Modul Praktikum:

Pra-Pemrosesan Data dengan PySpark di Google Colab

Modul ini akan memandu Anda melalui langkah-langkah praktis untuk melakukan pra-pemrosesan data pada dataset Big Data menggunakan Apache Spark (PySpark).

Langkah 0: Persiapan Lingkungan di Google Colab

Sebelum memulai, kita perlu menginstal PySpark dan beberapa pustaka pendukung di lingkungan Colab.

```
# Instalasi PySpark dan pustaka findspark
!pip install pyspark findspark
```

Setelah instalasi selesai, kita perlu menginisialisasi SparkSession, yang merupakan titik masuk untuk memprogram Spark dengan DataFrame API.

```
import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

# Membuat SparkSession
spark = SparkSession.builder \
    .master("local[*]") \
    .appName("PraktikumPreprocessing") \
    .getOrCreate()

print("SparkSession berhasil dibuat!")
```

Langkah 1: Membuat Dataset Sampel

Untuk mempermudah praktikum, kita akan membuat DataFrame Spark secara manual. Dataset ini sengaja dibuat "kotor" agar kita bisa mempraktikkan semua teknik pra-pemrosesan.

Dataset ini berisi data fiktif pendaftaran pelanggan.

```
# Data sampel yang "kotor"
data_kotor = [
   (1, 'Budi Susanto', 25, 5500000, 'L', '2022-01-15', 'Jakarta', 'Transaksi berhasil, barang bagus'),
   (2, 'Ani Lestari', None, 8000000, 'P', '2022-02-20', 'Bandung', 'Pengiriman cepat dan barang sesuai'),
   (3, 'Candra Wijaya', 35, 12000000, 'L', '2022-01-18', 'Surabaya', 'Sangat puas dengan pelayanannya'),
   (4, 'Dewi Anggraini', 22, 4800000, 'P', '2022-03-10', 'JKT', 'Barang diterima dalam kondisi baik'),
   (5, 'Eka Prasetyo', 45, 15000000, 'L', '2022-04-01', 'Jakarta', 'Transaksi gagal, mohon diperiksa'),
   (6, 'Budi Susanto', 25, 5500000, 'L', '2022-01-15', 'Jakarta', 'Transaksi berhasil, barang bagus'), # Data duplikat
   (7, 'Fina Rahmawati', 29, 9500000, 'P', '2022-05-12', 'Bandung', None), # Missing value di ulasan
   (8, 'Galih Nugroho', 31, -7500000, 'L', '2022-06-25', 'Surabaya', 'Barang oke'), # Gaji tidak valid (noisy)
   (9, 'Hesti Wulandari', 55, 25000000, 'P', '2022-07-30', 'Jakarta', 'Pelayanan ramah dan cepat'),
   (10, 'Indra Maulana', 150, 6200000, 'L', '2022-08-05', 'Medan', 'Produk original') # Usia tidak valid (outlier)
]

# Mendefinisikan skema DataFrame
skema = StructType([
   StructField("id_pelanggan", IntegerType(), True),
   StructField("nama", StringType(), True),
   StructField("usia", IntegerType(), True),
   StructField("gaji", IntegerType(), True),
   StructField("jenis_kelamin", StringType(), True),
   StructField("tgl_registrasi", StringType(), True),
   StructField("kota", StringType(), True),
   StructField("ulasan", StringType(), True)
])

# Membuat DataFrame
df = spark.createDataFrame(data=data_kotor, schema=skema)
```

```
# Menampilkan data awal
print("Dataset Awal:")
df.show(truncate=False)
df.printSchema()
```

BAGIAN 1: DATA CLEANING

1.1 Menangani Missing Values

Pertama, kita identifikasi di mana saja ada nilai yang hilang (null atau None).

```
# Menghitung jumlah missing values di setiap kolom
df.select([count(when(isnull(c), c)).alias(c) for c in df.columns]).show()
```

Ada nilai yang hilang di kolom usia dan ulasan. Mari kita coba beberapa strategi:

```
# Strategi 1: Menghapus baris yang memiliki nilai null
df_dropped = df.na.drop()
print("Data setelah baris null dihapus:")
df_dropped.show()

# Strategi 2: Mengisi nilai null (Imputasi)
# Untuk kolom numerik 'usia', kita isi dengan rata-rata usia
mean_usia = df.select(mean(df['usia'])).collect()[0][0]
df_filled = df.na.fill({'usia': int(mean_usia)})

# Untuk kolom string 'ulasan', kita isi dengan teks default
df_filled = df_filled.na.fill({'ulasan': 'Tidak ada ulasan'})

print("Data setelah imputasi:")
df_filled.show(truncate=False)

# Kita akan menggunakan df_filled untuk langkah selanjutnya
df_bersih = df_filled
```

1.2 Menangani Data Duplikat

Dataset kita memiliki satu baris duplikat. Mari kita identifikasi dan hapus.

```
# Menghitung jumlah total baris sebelum penghapusan
print(f"Jumlah baris sebelum hapus duplikat: {df_bersih.count()}")

# Menghapus baris yang duplikat
df_bersih = df_bersih.dropDuplicates()

# Menghitung jumlah total baris setelah penghapusan
print(f"Jumlah baris setelah hapus duplikat: {df_bersih.count()}")
df_bersih.show()
```

1.3 Memperbaiki Data Tidak Konsisten & Berisik (Noisy)

- **Inkonsistensi:** Kolom kota memiliki 'JKT' yang seharusnya 'Jakarta'.
- **Noisy Data:** Kolom gaji memiliki nilai negatif dan kolom usia memiliki nilai tidak wajar (150).

```
# Standarisasi kolom 'kota'
df_bersih = df_bersih.withColumn("kota", \
   when(df_bersih["kota"] == "JKT", "Jakarta") \
   .otherwise(df_bersih["kota"]))

# Memperbaiki data berisik
# Ganti gaji negatif menjadi nilai absolutnya
df_bersih = df_bersih.withColumn("gaji", abs(df_bersih["gaji"]))

# Hapus baris dengan usia yang tidak realistis (misal > 100)
df_bersih = df_bersih.filter(df_bersih["usia"] <= 100)

print("Data setelah perbaikan inkonsistensi dan noise:")
df_bersih.show()
```

BAGIAN 2: DATA TRANSFORMASI

2.1 Standarisasi & Normalisasi

Teknik ini memerlukan fitur numerik dalam format vektor. Kita akan menggunakan VectorAssembler, StandardScaler, dan MinMaxScaler.

```
from pyspark.ml.feature import VectorAssembler, StandardScaler, MinMaxScaler

# Gabungkan fitur numerik ('usia', 'gaji') ke dalam satu kolom vektor
assembler = VectorAssembler(inputCols=["usia", "gaji"], outputCol="fitur_numerik")
df_vector = assembler.transform(df_bersih)

# 1. Standarisasi (Z-score Normalization)
scaler_std = StandardScaler(inputCol="fitur_numerik", outputCol="fitur_standar",
withStd=True, withMean=True)
scaler_std_model = scaler_std.fit(df_vector)
df_standar = scaler_std_model.transform(df_vector)

print("Data setelah Standarisasi:")
df_standar.select("usia", "gaji", "fitur_standar").show(truncate=False)

# 2. Normalisasi (Min-Max Scaling)
scaler_minmax = MinMaxScaler(inputCol="fitur_numerik", outputCol="fitur_normal")
scaler_minmax_model = scaler_minmax.fit(df_vector)
df_normal = scaler_minmax_model.transform(df_vector)

print("Data setelah Normalisasi:")
df_normal.select("usia", "gaji", "fitur_normal").show(truncate=False)

# Kita akan gunakan df_bersih untuk langkah selanjutnya
```

2.2 Agregasi Data

Mari kita kelompokkan data berdasarkan kota dan hitung rata-rata gaji serta jumlah pelanggan.

```
# Agregasi data
df_agregat = df_bersih.groupBy("kota") \
   .agg(
      count("id_pelanggan").alias("jumlah_pelanggan"),
      avg("gaji").alias("rata_rata_gaji")
   )

print("Hasil Agregasi per Kota:")
df_agregat.show()
```

2.3 Diskretisasi (Binning)

Kita akan mengubah fitur kontinu usia menjadi kategori kelompok_usia.

```
from pyspark.ml.feature import Bucketizer

# Tentukan batasan untuk setiap bin/kelompok
# Bins: <20 (Remaja), 20-40 (Dewasa), >40 (Paruh Baya)
splits = [0, 20, 40, float('Inf')]

bucketizer = Bucketizer(splits=splits, inputCol="usia", outputCol="kelompok_usia")
df_binned = bucketizer.transform(df_bersih)

print("Data setelah Diskretisasi Usia:")
df_binned.select("usia", "kelompok_usia").show()
```

BAGIAN 3: FEATURE ENGINEERING

3.1 Ekstraksi Fitur dari Tanggal & Waktu

Kita akan membuat fitur-fitur baru dari kolom tgl_registrasi.

```
# Ubah tipe data kolom tanggal dari string ke timestamp
df_engineered = df_bersih.withColumn("timestamp_reg", to_timestamp("tgl_registrasi",
"yyyy-MM-dd"))

# Ekstraksi fitur-fitur baru
df_engineered = df_engineered.withColumn("tahun_reg", year("timestamp_reg"))
df_engineered = df_engineered.withColumn("bulan_reg", month("timestamp_reg"))
df_engineered = df_engineered.withColumn("hari_reg", dayofmonth("timestamp_reg"))
df_engineered = df_engineered.withColumn("hari_dalam_minggu",
dayofweek("timestamp_reg")) # 1=Minggu, 7=Sabtu

print("Data setelah ekstraksi fitur tanggal:")
df_engineered.select("tgl_registrasi", "tahun_reg", "bulan_reg", "hari_dalam_minggu").show()
```

3.2 Encoding Variabel Kategorikal

Kita akan melakukan encoding pada kolom jenis_kelamin dan kota menggunakan StringIndexer dan OneHotEncoder.

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder

# 1. StringIndexer: Mengubah string menjadi indeks numerik
indexer_jk = StringIndexer(inputCol="jenis_kelamin", outputCol="jk_index")
indexer_kota = StringIndexer(inputCol="kota", outputCol="kota_index")

df_indexed = indexer_jk.fit(df_engineered).transform(df_engineered)
df_indexed = indexer_kota.fit(df_indexed).transform(df_indexed)

# 2. OneHotEncoder: Mengubah indeks menjadi vektor biner
ohe = OneHotEncoder(inputCols=["jk_index", "kota_index"], outputCols=["jk_ohe",
"kota_ohe"])
df_encoded = ohe.fit(df_indexed).transform(df_indexed)

print("Data setelah One-Hot Encoding:")
df_encoded.select("jenis_kelamin", "jk_index", "jk_ohe", "kota", "kota_index",
"kota_ohe").show(truncate=False)
```

3.3 Ekstraksi Fitur dari Teks (TF-IDF)

Kita akan memproses kolom ulasan untuk mengekstrak fitur menggunakan TF-IDF.

```
from pyspark.ml.feature import Tokenizer, HashingTF, IDF

# 1. Tokenizer: Memecah kalimat menjadi kata-kata (token)
tokenizer = Tokenizer(inputCol="ulasan", outputCol="kata")
df_tokenized = tokenizer.transform(df_encoded)

# 2. HashingTF: Menghitung frekuensi kata (Term Frequency)
hashingTF = HashingTF(inputCol="kata", outputCol="tf_raw", numFeatures=20)
df_tf = hashingTF.transform(df_tokenized)

# 3. IDF: Menghitung bobot IDF
idf = IDF(inputCol="tf_raw", outputCol="fitur_tfidf")
idfModel = idf.fit(df_tf)
```

```
df_tfidf = idfModel.transform(df_tf)

print("Data setelah Ekstraksi Fitur TF-IDF dari Ulasan:")
df_tfidf.select("ulasan", "fitur_tfidf").show(truncate=False)
```

Langkah Terakhir: Menghentikan SparkSession

Setelah selesai, jangan lupa untuk menghentikan sesi Spark untuk melepaskan sumber daya.

```
spark.stop()
print("SparkSession telah dihentikan.")
```

BAGIAN 4: LATIHAN DAN TUGAS

Untuk memperdalam pemahaman Anda, kerjakan latihan dan tugas di bawah ini.

4.1 Latihan

Jawablah pertanyaan atau selesaikan instruksi berikut berdasarkan df_bersih yang telah kita buat sebelumnya:

1. **Agregasi Lanjutan:**
   - Kelompokkan data berdasarkan jenis_kelamin dan kota.
   - Hitung gaji **maksimum** dan usia **minimum** untuk setiap kelompok.
2. **Diskretisasi Gaji:**
   - Buatlah kategori baru untuk kolom gaji dengan nama level_gaji.
   - Gunakan Bucketizer dengan batasan sebagai berikut:
     - Gaji < 7,000,000: "Rendah"
     - Gaji 7,000,000 - 15,000,000: "Menengah"
     - Gaji > 15,000,000: "Tinggi"
   - Tampilkan kolom gaji dan level_gaji yang baru.
3. **Feature Engineering Sederhana:**
   - Buat fitur interaksi baru bernama usia_x_gaji yang merupakan hasil perkalian antara kolom usia dan gaji.
   - Tampilkan 5 baris pertama dari id_pelanggan, usia, gaji, dan usia_x_gaji.

4.2 Tugas

Buatlah sebuah alur pra-pemrosesan data lengkap dari awal menggunakan dataset baru di bawah ini. Lakukan semua langkah yang diperlukan (Data Cleaning, Transformasi, dan

Feature Engineering) hingga data siap untuk digunakan oleh model machine learning.

Dataset Tugas: Data Produk Elektronik

```
# Jalankan sel ini untuk membuat DataFrame tugas
data_produk = [
    (101, 'Laptop A', 'Elektronik', 15000000, 4.5, 120, '2023-01-20', 'stok_tersedia'),
    (102, 'Smartphone B', 'Elektronik', 8000000, 4.7, 250, '2023-02-10', 'stok_tersedia'),
    (103, 'Headphone C', 'Aksesoris', 1200000, 4.2, None, '2023-02-15', 'stok_habis'),
    (104, 'Laptop A', 'Elektronik', 15000000, 4.5, 120, '2023-01-20', 'stok_tersedia'), # Duplikat
    (105, 'Tablet D', 'Elektronik', 6500000, None, 80, '2023-03-01', 'stok_tersedia'),
    (106, 'Charger E', 'Aksesoris', 250000, -4.0, 500, '2023-03-05', 'Stok_Tersedia'), # Rating
tidak valid & Status inkonsisten
    (107, 'Smartwatch F', 'Elektronik', 3100000, 4.8, 150, '2023-04-12', 'stok_habis')
]

skema_produk = StructType([
    StructField("id_produk", IntegerType()),
    StructField("nama_produk", StringType()),
    StructField("kategori", StringType()),
    StructField("harga", IntegerType()),
    StructField("rating", FloatType()),
    StructField("terjual", IntegerType()),
    StructField("tgl_rilis", StringType()),
    StructField("status_stok", StringType())
])

df_tugas = spark.createDataFrame(data=data_produk, schema=skema_produk)
df_tugas.show()
```

Instruksi Tugas:

1. **Lakukan Data Cleaning:**
    - Tangani nilai yang hilang (None) pada kolom terjual dan rating. Pilih metode imputasi yang menurut Anda paling sesuai.
    - Hapus data duplikat.
    - Perbaiki nilai rating yang tidak valid (negatif).
    - Standarisasi nilai pada kolom status_stok (misalnya, ubah semua menjadi huruf kecil).
2. **Lakukan Data Transformasi:**

    - Lakukan **standarisasi** pada kolom numerik (harga, rating, terjual).
3. **Lakukan Feature Engineering:**
    - Ekstrak fitur bulan_rilis dari kolom tgl_rilis.
    - Lakukan **One-Hot Encoding** pada kolom kategori dan status_stok.
4. **Tampilkan Hasil Akhir:**
    - Tampilkan 10 baris pertama dari DataFrame akhir yang telah bersih dan memiliki fitur-fitur baru. Pastikan semua kolom hasil transformasi dan engineering terlihat.