LAPORAN AKHIR PROYEK PENGOLAHAN DATA BESAR

Clustering Analysis With Spark MLib API for E-Commerce Dataset

Disusun oleh:

1. **12S17025** **Evola R A Tampubolon**
2. **12S17050** **Kotrel Manurung**
3. **12S17059** **Ekis Naomi Lasma**
4. **12S17067** **Fradina Kristina Sinambela**

PROGRAM STUDI SARJANA SISTEM INFORMASI

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

INSTITUT TEKNOLOGI DEL

2021

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang semakin meluas menghasilkan berbagai jenis data dalam jumlah yang sangat besar. Pemanfaatan teknologi pun sudah banyak digandrungi oleh berbagai bidang, salah satunya *e-commerce*. Data yang dihasilkan oleh sebuah layanan *e-commerce* sangatlah besar dan beragam, data tersebut pada dasarnya sangat dibutuhkan untuk melakukan analisis yang bertujuan untuk meningkatkan kualitas barang, pelayanan ataupun untuk menghasilkan sebuah laporan yang diperlukan sebagai dokumentasi.

Salah satu solusi yang dapat mengatasi analisis data dalam jumlah besar dan beragam adalah dengan menggunakan teknologi yang sedang berkembang juga yaitu algoritma *clustering*. *Clustering* adalah sebuah algoritma yang membantu untuk menemukan kesamaan atau *similarity* yang tersembunyi antar data, seperti kesamaan antara harga produk atau layanan yang diberikan kepada pelanggan dalam jangka waktu tertentu. Dalam hal ini, algoritma *clustering* sangat membantu dalam melakukan pengelompokan terhadap pelanggan dengan satu pola yang sama. Terdapat beberapa pendekatan yang dapat digunakan untuk menangani case *clustering* seperti partition based, density based, hierarchical based dan model based approaches. Beberapa algoritma yang diusulkan pada pendekatan tersebut adalah k-means *clustering*, c-means *clustering*, DBSCAN, CLARA, BIRCH dan Gaussian Mixture Mode. Dalam pengerjaan proyek ini digunakan algoritma K-means yang merupakan *clustering* dengan pendekatan partition based dan populer dalam golongannya.

Apache Spark adalah sebuah platform *cluster computing* yang dimanfaatkan untuk melakukan pengelolaan terhadap big data dengan waktu cepat. Spark diyakini dapat menjalankan program hingga 100 kali lebih cepat di memori atau 10 kali lebih cepat pada disk dibandingkan dengan penggunaan Hadoop. MLlib adalah sebuah *library machine learning* yang menyediakan berbagai algoritma dan dirancang untuk dapat menskalakan cluster untuk klasifikasi, regression, *clustering*, collaborative filtering, dan lainnya.

Pada pengerjaan proyek kali ini digunakan sebuah kumpulan data transaksional yang berisi pencatatan transaksi yang terjadi pada rentang waktu 01/12/2010 hingga 09/12/2011. Data tersebut didapatkan dari pedagang yang terdaftar tidak membuka online store dan bertempat

di UK. Perusahaan tersebut menjual hadiah unik atau cenderamata untuk berbagai kesempatan. Kebanyakan pelanggan perusahaan tersebut adalah grosir. Data yang digunakan didukung dan dipublikasikan oleh The UCI Machine Learning Repository. Untuk itu dilakukan segmentasi *customer* berdasarkan profil dari masing-masing *customer* kemudian menganalisis hasil segmentasinya untuk diketahui pelanggan yang profitable maupun sebaliknya. Untuk melakukan *profiling* pada *customer* maka diperlukan sebuah model yang memberikan gambaran segala aktivitas pelanggan, kebutuhan, keinginan, dan juga konsentrasi terhadap produk dan layanan perusahaan. Model yang umum dalam mengelompokkan pelanggan adalah model *Recency, Frequency, Monetary* (RFM), yaitu melakukan mengelompokkan pelanggan berdasarkan interval waktu transaksi terakhir pelanggan, frekuensi transaksi, dan besaran nilai yang dikeluarkan sebagai royalty perusahaan.

1.2 Tujuan

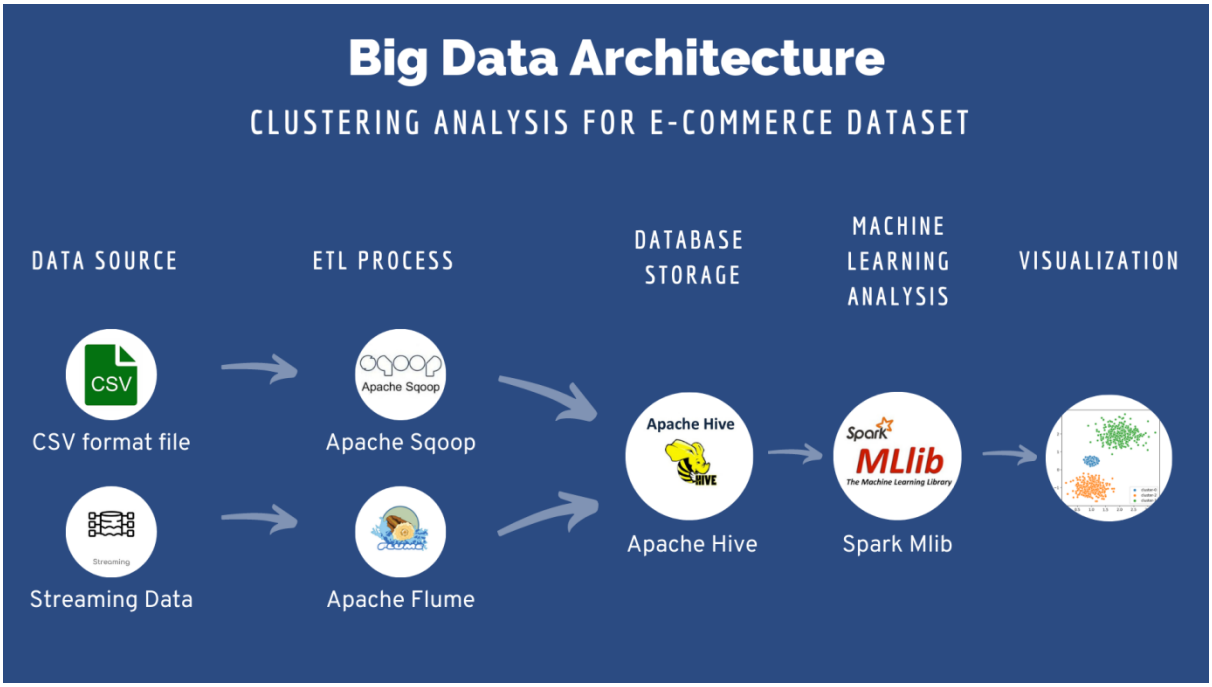
Adapun yang menjadi tujuan dari pengerjaan proyek *Clustering* Analysis With Spark MLib API untuk Dataset E-Commerce, adalah sebagai berikut:

1. Merancang arsitektur Hadoop dan Apache Spark yang menjadi solusi pengelolaan data transaksi E-Commerce UK Retail.
2. Menghasilkan nilai ataupun *score segmen Recency, Frequency,* dan *Monetary* dengan menggunakan analisis RFM *Segmentation.*
3. Menghasilkan segmentasi *customer* dengan menggnakan gabungan dari analisis RFM dengan *Library Machine Learning* untuk *clustering* yakni K-Means.
4. Merepresentasikan hasil *clustering* yang didapat dengan menggunakan tools Visualisasi Tableau dan juga menggunakan visualisasi *scatter plot* dari *library pandas.*

BAB 2 TINJAUAN PUSTAKA

2.1 Arsitektur *Big Data*

Arsitektur *big data* merupakan suatu struktur logis dan fisik yang dapat dirancang untuk menangani data yang akan disimpan untuk dapat diakses dan dikelola dalam suatu struktur data besar pada lingkungan teknologi informasi. Perancangan arsitektur *big data* merupakan proses ataupun aktivitas yang kompleks karena hasil dari perancangan tersebut dapat dijadikan sebagai sebuah referensi dalam menentukan solusi dari permasalahan infrastruktur data besar. Bervariasinya sumber data yang ada pada masa sekarang ini memiliki standar yang masing-masing bervariasi. Hal ini dapat menimbulkan adanya permasalahan pengolahan data yang melibatkan banyak sumber data. Arsitektur *big data* yang dirancang harus dapat bekerja tidak hanya untuk berbagai sumber data, namun juga untuk kemungkinan adanya perubahan skema data struktur yang digunakan untuk men-transport dan menyimpan data. Berikut merupakan rancangan arsitektur *big data* yang dirancang oleh tim pengembang.



Gambar 1 Perancangan Arsitektur *Big Data*

2.1.1 *Data Source*

Data source pada arsitektur *big data* merupakan sumber data yang akan digunakan pada pemrosesan data besar. *Data Source* dapat dibedakan menjadi dua jenis tipe data utama

dalam susunan big data, yaitu data terstruktur dan data tidak terstruktur. Data terstruktur *(structured data)* secara umum mengacu pada data yang memiliki panjang dan format yang telah terdefinisi sebelumnya. Sebaliknya, data yang tidak terstruktur *(unstructured data)* merupakan data yang tidak mengikuti suatu susunan format tertentu. Selain itu, sumber dari data dapat dibagi kedalam dua buah kategori, yakni data yang dihasilkan oleh mesin (komputer) dan data yang dihasilkan oleh manusia. [1] Dalam pengerjaan proyek ini, sumber data yang digunakan berasal dari *E-Commerce* Dataset yang diperoleh dari UCI *Machine Learning dataset* yang dapat diakses secara umum dari situs Kaggle. Dataset yang digunakan pada pengerjaan proyek memiliki beberapa kolom dengan penjelasan sebagai berikut:

1. **InvoiceNo,** berisi ID unik untuk setiap penjualan.
2. **StockCode,** berisi ID unik untuk setiap produk.
3. **Quantity,** berisi jumlah barang dalam setiap pembelian.
4. **InviceDate,** berisi tanggal penjualan dari setiap invoice.
5. **UnitPrice,** berisi harga satuan produk.
6. ***Customer*ID,** berisi ID unik untuk setiap pelanggan.

2.1.2 ETL *Process*

Dataset yang dimiliki kemudian diproses berdasarkan pemrosesan *Extract, Transfer, Load* dengan menggunakan Apache Sqoop untuk pemrosesan *batch* data, dan Apache Flume untuk pemrosesan data *streaming*. Pemilihan Apache Sqoop sebagai *tool* dalam pemrosesan ETL untuk data *batch* adalah karena data yang akan diolah merupakan dataset terstruktur yang berbentuk batch (tidak streaming) sehingga nantinya akan memerlukan sebuah basis data sementara yang digunakan untuk menampung data *batch.*

2.1.3 *Database Storage*

Data yang sudah melalui proses ETL akan disimpan pada Apache Hive yang merupakan sistem gudang data terdistribusi dan toleran terhadap kesalahan (fault tolerant) yang memungkinkan untuk dilakukan analisis dalam skala besar. Pemilihan apache hive sebagai *database storage* adalah karena menyediakan penyimpanan informasi yang dapat dengan mudah dilakukan analisis untuk pengambilan keputusan berdasarkan informasi dan data yang disimpan, juga memungkinkan pengguna untuk membaca, menulis dan mengelola petabyte data menggunakan SQL. Apache Hive dibangun di atas Apache Hadoop yang menghasilkan

integrasi menakjubkan yang memberikan kecepatan bekerja pada data berukuran petabyte dan data dalam jumlah sangat besar.

2.1.4 *Machine Learning Analysis*

Spark MLib digunakan untuk proses pembelajaran mesin pada Apache Spark yang dapat diskalakan untuk proses analisis berkualitas tinggi dan pada kecepatan yang tinggi yang dibangun diatas DataFrame untuk membangun *pipeline* pembelajaran mesin. Spark MLib terdiri dari algoritma dan utilitas popular. Terdapat juga beberapa pembelajaran mesin pada Spark MLib, diantaranya seperti regresi, klasifikasi, *clustering*, penambangan pola, dan pemfilteran kolaboratif. Spark ML menjadi API yang utama digunakan pada *Machine Learning* yang terdapat pada Spark.

2.1.5 *Visualization*

Visualisasi data merupakan sebuah ilmu yang merepresentasikan penggambaran dari sebuah data dimana segala informasi telah melalui proses abstraksi ke dalam bentuk skematik, termasuk juga di dalamnya karakteristik ataupun variabel untuk setiap unit informasi. Tujuan diterapkannya visualisasi data adalah untuk mengkomunikasikan informasi maupun data secara jelas dan efisien dapat melalui penggambaran dengan menggunakan idiom-idiom visualisasi seperti grafik statistik, *scatter plot*, *bar chart*, *pie chart*, dan lain sebagainya sehingga memudahkan pembaca dalam memahami konsep dan dapat mengidentifikasi pola-pola yang terdapat dalam data. [2] Berikut merupakan beberapa manfaat yang didapat dengan menerapkan visualisasi data untuk kapasitas data yang besar:

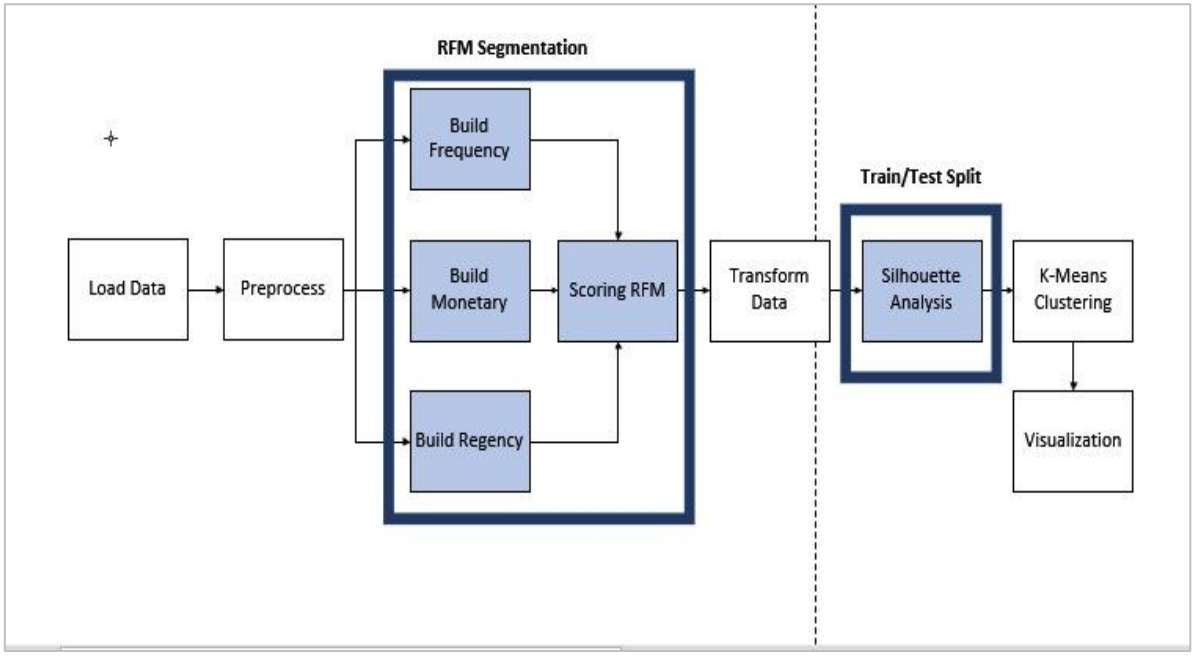
1. *Communicate Findings in Constructive Ways*

   Dengan adanya visualisasi data, maka dapat memungkinkan untuk merangkum keseluruhan informasi yang bersifat kompleks menjadi sebuah serangkaian gambaran sederhana.
2. *Faster Action*

   Memberikan bentuk komunikasi yang jelas dan memungkinkan beberapa pihak dalam melakukan penafsiran dalam pengambilan keputusan berdasarkan informasi yang didapat dari hasil visualisasi data dan juga dapat dengan lebih cepat memberikan respon yang sesuai terhadap perubahan kebutuhan pada kondisi tertentu.
3. *Understand Connections Between Operations and Results*

Visualisasi data memungkinkan pembaca dalam menemukan korelasi antara fungsi bisnis dan kinerja dalam lingkungan yang kompetitif. Maka dengan adanya visualisasi, pihak perusahaan ataupun organisasi dapat dengan tanggap menanggulangi apabila ditemukan kendala ataupun permasalahan.

2.2 *Machine Learning Pipeline*



Gambar 2 Machine Learning Pipeline

1. ***Load Data***

   Tahap pertama yang dilakukan dalam pengerjaan proyek ini adalah memuat data yang akan digunakan selama pengerjaan dan akan dilanjutkan dengan analisis data jika diperlukan seperti memahami isi dari data tersebut dan hal apa yang perlu dilakukan untuk mendapatkan informasi yang berguna sebagai hasil dari pengerjaan proyek.

2. ***Preprocess***

   Data yang digunakan dalam pengerjaan proyek merupakan data mentah yang harus melalui beberapa proses untuk dapat digunakan oleh model yang dipakai. Data preprocessing dapat juga disebut sebagai proses transformasi data. Dalam pengerjaan proyek ini akan dilakukan perubahan format data menjadi parquet agar dapat digunakan oleh model yang dipakai. Setelah data melalui preprocessing akan dilakukan pemilihan model yang akan digunakan sesuai dengan latar belakang dan tujuan dilakukannya pengerjaan proyek ini.

3. **RFM Segmentation**

Data yang sudah siap pakai akan digunakan untuk melalui RFM Segmentation yang dipilih untuk melakukan identifikasi kelompok (cluster) pelanggan sesuai dengan kebutuhan. Alasan lain untuk memilih metode ini adalah metode ini menggunakan skala numerik yang objektif sehingga menghasilkan gambaran tingkat tinggi yang ringkas dan informatif mengenai pelangga. RFM adalah singkatan dari Frequency, Monetary dan Regency yang artinya akan dijelaskan pada poin dibawah ini.

a. **Build Frequency**

Frequency adalah seberapa sering pelanggan melakukan transaksi dengan penjual atau total transaksi selama jangka waktu tertentu (jangka waktu sesuai dengan data yang digunakan). Dengan dilakukannya perhitungan frequency ini dapat dilihat secara kasar jumlah dan gambaran cluster yang akan dibangun.

b. **Build Monetary**

Monetary adalah berapa banyak dana yang dikeluarkan oleh pelanggan terhadap pembelian sebuah barang dalam jangka waktu tertentu. Segmentasi pelanggan dilkukan dengan beberapa pertimbangan yang harus diperhatikan dan dilakukan dengan baik. Salah satu factor sekunder yang dimaksud adalah monetary dibagi dengan frekuensi untuk menunjukkan pembelian rata-rata.

c. **Build Regency**

Recency adalah jangka waktu yang telah berlalu dihitung sejak transaksi terkahir pelanggan dengan waktu terakhir data dikumpulkan. Beberapa kasus menyatakan bahwa semakin baru pelanggan melakukan transaksi dengan penjual maka akan semakin besar kemungkinan sikap responsive pelanggan terhadap komunikasi dengan penjual tersebut.

4. ***Transform* Data**

Transform data yang dilakukan sebagai tahap persiapan sebelum memasuki tahapan K-Means clustering dilakukan dua langkah yaitu dengan membangun feature matrix menggunakan vector dense dan melakukan normalisasi data dengan fungsi MinMax Scaler.

5. **Silhouette Analysis**

Silhouette analysis adalah salah satu metode evaluasi yang digunakan dalam clustering yang berguna untuk menghitung koefisien silhouette dan dengan mudah

dapat menemukan angka pasti untuk k yang akan digunakan. Nilai dari silhouette coefficient ada diantara rentang [-1, 1], angka 1 menunjukkan angka yang terbaik dimana titik data o sangat kompak atau sejenis dalam cluster tempatnya berada dan jauh dari cluster lainnya. Sedangkan nilai -1 adalah kebalikannya yaitu nilai terburuk yang menyatakan bahwa nilai yang mendekati 0 menunjukkan cluster yang overlapping atau tumpang tindih.

6. **K-Means *Clustering***

   K-means clustering adalah unsupervised machine learning algorithm yang paling sederhana dan populer dalam penggunaannya. Unsupervised machine learning menggunakan data tanpa label untuk memahami sifat struktur data seperti grup, kluster dan lainnta. Clustering K-means dilakukan untuk membentuk partisi sejumlah n objek menjadi k cluster dimana setiap objek termasuk ke dalam bagian cluster dengan mean atau nilai rata-rata terdekat. Metode ini dapat menghasilkan nilai k yang paling tepat dan cluster yang berbeda dengan kemungkinan perbedaan paling besar. Tujuan dari K-means clustering adalah untuk meminimalkan objective function dan atau fungsi sum of squared error (SSE).

7. **Visualization**

   Hasil clustering yang didapatkan melalui beberapa proses sebelumnya akan diubah menjadi bentuk yang mudah dipahami dan dapat memilih sesuai dengan kebutuhan. Bentuk tersebut disebut dengan visualisasi, visualisasi dapat dilakukan dengan menggunakan dua cara dalam kasus ini, yang pertama dengan menggunakan scatter plot dengan menggunakan fungsi pandas dan menggunakan tools luar yaitu Tableau untuk membuat visualisasi lebih menarik dan lebih informatif.

BAB 3 IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai lingkungan implementasi, hingga penerapan dan peng-implementasian *clustering* untuk dataset *E-Commerce* menggunakan *Spark Machine Learning Library API.*

3.1 Lingkungan Implementasi

Lingkungan implementasi pada pengerjaan proyek ini mencakup perangkat keras (Hardware) dan juga perangkat lunak (*Software)* yang digunakan dalam proses implementasi. Selanjutnya akan dijelaskan detail perangkat keras dan perangkat lunak yang digunakan pada proses pengerjaan proyek.

3.1.1 Perangkat Keras *(Hardware)*

Daftar perangkat keras yang digunakan untuk proses implementasi pada proyek ini dapat dilihat pada tabel berikut ini.

Tabel 1 Perangkat Keras

Hardware	Spesifikasi
Laptop (PC)	Acer
Processor	Core i3
RAM	4 GB

3.1.2 Perangkat Lunak *(Software)*

Spesifikasi perangkat lunak yang digunakan dalam proses implementasi pada proyek ini dapat dilihat pada tabel berikut ini.

Tabel 2 Perangkat Lunak

Software	Spesifikasi
Sistem Operasi	Windows 10
Editor	Jupyter Notebook, Google Colaboratory
Documentation	Microsoft Office 2013
Bahasa Pemrograman	Python

3.2 Implementasi *Start* Spark Session

Proses pertama dalam pengerjaan proyek adalah dengan memulai sesi dari spark untuk proses implementasi proyek. Terlebih dahulu melakukan import pyspark dan *library-library* yang akan digunakan untuk pengerjaan proyek yakni seperti *library* K-Means, *library* K-Means Model, *library Clustering*Evaluator dan *library* matplotlib.

```
#import findspark
#findspark.init()
import pyspark
from pyspark.ml.clustering import KMeans
from pyspark.ml.clustering import KMeansModel
from pyspark.ml.evaluation import ClusteringEvaluator
import matplotlib.pyplot as plt
%matplotlib inline
```

Setelah seluruh *library* yang dibutuhkan dalam pengerjaan proyek telah di-*import*, selanjutnya adalah menentukan berapa jumlah *core* yang akan dipakai pada proses *clustering* dengan spark mlib dan juga menentukan nama aplikasi. Atau dapat dilihat pada potongan *code* berikut ini.

```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .master("local[2]") \
    .appName("Proyek PBD Cluster E-Commerce") \
    .getOrCreate()
```

3.3 Implementasi *Convert* dan *Load Dataset*

Untuk proses konversi dataset, data yang sebelumnya masih dalam format .csv kemudian diubah kedalam hadoop format file yakni Parquet.

```
from google.colab import drive
drive.mount('/content/drive/')
data = spark.read.load("/content/drive/MyDrive/PROYEK PBD/dataset/d
ata.csv",
```

```
format="csv", inferSchema="true", header="true")
#convert dataset
data.write.format("parquet").mode("overwrite").save("/content/drive
/MyDrive/PROYEK PBD/dataset/data_parq")
```

Proses selanjutnya adalah memuat dataset dari file *local* dengan ekstensi .parquet ke dalam spark *dataframe* dengan menggunakan fungsi spark.read. Berikut merupakan potongan kode yang dapat dijalankan untuk peng-implementasian *import* dataset.

```
data_parq = spark.read.load("/content/drive/MyDrive/PROYEK PBD/data
set/data_parq",
format="parquet", inferSchema="true", header="true")
```

3.4 Implementasi *Exploratory Data*

Pada tahapan ini akan dilakukan investigasi awal sehingga akan diperoleh informasi mengenai kondisi dataset yang akan digunakan. Proses ini bertujuan untuk mempersiapkan pembentukan model *machine learning*. Berikut merupakan potongan *code* yang dapat digunakan untuk tahapan implementasi *exploratory data.*

```
data_parq.show(10)
data_parq.printSchema()

#check deskripsi dataset
data_parq.describe().show()
```

3.5 Implementasi *Data Cleaning* dan *Data Manipulation*

1. ***Data Cleaning***

   Record data yang memiliki nilai null akan dihapus dari frame data dengan menggunakan fungsi .dropna()

   ```
   from pyspark.sql.functions import count
   #check and remove the null values
   def my_count(df_in):
       df_in.agg( *[ count(c).alias(c) for c in df_in.columns ] ).
   show()
   ```

```
ecom_data = data_parq.dropna(how='any')
my_count(ecom_data)
```

2. *Data Manipulation*

a. Manipulasi Perhitungan *Total Price*

Manipulasi Perhitungan Total Price dilakukan dengan perhitungan untuk kolom baru yakni Total Price dengan menggunakan perkalian antara Quantity dengan Unit Price (Quantity * Unit Price)

```
from pyspark.sql.functions import round

ecom_data = ecom_data.withColumn('TotalPrice', round( ecom_d
ata.Quantity * ecom_data.UnitPrice, 2 ) )
ecom_data.show(5)
```

b. Manipulasi Format Tanggal

Jika dilihat pada kolom InvoiceDate masi dalam format MM/dd/yy HH:mm. Dikarenakan Spark sangat sensitive terhadap format tanggal, maka diperlukan membuat kolom baru dengan nama kolom NewInvoiceDate dengan menngunakan value yang terdapat pada kolom InvoiceDate.

```
from pyspark.sql.functions import to_date
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
ecom_data.withColumn("NewInvoiceDate", to_date(col("InvoiceD
ate"), "MM/dd/yyyy HH:mm")).show(5)
```

c. Manipulasi Perhitungan *Duration (Time Difference)*

```
from pyspark.sql.functions import mean, min, max, sum, dated
iff, to_date
date_max = ecom_data.select(max('NewInvoiceDate')).toPandas(
)
current = to_utc_timestamp( unix_timestamp(lit(str(date_max.
iloc[0][0])), \
                                'yy-MM-
dd HH:mm').cast('timestamp'), 'UTC' )
#Calculatre Duration
ecom_data = ecom_data.withColumn('Duration', datediff(lit(cu
```

```
rrent), 'NewInvoiceDate'))
```

3.6 Implementasi Segmentasi RFM

RFM atau *Recency*, *Frequency*, dan *Monetary* merupakan sebuah metode yang digunakan untuk menganalisis nilai ataupun value dari *customer*. Analisis RFM biasanya digunakan pada basis data *marketing* dan *direct marketing* dan telah mendapatkan perhatian khusus dalam industry jasa retai dan professional. Berikut merupakan rincian lebih lanjut mengenai RFM:

a. *Recency Value*, merepresentasikan seberapa baru pelanggan membeli sesuatu? Sebagai contoh: Berapa lama durasi sejak pembelian terakhir.
b. *Frequency Value*, merepresentasikan seberapa sering pelanggan membeli? Sebagai contoh: Berapa kali jumlah total pembelian.
c. *Monetary Value*, merepresentasikan seberapa banyak yang dibelanjakan pelanggan? Sebagai contoh: Berapa total uang yang dibelanjakan perlanggan.

1. Membangun Recency, Frequency dan Monetary

```
recency = ecom_data.groupBy('CustomerID').agg(min('Duration').a
lias('Recency'))

frequency = ecom_data.groupBy('CustomerID', 'InvoiceNo').count(
)\
                        .groupBy('CustomerID')\
                        .agg(count("*").alias("Frequency"))

monetary = ecom_data.groupBy('CustomerID').agg(round(sum('Total
Price'), 2).alias('Monetary'))

rfm = recency.join(frequency,'CustomerID', how = 'inner')\
             .join(monetary,'CustomerID', how = 'inner')
```

2. Segmentasi RFM

a. Menentukan dan mendefenisikan user dengan menggunakan *Cutting Point.*

Cutting Point merupakan sebuah metode yang digunakan untuk memilah penggunaan suatu kriteria untuk dijadikan sebagai pertimbangan dalam pengambilan sebuah keputusan terhadap suatu masalah. Metode *Cutting Point* juga digunakan untuk

memastikan seberapa penting derajat sebuah kriteria. Pada proyek ini, ditentukan nilai *cutting point* yang akan digunakan adalah quartile dari total populasi data.

```
def describe_pd(df_in, columns, style):
    '''
    Function to union the basic stats results and deciles
    :param df_in: the input dataframe
    :param columns: the cloumn name list of the numerical va
riable
    :param style: the display style
    :return : the numerical describe info. of the input data
frame
    :author: MIng Chen and Wenqiang Feng
    :email: von198@gmail.com
    '''
    if style == 1:
        percentiles = [25, 50, 75]
    else:
        percentiles = np.array(range(0, 110, 10))

    percs = np.transpose([np.percentile(df_in.select(x).coll
ect(), percentiles) for x in columns])
    percs = pd.DataFrame(percs, columns=columns)
    percs['summary'] = [str(p) + '%' for p in percentiles]

    spark_describe = df_in.describe().toPandas()
    new_df = pd.concat([spark_describe, percs],ignore_index=
True)
    new_df = new_df.round(2)
    return new_df[['summary'] + columns]

cols = ['Recency','Frequency','Monetary']
describe_pd(rfm,cols,1)
```

b. Mendefenisikan Score Segmen RFM dengan menggunakan *Cutting Point (Quartil).*

Berikut merupakan Tabel yang merepresenatsikan score segmen RFM dengan menggunakan *cutting point* yang sebelumnya telah didefenisikan dengan menggunakan total quartile dari populasi dataset.

Tabel 3 Pendefenisian Score Segmen

Quatil	Recency	Frequency	Monetary
1	</= 16	</= 1	</= 4287.63
2	</= 50	</= 3	</= 648.07
3	</= 143	</= 5	</= 1611.72
4	> 143	> 5	> 1611.72

Berikut merupakan pengimplementasian dalam berupa kode program.

```
def RScore(x):
    if  x <= 16:
        return 1
    elif x<= 50:
        return 2
    elif x<= 143:
        return 3
    else:
        return 4

def FScore(x):
    if  x <= 1:
        return 4
    elif x <= 3:
        return 3
    elif x <= 5:
        return 2
    else:
        return 1

def MScore(x):
    if  x <= 293:
        return 4
    elif x <= 648:
        return 3
    elif x <= 1611:
        return 2
```

```
    else:
        return 1

from pyspark.sql.functions import udf
from pyspark.sql.types import StringType, DoubleType

R_udf = udf(lambda x: RScore(x), StringType())
F_udf = udf(lambda x: FScore(x), StringType())
M_udf = udf(lambda x: MScore(x), StringType())
```

c. Pengimplementasian Segmentasi R, F, dan M

Pada tahapan ini, hasil pendefenisian masing-masing score segmen RFM yang didapat dari hasil cutting point kemudian di*assign* kedalam tabel bersamaan dengan kolom CutomerID, Recency, Frequency dan Monetary.

```
rfm_seg = rfm.withColumn("r_seg", R_udf("Recency"))
rfm_seg = rfm_seg.withColumn("f_seg", F_udf("Frequency"))
rfm_seg = rfm_seg.withColumn("m_seg", M_udf("Monetary"))
```

d. Menentukan Score Gabungan RFM

Dengan menggabungkan score masing-masing segmen R, F, dan M menggunakan fungsi concanate.

```
from pyspark.sql import functions as F
rfm_seg = rfm_seg.withColumn('RFMScore',
                             F.concat(F.col('r_seg'),F.col('
f_seg'), F.col('m_seg')))
```

3.7 Implementasi K-Means *Clustering*

1. Membangun feature matrix dengan menggunakan vector dense.

Dense matrix dibuat dengan memanggil fungsi matriks. Argumen menentukan nilai koefisien, dimensi, dan jenis (bilangan bulat, ganda, atau kompleks) dari matriks. Matriks dengan satu blok-kolom dapat diwakili oleh satu daftar untuk mendifinisikan data yang akan digunakan clustering.

```
from pyspark.sql import Row
from pyspark.ml.linalg import Vectors
#Method (good for large features):
def transData(data):
    return data.rdd.map(lambda r: [r[0],Vectors.dense(r[1:])]
).toDF(['CustomerID','rfm'])
transformed = transData(rfm)
```

2. Melakukan Normalisasi Data.

Normalisasi data diterapkan dengan menggunakan MinMax Scaler yang bekerja men-scaling atau menyesuaikan data dalam rentang tertentu (range nilai *minimum* hingga nilai maksimum)

```
#Scaler the feature matrix
from pyspark.ml.feature import MinMaxScaler

scaler = MinMaxScaler(inputCol="rfm",\
         outputCol="features")
scalerModel =  scaler.fit(transformed)
scaledData = scalerModel.transform(transformed)
```

3. Menentukan Jumlah *Cluster* Optimal.

Dalam menentukan jumlah *cluster* yang paling baik, dapat menggunakan analisis *Silhouette* dengan pertimbanangan sebagai berikut

Tabel 4 Nilai Silhouette Coefficient

No	Rentang Nilai SC	Keterangan
1	0.7 < SC <= 1	*Strong Structure*
2	0.5 < SC <= 0.7	*Medium Structure*
3	0.25 < SC <= 0.5	*Weak Structure*
4	SC <= 0.25	*No Structure*

Untuk melihat *cluster* dengan jumlah K yang paling optimal, maka dilakukan percobaan sebanyak 3 kali untuk membandingkan nilai silhouette yang paling baik

dihasilkan dengan meng-assign nilai K = 3, K = 4, dan K = 5. Berikut merupakan potongan program peng-implementasian apabila menggunakan K = 3.

```
kmeans = KMeans().setK(3).setSeed(1)
model = kmeans.fit(scaledData)

#Make predictions, i.e., get cluster index
predictions = model.transform(scaledData)
#Evaluate clustering by computing Silhouette score
evaluator = ClusteringEvaluator()

silhouettes = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(s
ilhouettes))

# Shows the result.
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

4. Implementasi K-Means *Clustering*

Dengan menggunkan nilai K yang paling optimal yang didapat dari analisis silhouette, maka kemudian dilakukan pengimplementasian *cluster* k-means.

```
k = 3
kmeans = KMeans().setK(k).setSeed(1)
model = kmeans.fit(scaledData)
#Make predictions
predictions = model.transform(scaledData)
predictions.show(5,False)

results = rfm.join(predictions.select('CustomerID','prediction
'),'CustomerID',how='left')
```

3.8 Implementasi Visualisasi Segmen RFM dan K-Means *Clustering*

Dalam membuat visualisasi dari hasil segmentasi RFM dan K-Means *Clustering* yang telah diperoleh, digunakan Tableau dan visualisasi dengan menggunakan *library* dari *pandas* sebagai alat (tools) visualisasi yang diharapkan dapat mengelola data dengan baik. Data yang ditampilkan dengan visualisasi dapat lebih mudah untuk dipahami dan dianalisis oleh pengguna (user) dan juga lebih efektif untuk disajikan dalam bentuk visual daripada data dalam bentuk angka. Visualisasi memungkinkan analisis untuk mengidentifikasi trend, menemukan pola, dan hal lain dari data yang digunakan. Berikut merupakan potongan program yang dijalankan dalam membuat visualisasi berupa *scatter plot* dengan menggunkan library yang tersedia pada pandas.

```
from sklearn.datasets.samples_generator import make_blobs
from sklearn.metrics import pairwise_distances_argmin

X, y_true = make_blobs(n_samples=300, centers=3,
                       cluster_std=0.60, random_state=0)

rng = np.random.RandomState(42)
centers = [0, 3] + rng.randn(3, 2)

def draw_points(ax, c, factor=1):
    ax.scatter(X[:, 0], X[:, 1], c=c, cmap='viridis',
               s=50 * factor, alpha=0.3)

def draw_centers(ax, centers, factor=1, alpha=1.0):
    ax.scatter(centers[:, 0], centers[:, 1],
               c=np.arange(3), cmap='viridis', s=200 * factor, alph
a=alpha)
    ax.scatter(centers[:, 0], centers[:, 1],
               c='black', s=50 * factor, alpha=alpha)

def make_ax(fig, gs):
    ax = fig.add_subplot(gs)
    ax.xaxis.set_major_formatter(plt.NullFormatter())
    ax.yaxis.set_major_formatter(plt.NullFormatter())
    return ax

fig = plt.figure(figsize=(15, 3))
```

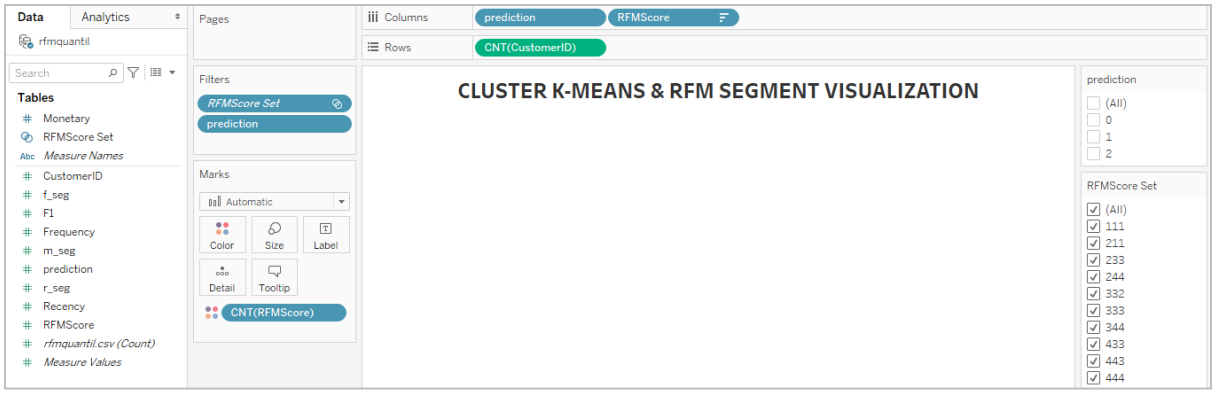
```
gs = plt.GridSpec(3, 15, left=0.02, right=0.98, bottom=0.05, top=0.
95, wspace=0.2, hspace=0.2)
ax0 = make_ax(fig, gs[:4, :4])
ax0.text(0.98, 0.98, "Random Initialization", transform=ax0.transAx
es,
         ha='right', va='top', size=16)
draw_points(ax0, 'gray', factor=2)
draw_centers(ax0, centers, factor=2)

for i in range(3):
    ax1 = make_ax(fig, gs[:2, 4 + 2 * i:6 + 2 * i])
    ax2 = make_ax(fig, gs[2:, 5 + 2 * i:7 + 2 * i])

    # E-step
    y_pred = pairwise_distances_argmin(X, centers)
    draw_points(ax1, y_pred)
    draw_centers(ax1, centers)

    # M-step
    new_centers = np.array([X[y_pred == i].mean(0) for i in range(3
)])
    draw_points(ax2, y_pred)
    draw_centers(ax2, centers, alpha=0.3)
    draw_centers(ax2, new_centers)
    for i in range(3):
        ax2.annotate('', new_centers[i], centers[i],
                     arrowprops=dict(arrowstyle='->', linewidth=1))
```

Berikut merupakan implementasi visualisasi yang dijalankan pada *tools* Tableau, dengan menggunakan data yang telah diproses sedemikian rupa sehingga menghasilkan segmen RFM dan tiga buah *cluster* dari pemrosesan dengan menggunkan algoritma K-Means. Atribut prediction dan RFMScore digunakan sebagai columns dan atribut CostumerID digunakan sebagai rows. Pada Tableau diterapkan juga filterisasi, dimana menggunakan prediction yang dapat difilter dan RFMScore yang di-set hanya menggunkan Top 10.



Gambar 3 Implementasi Visualisasi dengan Tableau

BAB 4 HASIL DAN PEMBAHASAN

4.1 Hasil *Exploratory Data*

Melalui implementasi *exploratory data*, maka dapat diketahui bagaimana karakteristik dari data yang dimiliki, seperti identitas masing-masing kolom dan juga tipe data untuk masing-masing kolom. Dapat diketahui juga apabila data yang dimiliki terdapat nilai *null* didalamnya, sehingga dapat ditentukan metode yang akan digunakan untuk mengatasinya. Berikut merupakan hasil dari proses *exploratory data.*

```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|   InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/2010 8:26|     2.55|     17850|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/2010 8:26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/2010 8:26|     7.65|     17850|United Kingdom|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/2010 8:26|     4.25|     17850|United Kingdom|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/2010 8:28|     1.85|     17850|United Kingdom|
|   536366|    22632|HAND WARMER RED P...|       6|12/1/2010 8:28|     1.85|     17850|United Kingdom|
|   536367|    84879|ASSORTED COLOUR B...|      32|12/1/2010 8:34|     1.69|     13047|United Kingdom|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
only showing top 10 rows

root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: string (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- Country: string (nullable = true)
```

Gambar 4 *Exploratory Data*

4.2 Hasil *Data Cleaning* dan *Data Manipulation*

1. Hasil *Data Cleaning*

Dengan menerapkan fungsi .dropna() maka semua *record data* yang berupa *null* akan di-drop dari frame data. Sebagai contoh, pada kolom *Customer*ID, terdapat nilai null, maka *record* data ini akan dihapus.

```
+---------+---------+-----------+--------+-----------+---------+----------+-------+
|InvoiceNo|StockCode|Description|Quantity|InvoiceDate|UnitPrice|CustomerID|Country|
+---------+---------+-----------+--------+-----------+---------+----------+-------+
|   406829|   406829|     406829|  406829|     406829|   406829|    406829| 406829|
+---------+---------+-----------+--------+-----------+---------+----------+-------+
```

Gambar 5 Hasil Data Cleaning

2. Hasil *Data Manipulation*

a. Hasil Manipulasi perhitungan *Total Price*

Untuk proses manipulasi perhitungan *Total Price* maka digunakan proses perkalian antara value dari kolom Quantity (Jumlah Barang) dengan value kolom UnitPrice (harga satuan). Sehingga akan diperoleh sebuah tabel baru pada *frame* data dengan nama kolom *Total Price*.

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice
536365	85123A	WHITE HANGING HEA...	6	12/1/2010 8:26	2.55	17850	United Kingdom	15.3
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34
536365	84406B	CREAM CUPID HEART...	8	12/1/2010 8:26	2.75	17850	United Kingdom	22.0
536365	84029G	KNITTED UNION FLA...	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34
536365	84029E	RED WOOLLY HOTTIE...	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34

only showing top 5 rows

Gambar 6 Manipulasi Perhitungan Total Price

b. Hasil Manipulasi format Tanggal

Dengan menggunakan *value* yang terdapat pada kolom InvoiceDate, maka format tanggal yang sebelum nya berupa mm/dd/yy hh:mm akan diubah menjadi format yyyy-mm-dd dalam sebuah kolom baru dengan nama NewInvoiceDate.

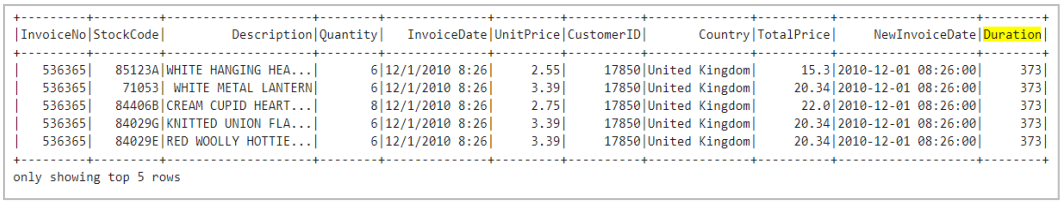
InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	NewInvoiceDate
536365	85123A	WHITE HANGING HEA...	6	12/1/2010 8:26	2.55	17850	United Kingdom	15.3	2010-12-01
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010-12-01
536365	84406B	CREAM CUPID HEART...	8	12/1/2010 8:26	2.75	17850	United Kingdom	22.0	2010-12-01
536365	84029G	KNITTED UNION FLA...	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010-12-01
536365	84029E	RED WOOLLY HOTTIE...	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010-12-01

only showing top 5 rows

Gambar 7 Manipulasi Format Tanggal

c. Hasil Manipulasi perhitungan *Time Difference (Duration)*

Dengan menggunakan hasil dari manipulasi format tanggal sebelumnya, dilakukan perhitungan hari antara tanggal *invoice* dikeluarkan dengan tanggal terakhir pengumpulan data yang digunakan. Hasil dari manipulasi yang *time difference* ini akan digunakan untuk melakukan perhitungan nilai segmen *Recency* pada analisis RFM.



InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	NewInvoiceDate	Duration
536365	85123A	WHITE HANGING HEA...	6	12/1/2010 8:26	2.55	17850	United Kingdom	15.3	2010-12-01 08:26:00	373
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010-12-01 08:26:00	373
536365	84406B	CREAM CUPID HEART...	8	12/1/2010 8:26	2.75	17850	United Kingdom	22.0	2010-12-01 08:26:00	373
536365	84029G	KNITTED UNION FLA...	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010-12-01 08:26:00	373
536365	84029E	RED WOOLLY HOTTIE...	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010-12-01 08:26:00	373

only showing top 5 rows

Gambar 8 Manipulasi Perhitungan *Time Difference*

4.3 Hasil Segmentasi RFM

1. Hasil Membangun Model RFM

Pada bab implementasi sebalumnya telah dijelaskan mengenai setiap langkah-langkah yang harus dilalui untuk mendapatkan hasil yang diinginkan dalam *customer* segmentation. RFM Segmentation digunakan untuk dapat melakukan analisis berupa mencari atau menghitung RFM score untuk setiap *customer*. Untuk itu, maka diperlukan membangun model segmen Recency, Frequency dan Monetary. Segmen Recency berasal dari hasil peng-grupan kolom *Customer*ID yang diagregasikan dengan kolom Duration yang sebelumnya diperoleh dari hasil manipulasi. Segmen Frequency berasal dari hasil peng-grupan kolom *Customer*ID dan kolom InvoiceNo yang kemudian diagregasikan. Segmen Monetary berasal dari hasil peng-grupan kolom *Customer*ID dan kolom TotalPrice yang diperoleh dari hasil manipulasi yang kemudian diagregasikan. Model Segmentasi digambarkan pada gambar berikut ini.

CustomerID	Recency	Frequency	Monetary
15619	10	1	336.4
17389	0	43	31300.08
12940	46	4	876.29
13623	30	7	672.44
14450	180	3	483.25
15727	16	7	5178.96
15790	10	1	220.85
13285	23	4	2709.12
14570	280	2	218.06
16574	71	1	451.44

only showing top 10 rows

Gambar 9 Segmentasi RFM

2. **Hasil Implementasi *Cutting Point* Segmen RFM**

Setelah model segmen Recency, Frequency dan Monetary dibangun, maka proses yang selanjutnya dilakukan adalah menentukan nilai *cutting point* untuk masing-masing model. Sesuai dengan implementasi yang telah dijelaskan sebelumnya, maka nilai yang digunakan pada penentuan *cutting point* adalah berupa quartile atau 25% dari populasi. Misalnya untuk menemukan 25% total pengeluaran uang teratas (Monetary) pada populasi data online retail yang digunakan pada proyek adalah sebesar 293.362. Hasil *cutting point* model masing-masing segmen dapat dilihat pada gambar berikut ini.

	summary	Recency	Frequency	Monetary
0	count	4372	4372	4372
1	mean	91.58119853613907	5.07548032936871	1898.4597003659658
2	stddev	100.77213931384833	9.338754163574729	8219.345141139722
3	min	0	1	-4287.63
4	max	373	248	279489.02
5	25%	16	1	293.362
6	50%	50	3	648.075
7	75%	143	5	1611.72

Gambar 10 Nilai Cutting Point Menggunakan Quartil

3. **Hasil Implementasi *Score* Segmen RFM**

Hasil implementasi Score Segmen RFM diperoleh dari hasil penggabigan score masing-masing segmen R, F, dan M dengan pendefenisian setiap populasi yang termasuk kedalam cutting point 25% maka masuk ke *score* segmen 1, populasi yang termasuk kedalam cutting point 50% maka masuk ke *score* segmen 2, populasi yang termasuk kedalam cutting point 75% maka masuk ke *score* segmen 3, dan populasi yang termasuk kedalam cutting point lebih besar dari nilai yang terdapat pada *cutting point* 75% maka masuk ke *score* segmen 4. Lebih lanjut dapat dilihat pada gambar berikut ini.

CustomerID	Recency	Frequency	Monetary	r_seg	f_seg	m_seg	RFMScore
16549	10	10	4154.64	1	1	1	111
13225	3	8	6083.04	1	1	1	111
15382	14	8	5927.86	1	1	1	111
17754	0	6	1739.92	1	1	1	111
14713	9	12	2664.26	1	1	1	111
17389	0	43	31300.08	1	1	1	111
12471	2	49	18740.92	1	1	1	111
15727	16	7	5178.96	1	1	1	111
17809	16	15	4627.62	1	1	1	111
18161	10	6	1612.79	1	1	1	111

only showing top 10 rows

Gambar 11 RFM Score

4.4 Hasil Implementasi K-Means *Clustering*

1. **Hasil Transformasi Data dengan Membangun Feature Matrix Menggunakan Vector Dense**

   Dalam implementasi K-means *Clustering* yang sudah dipaparkan pada bab sebelumnya dihasilkan data yang sudah di transformasikan dengan membangun *dense vector feature matrix*. Berikut merupakan hasil transformasi dengan membangun feature matrix menggunakan vector dense.

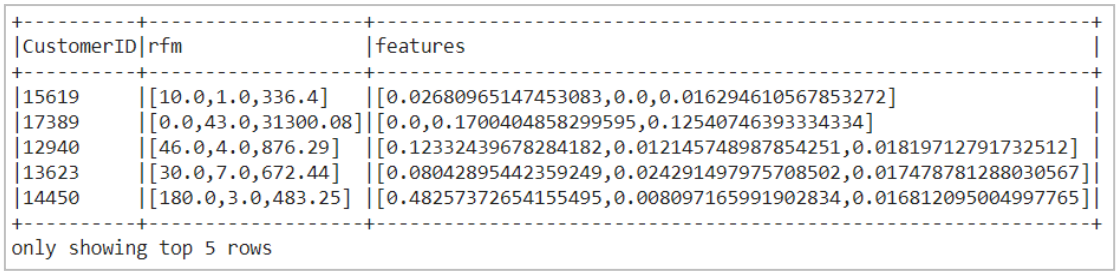
CustomerID	rfm
15619	[10.0,1.0,336.4]
17389	[0.0,43.0,31300.08]
12940	[46.0,4.0,876.29]
13623	[30.0,7.0,672.44]
14450	[180.0,3.0,483.25]

only showing top 5 rows

Gambar 12 Transformasi Data

2. Hasil Implementasi Normalisasi Data

Normalisasi data diterapkan dengan menggunakan MinMax Scaler yang bekerja men-scaling atau menyesuaikan data dalam rentang tertentu (range nilai *minimum* hingga nilai maksimum). Berikut merupakan detail dari hasil normalisasi data menggunakan MinMax Scaler.

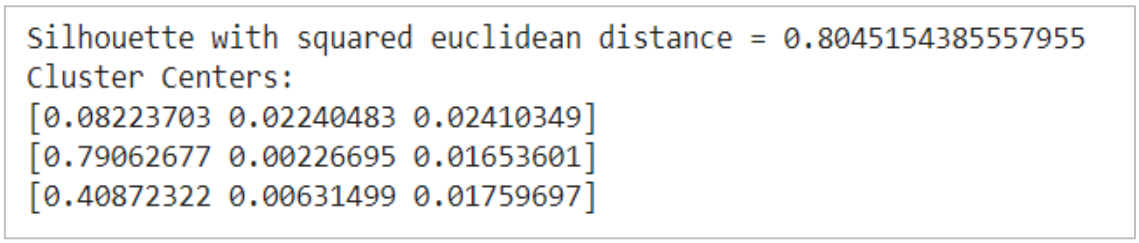


```
+----------+------------------+--------------------------------------------------------------+
|CustomerID|rfm               |features                                                      |
+----------+------------------+--------------------------------------------------------------+
|15619     |[10.0,1.0,336.4]  |[0.02680965147453083,0.0,0.016294610567853272]                |
|17389     |[0.0,43.0,31300.08]|[0.0,0.1700404858299595,0.12540746393334334]                 |
|12940     |[46.0,4.0,876.29] |[0.12332439678284182,0.012145748987854251,0.01819712791732512]|
|13623     |[30.0,7.0,672.44] |[0.08042895442359249,0.024291497975708502,0.017478781288030567]|
|14450     |[180.0,3.0,483.25]|[0.48257372654155495,0.008097165991902834,0.016812095004997765]|
+----------+------------------+--------------------------------------------------------------+
only showing top 5 rows
```

Gambar 13 Normalisasi dengan MinMax Scaler

3. Hasil Penentuan Jumlah *Cluster* Optimal

Untuk melihat *cluster* dengan jumlah K yang paling optimal, maka dilakukan percobaan sebanyak 3 kali untuk membandingkan nilai *silhouette* yang paling baik dihasilkan dengan meng-assign nilai K = 3, K = 4, dan K = 5. Setelah dilakukan percobaan, maka didapatkan hasil dimana apabila menggunakan nilai K = 3 sebagai jumlah *cluster*, maka nilai *silhouette* yang diperoleh adalah 0.8045. Sehingga berdasarkan skema penggunaan *silhouette*, nilai *silhouette* yang paling mendekati 1 merupakan nilai silhouette yang teroptimal yakni *cluster* sebanyak 3 buah.



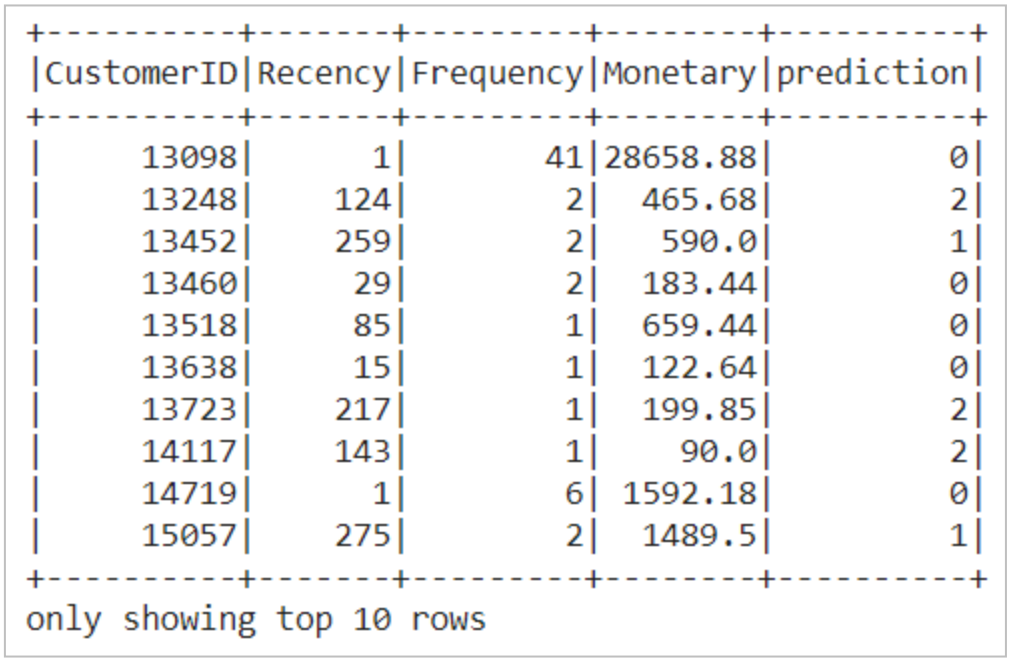
```
Silhouette with squared euclidean distance = 0.8045154385557955
Cluster Centers:
[0.08223703 0.02240483 0.02410349]
[0.79062677 0.00226695 0.01653601]
[0.40872322 0.00631499 0.01759697]
```

Gambar 14 Nilai Silhouette

4. Hasil Penentuan Jumlah *Cluster* Optimal

Setelah dilakukan transformasi data, normalisasi data dan penentuan jumlah *cluster* yang paling optimal, maka kemudian akan dilanjutkan proses implementasi clusterisasi dengan menggunakan algoritma K-Means. Dengan men-assign nilak K = 3, maka

model akan di *fit* sehingga akan dihasilkan transformasi berupa prediksi. Berikut merupakan hasil *cluster* yang didapat dengan menggunakan K-Means.



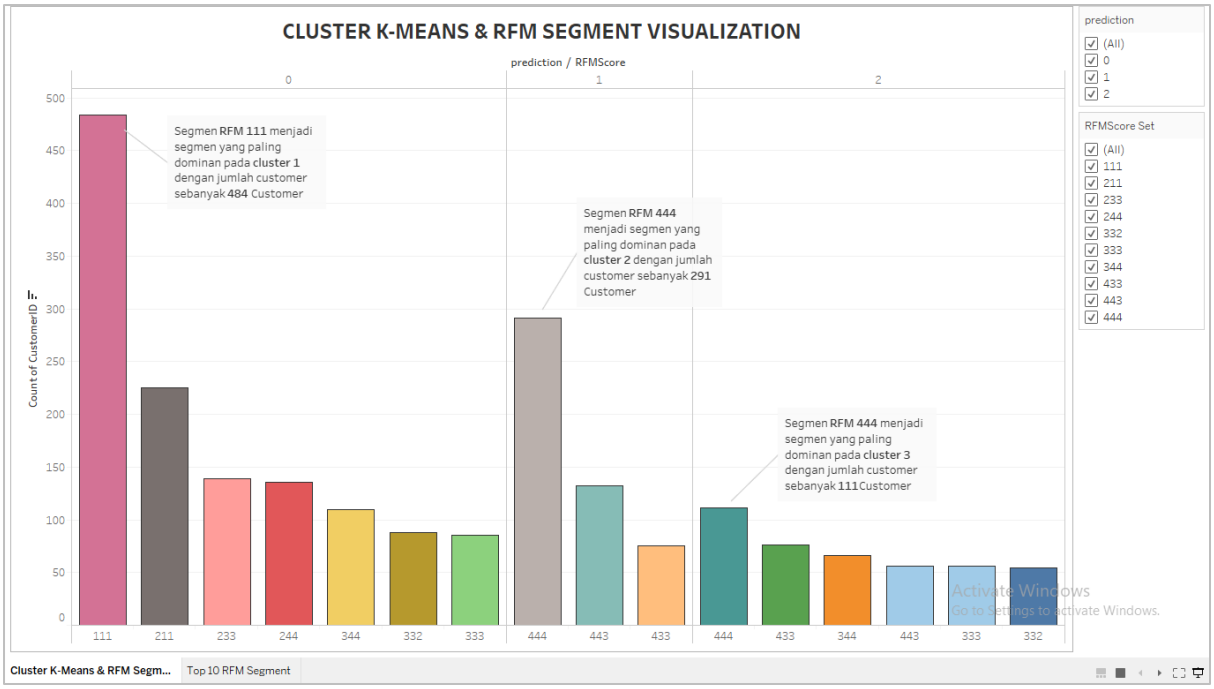
CustomerID	Recency	Frequency	Monetary	prediction
13098	1	41	28658.88	0
13248	124	2	465.68	2
13452	259	2	590.0	1
13460	29	2	183.44	0
13518	85	1	659.44	0
13638	15	1	122.64	0
13723	217	1	199.85	2
14117	143	1	90.0	2
14719	1	6	1592.18	0
15057	275	2	1489.5	1

only showing top 10 rows

Gambar 15 Hasil Cluster dengan K-Means

4.5 Hasil Implementasi Visualisasi Segmen RFM dan K-Means *Clustering*

Visualisasi yang dibuat dengan menggunakan Tableau terdiri dari dua visualisasi, visualisasi yang pertama menampilkan hasil *clustering* dengan RFM segment yang dibagi menjadi tiga kluster yaitu ***Cluster* 1 (*Prediction* 0)**, ***Cluster 2 (Prediction 1),*** dan ***Cluster 3 (Prediction 2)***. Pada setiap kluster dimuat beberapa segmen RFM yang diurutkan berdasarkan segmen yang paling dominan hingga segmen yang paling tidak dominan (namun hanya ditampilkan beberapa segmen RFM pada visualisasi untuk memudahkan mendapatkan informasi). Visualisasi yang dibuat disertai dengan 2 filter yaitu filter untuk prediction dan filter untuk RFMScore, dimana filter ini diharapkan dapat membantu user dalam melihat informasi apa yang dibutuhkan. Visualisasi yang pertama dapat dilihat pada gambar berikut ini.

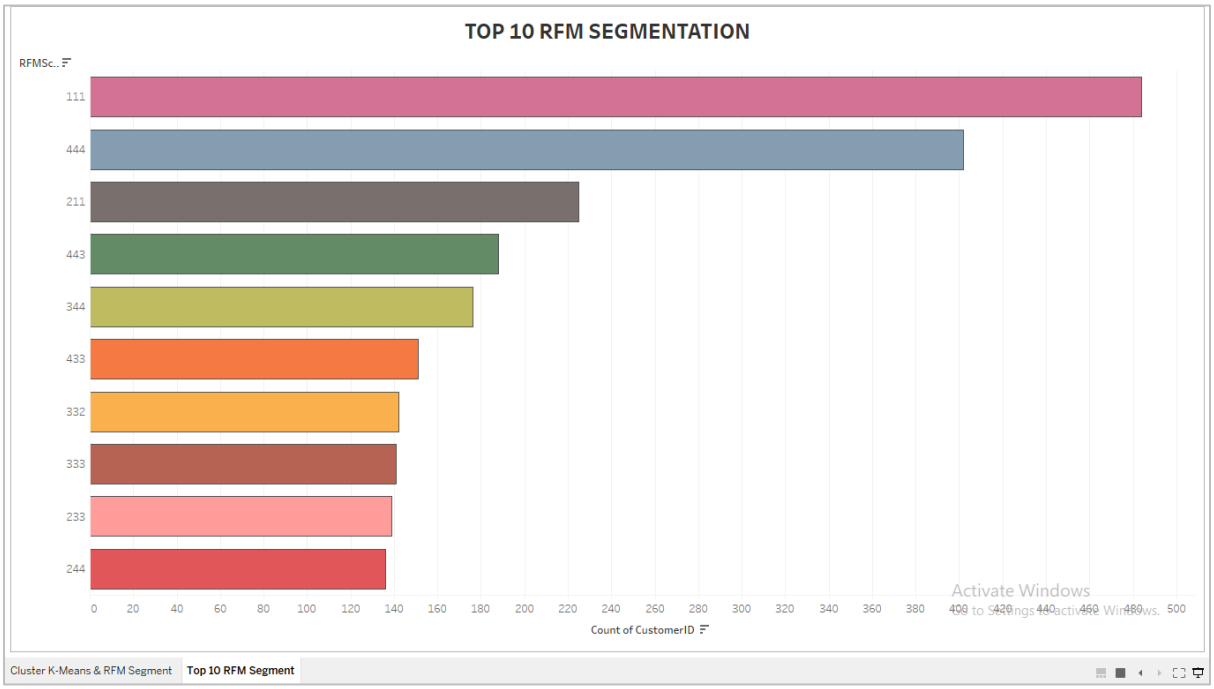


Gambar 16 Visualisasi Cluster K-Means dan Segmen RFM dengan Tableau

Dari visualisasi yang ditampilkan, dapat disimpulkan dengan menggunakan segmentasi RFM dan juga algoritma cluster K-Means, diperoleh 3 buah *cluster* dimana pada ***Cluster* 1 (*Prediction* 0),** segmen 111 menjadi segmen yang paling dominan dengan jumlah *customer* sebanyak 484 *customer*. Segmen 111 merepresentasikan bahwa rata-rata durasi sejak pembelian terakhir (Recency) *customer* adalah lebih kecil atau sama dengan 16, jumlah berapa total pembelian (Frequency) *customer* berbelanja adalah lebih kecil atau sama dengan 1 kali, dan rata-rata total uang yang dibelanjakan (Monetary) *customer* adalah sebesar 4287.63. Pada ***Cluster 2 (Prediction 1),*** segmen 444 menjadi segmen yang paling dominan dengan jumlah *customer* sebanyak 291 *customer*. Segmen 444 merepresentasikan bahwa rata-rata durasi sejak pembelian terakhir (Recency) *customer* adalah lebih besar dari 143, jumlah berapa total pembelian (Frequency) *customer* berbelanja adalah lebih besar atau sama dengan 5 kali, dan rata-rata total uang yang dibelanjakan (*Monetary*) *customer* adalah lebih besar dari 1611.72. Sama dengan Cluster 2, pada ***Cluster 3 (Prediction 2)*** segmen 444 menjadi segmen yang paling dominan dengan jumlah *customer* sebanyak 111 *customer*. Segmen 444 merepresentasikan bahwa rata-rata durasi sejak pembelian terakhir (Recency) *customer* adalah lebih besar dari 143, jumlah berapa total pembelian (Frequency) *customer*

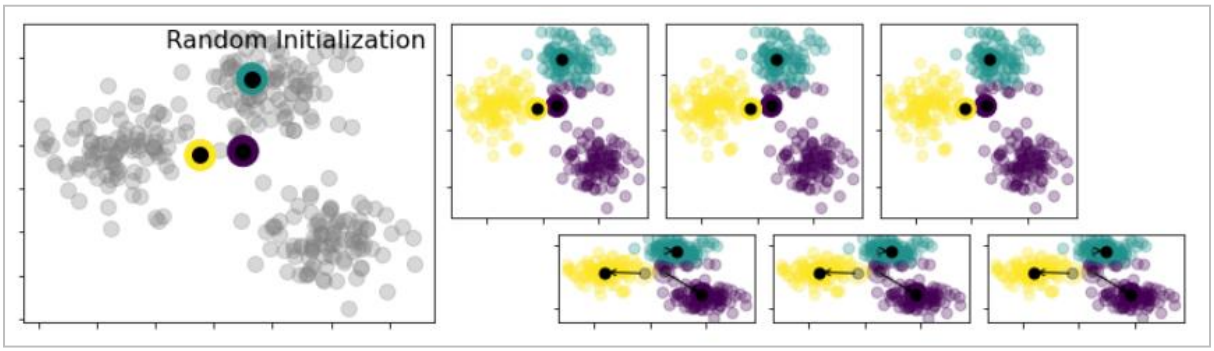
berbelanja adalah lebih besar atau sama dengan 5 kali, dan rata-rata total uang yang dibelanjakan (*Monetary*) *customer* adalah lebih besar dari 1611.72.

Visualisasi dengan menggunakan Tableau yang kedua adalah menemukan trend 10 segmen RFM dengan jumlah *customer* terbanyak secara *general.* Dapat dilihat melalui visualisasi berikut bahwa Segmen RFM 111 merupakan segmen dengan jumlah populasi *customer* terbanyak.



Gambar 17 Visualisasi Top 10 Segmen RFM

Selanjutnya adalah visualisasi persebaran data melalui gambaran *scatter plot* yang menggunakan *library* dari pandas. Berikut merupakan gambaran ketiga *cluster* apabila digambarkan dengan plot.



Gambar 18 Visualisasi Cluster dengan Scatter Plot

REFERENSI

[1] W. Aprilius, "Big Data dan Perawatan Kesehatan Studi Awal Menuju Perawatan kesehatan Masa Depan," *ULTIMA InfoSys*, vol. VI, no. 1, pp. 64-70, 2015.

[2] R. Widyan, "Platfrom Visualisasi Data Untuk Pemerintah Amsterdam Sebagai Solusi Pembersihan Kota Secara Efektif," 2017.