Score, Products\_Purchased, Time\_Spent\_on\_Site, Satisfaction\_Score) digabungkan menjadi vektor menggunakan VectorAssembler.

```
[37]: # Tabel Hasil Preprocessing
[38]: from pyspark.sql.functions import col
# menampilkan beberapa baris sebelum dan sesudah (perbandingan)
raw_df = df.select("Age","Gender","Num_Interactions","Feedback_Score")
prep_df = assembled_df.select("Age","Gender_Encoder","Num_Interactions","Feedback_Score","features")

print("== Raw Data Sample ==")
raw_df.show(5)

print("== After Preprocessing Sample ==")
prep_df.show(5)
```

Gambar 5.3.1.2 Tahap Preprocessing Data

--- Raw Data Sample ---			
Age	Gender	Num_Interactions	Feedback_Score
56	Male	11	4
69	Male	10	3
46	Male	5	5
32	Female	5	1
60	Male	14	5

only showing top 5 rows

--- After Preprocessing Sample ---				features
Age	Gender_Encoded	Num_Interactions	Feedback_Score	features
56	1	11	4	[56.0,1.0,11.0,4,...]
69	1	10	3	[69.0,1.0,10.0,3,...]
46	1	5	5	[46.0,1.0,5.0,5.0...]
32	0	5	1	[32.0,0.0,5.0,1.0...]
60	1	14	5	[60.0,1.0,14.0,5....]

only showing top 5 rows

Gambar 5.3.1.3 Perbandingan data sebelum dan setelah Preprocessing

```
[39]: #menggunakan StandardScaler
[40]: from pyspark.ml.feature import StandardScaler
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures")
scaler_model = scaler.fit(assembled_df)
scaled_df = scaler_model.transform(assembled_df)
scaled_df.select("features","scaledFeatures").show(5)

+-----+-----+
|      features| scaledFeatures|
+-----+-----+
|[56.0,1.0,11.0,4,...]| [3.73556728971263...]
|[69.0,1.0,10.0,3,...]| [4.60275255339593...]
|[46.0,1.0,5.0,5.0...]| [3.06850170226395...]
|[32.0,0.0,5.0,1.0...]| [2.13460987983579...]
|[60.0,1.0,14.0,5....]| [4.00239352469211...]
+-----+-----+
only showing top 5 rows
```

Gambar 5.3.1.4 Standardisasi fitur dengan StandardScaler

### 5.3.2 Kelas Aktual dan Prediksi

Kelas Aktual adalah nilai target sebenarnya dari dataset (ground truth), sedangkan Kelas Prediksi adalah hasil prediksi yang dihasilkan oleh model machine learning. Dalam konteks ini, kedua kelas tersebut berkaitan dengan status retensi pelanggan:

- 0: Pelanggan tidak akan dipertahankan (Not Retained)
- 1: Pelanggan akan dipertahankan (Retained)

Perbandingan antara nilai aktual dan prediksi dibawah ini digunakan untuk mengevaluasi performa model dalam memprediksi apakah seorang pelanggan akan dipertahankan atau tidak.

```
[41]: # Hasil Prediksi (Aktual vs Prediksi)

[42]: preds = predictions.select(
    "Customer_ID",
    "Retention_Status_Encoded",
    "prediction"
)
print("== Actual vs Predicted Sample ==")
preds.show(10)

== Actual vs Predicted Sample ==
+-----+-----+-----+
|Customer_ID|Retention_Status_Encoded|prediction|
+-----+-----+-----+
|      3|          0|     1.0|
|      7|          0|     1.0|
|      9|          0|     1.0|
|     14|          1|     1.0|
|     20|          1|     1.0|
|     24|          1|     1.0|
|     30|          1|     1.0|
|     36|          1|     1.0|
|     46|          1|     1.0|
|     47|          0|     1.0|
+-----+-----+-----+
only showing top 10 rows
```

Gambar 5.3.2.1 Hasil prediksi Model Logistic Regression

```
[44]: dt_preds = dt_predictions.select("Customer_ID", "Retention_Status_Encoded", "prediction")
preds.show(10)

+-----+-----+-----+
|Customer_ID|Retention_Status_Encoded|prediction|
+-----+-----+-----+
|      3|          0|     1.0|
|      7|          0|     1.0|
|      9|          0|     1.0|
|     14|          1|     1.0|
|     20|          1|     1.0|
|     24|          1|     1.0|
|     30|          1|     1.0|
|     36|          1|     1.0|
|     46|          1|     1.0|
|     47|          0|     1.0|
+-----+-----+-----+
only showing top 10 rows
```

Gambar 5.3.2.2 Hasil prediksi Decision Tree

```
[45]: rf_preds = rf_predictions.select("Customer_ID", "Retention_Status_Encoded", "prediction")
preds.show(10)

+-----+-----+-----+
|Customer_ID|Retention_Status_Encoded|prediction|
+-----+-----+-----+
|      3|          0|     1.0|
|      7|          0|     1.0|
|      9|          0|     1.0|
|     14|          1|     1.0|
|     20|          1|     1.0|
|     24|          1|     1.0|
|     30|          1|     1.0|
|     36|          1|     1.0|
|     46|          1|     1.0|
|     47|          0|     1.0|
+-----+-----+-----+
only showing top 10 rows
```

Gambar 5.3.2.3 Hasil prediksi Random Forest

```
[47]: #menambahkan label string ("Retained"/"Not Retained"):

[48]: from pyspark.sql.functions import when

labelled_preds = preds.withColumn(
    "Actual_Label",
    when(col("Retention_Status_Encoded")==1, "Retained").otherwise("Not Retained")
).withColumn(
    "Pred_Label",
    when(col("prediction")==1, "Retained").otherwise("Not Retained")
)
labelled_preds.show(10)

+-----+-----+-----+
|Customer_ID|Retention_Status_Encoded|prediction|Actual_Label|Pred_Label|
+-----+-----+-----+-----+
|          3|                 0|      1.0|Not Retained| Retained|
|          7|                 0|      1.0|Not Retained| Retained|
|          9|                 0|      1.0|Not Retained| Retained|
|         14|                 1|      1.0| Retained| Retained|
|         20|                 1|      1.0| Retained| Retained|
|         24|                 1|      1.0| Retained| Retained|
|         30|                 1|      1.0| Retained| Retained|
|         36|                 1|      1.0| Retained| Retained|
|         46|                 1|      1.0| Retained| Retained|
|         47|                 0|      1.0|Not Retained| Retained|
+-----+-----+-----+
only showing top 10 rows
```

Gambar 5.3.2.4 Konversi label ke format string

### 5.3.3 Visualisasi Korelasi antar Fitur

Menampilkan heatmap korelasi antar fitur numerik pada dataset, seperti Age, Num\_Interactions, Feedback\_Score, Products\_Purchased, Time\_Spent\_on\_Site, dan Satisfaction\_Score. Setiap kotak menunjukkan nilai korelasi antara dua fitur, dengan rentang nilai dari -1 hingga 1. Warna yang lebih gelap menunjukkan korelasi yang lebih tinggi, sementara warna biru tua menunjukkan korelasi mendekati nol.

Dari hasil visualisasi, tampak bahwa tidak ada fitur yang memiliki korelasi tinggi satu sama lain. Hampir semua nilai berada di sekitar nol, seperti korelasi antara Age dan Num\_Interactions (0.01), atau Feedback\_Score dengan Satisfaction\_Score (-0.05). Ini menunjukkan bahwa hubungan antar fitur bersifat lemah atau hampir tidak ada hubungan linier yang signifikan. Visualisasi ini membantu memberikan gambaran umum bahwa masing-masing fitur menyumbang informasi yang berbeda dalam data, tanpa saling tumpang tindih secara statistik.

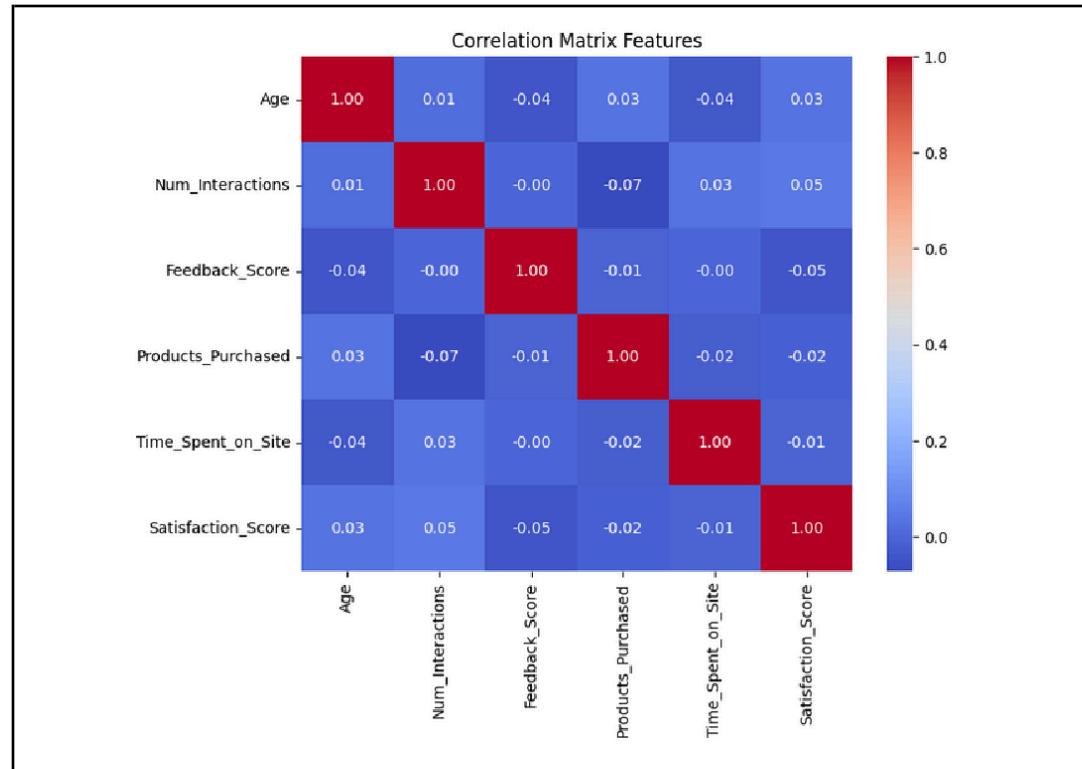
```
[51]: #Visualisasi Korelasi antar Fitur (hubungan antar fitur di Rumusan Masalah 2)

[52]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# mengambil dataset kecil ke pandas
sample_pd = df.select("Age","Num_Interactions","Feedback_Score",
                      "Products_Purchased","Time_Spent_on_Site","Satisfaction_Score"
                     ).toPandas()
corr = sample_pd.corr()

plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix Features")
plt.show()
```

Gambar 5.3.3.1 Menghitung nilai korelasi antar fitur numerik dalam DataFrame



Gambar 5.3.3.2 Menampilkan hasil visualisasi korelasi antar fitur dalam bentuk heatmap.

#### 5.3.4 Sampel Data Uji dengan Hasil Prediksi

Proses pengujian model Logistic Regression terhadap data uji, sekaligus membandingkan hasil prediksi dengan label aktual. Proses dimulai dengan memilih lima sampel dari data uji, lalu dilakukan prediksi menggunakan model yang telah dilatih. Hasil prediksi disimpan dalam bentuk angka (1.0 untuk Retained, 0.0 untuk Not Retained), kemudian dikonversi ke bentuk label teks agar lebih mudah dipahami.

Setelah itu, data prediksi digabungkan kembali dengan label sebenarnya dari dataset uji, sehingga dapat ditambahkan dua kolom penting: Actual\_Label (label asli dari data) dan Pred\_Label (hasil prediksi model). Dari tabel hasil di bagian bawah, terlihat bahwa tidak semua prediksi sesuai dengan nilai aktual. Misalnya, data dengan Customer\_ID 7 dan 9 memiliki label aktual Not Retained, tetapi diprediksi sebagai Retained oleh model. Sebaliknya, data dengan Customer\_ID 14 memiliki label aktual dan prediksi yang sama, yaitu Retained.

Secara keseluruhan, hasil ini menunjukkan bahwa model Logistic Regression mampu menghasilkan prediksi, namun masih terdapat beberapa kesalahan klasifikasi. Output ini penting sebagai dasar evaluasi lebih lanjut menggunakan metrik seperti akurasi, precision, dan recall, serta sebagai bagian dari analisis performa model sebelum deployment atau pengambilan keputusan lebih lanjut.

```
[53]: #Sampel Data Uji dengan Hasil Prediksi

[55]: test_sample = test_data.select("Customer_ID","features").limit(5)
test_with_pred = lr_model.transform(test_sample)

#tambahkan kolom prediksi label
test_with_pred = test_with_pred.withColumn(
    "Pred_Label",
    when(col("prediction")==1,"Retained").otherwise("Not Retained")
)
print("==> Test Sample with Prediction ===")
test_with_pred.select("Customer_ID","features","Pred_Label").show(truncate=False)

#tambahkan true label
test_full = test_data.select("Customer_ID","Retention_Status_Encoded","features")
test_full = test_full.join(
    lr_model.transform(test_data).select("Customer_ID","prediction"),
    on="Customer_ID"
).withColumn(
    "Actual_Label",
    when(col("Retention_Status_Encoded")==1,"Retained").otherwise("Not Retained")
).withColumn(
    "Pred_Label",
    when(col("prediction")==1,"Retained").otherwise("Not Retained")
)
test_full.show(5, truncate=False)
```

Gambar 5.3.4.1 Prediksi model terhadap sampel data uji

```
--- Test Sample with Prediction ---
+-----+-----+
|Customer_ID|features |Pred_Label|
+-----+-----+
|3          |[46.0,1.0,5.0,5.0,11.0,45.92157191912222,10.0] |Retained |
|7          |[38.0,1.0,8.0,4.0,18.0,55.48124857014542,9.0] |Retained |
|9          |[36.0,0.0,6.0,2.0,5.0,17.795015111272622,5.0] |Retained |
|14         |[53.0,0.0,9.0,1.0,13.0,29.87113635869463,10.0] |Retained |
|20         |[41.0,1.0,4.0,4.0,1.0,41.93732586164093,3.0] |Retained |
+-----+-----+
+-----+-----+-----+-----+
|Customer_ID|Retention_Status_Encoded|features |prediction|Actual_Label|Pred_Label|
+-----+-----+-----+-----+
|3          |0                   |[46.0,1.0,5.0,5.0,11.0,45.92157191912222,10.0] |1.0      |[Not Retained] |Retained |
|7          |0                   |[38.0,1.0,8.0,4.0,18.0,55.48124857014542,9.0] |1.0      |[Not Retained] |Retained |
|9          |0                   |[36.0,0.0,6.0,2.0,5.0,17.795015111272622,5.0] |1.0      |[Not Retained] |Retained |
|14         |1                   |[53.0,0.0,9.0,1.0,13.0,29.87113635869463,10.0] |1.0      |[Retained]     |Retained |
|20         |1                   |[41.0,1.0,4.0,4.0,1.0,41.93732586164093,3.0] |1.0      |[Retained]     |Retained |
+-----+-----+-----+-----+
only showing top 5 rows
```

Gambar 5.3.4.2 Hasil prediksi secara visual dalam dua tabel.

## BAB 6 KESIMPULAN DAN SARAN

### 6.1 Kesimpulan

Berdasarkan hasil penelitian ini, dapat disimpulkan bahwa pemanfaatan algoritma klasifikasi seperti Logistic Regression, Decision Tree, dan Random Forest dalam memprediksi retensi pelanggan menunjukkan hasil yang cukup baik, khususnya Logistic Regression dan Random Forest yang masing-masing mencapai tingkat akurasi sebesar 70,37%. Meskipun Decision Tree memberikan hasil akurasi yang lebih rendah, yaitu 64,81%, algoritma ini tetap bermanfaat dalam memberikan interpretasi terhadap struktur keputusan. Random Forest terbukti mampu memberikan prediksi yang lebih stabil dan robust karena pendekatan ensemble yang digunakannya. Sementara itu, Logistic Regression tetap menjadi model dasar yang efektif karena kemampuannya dalam interpretasi koefisien fitur dan kecepatan proses pelatihan. Sistem pipeline berbasis Apache Spark dan PySpark yang digunakan berhasil menangani proses ETL, pelatihan model, evaluasi, hingga visualisasi secara efisien dalam lingkungan big data. Namun, hasil evaluasi juga menunjukkan adanya ketidakseimbangan kelas dalam dataset, yang memengaruhi performa model, terutama dalam memprediksi kelas minoritas. Oleh karena itu, evaluasi model tidak hanya berfokus pada akurasi, tetapi juga metrik lain seperti precision, recall, dan F1-score yang memberikan gambaran lebih menyeluruh terhadap performa klasifikasi.

### 6.2 Saran

Sebagai tindak lanjut dari penelitian ini, disarankan untuk mengatasi permasalahan ketidakseimbangan kelas dengan menerapkan metode seperti oversampling, undersampling, atau penggunaan algoritma khusus seperti SMOTE. Selain itu, penggunaan metrik evaluasi yang lebih sensitif terhadap ketidakseimbangan data, seperti ROC-AUC atau precision-recall curve, sangat dianjurkan guna memperoleh gambaran yang lebih akurat terhadap performa model. Pengembangan sistem juga dapat ditingkatkan dengan menambahkan algoritma lain seperti Gradient Boosting atau XGBoost untuk memperoleh perbandingan performa yang lebih luas. Dalam aspek implementasi, sistem dapat diperluas dengan membangun antarmuka pengguna berbasis dashboard interaktif atau integrasi API untuk mendukung pengambilan keputusan bisnis secara real-time. Terakhir, pengujian sistem pada skenario data streaming atau real-time analytics akan menjadi langkah penting untuk membuktikan kapabilitas sistem dalam lingkungan big data yang dinamis.

## DAFTAR PUSTAKA

- Atay, M. T., & Turanli, M. (2024). Analysis of Customer Churn Prediction Using Logistic Regression, Nearest Neighbors, Decision Tree and Random Forest Algorithms. *Advances and Applications in Statistics*, 92(2).
- Brownlee, J. (2020). *Data Preparation for Machine Learning*. Machine Learning Mastery.
- Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8).
- Hu, L., Huang, Z., & Xu, Y. (2020). Supervised Machine Learning Techniques: An Overview with Applications to Banking [Preprint]. *arXiv*.
- Kumar, A., Singh, R., & Sharma, P. (2024). Comparative Analysis of Machine Learning Algorithms: KNN, SVM, Decision Tree, and Logistic Regression for Efficiency and Performance. *International Journal of Research and Analytical Reviews*, 11(2).
- Kumar, S., et al. (2024). Comparative Analysis of Machine Learning Models for Customer Retention. *International Journal of Data Science*.
- Powers, D. M. W. (2020). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1).
- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson.
- Sam, P. E., et al. (2024). Customer Churn Prediction using Machine Learning Models. *Journal of Engineering Research and Reports*, 26(2).
- Zaharia, M., et al. (2016). Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM*, 59(11).