University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING
Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science (ECECS)

TECHNICAL REPORT

Spring - 2024

Contents

Summary

Executive Summary

Using RFM (Recency, Frequency, Monetary) analysis for customer segmentation is an effective way to separate out various consumer categories according to how they transact. Businesses may process huge volumes of transactional data efficiently by using Spark for scalable and distributed RFM score calculation, and Azure Data Factory for data collection, preparation, and orchestration. Consumers can be divided into groups like High-Value, At-Risk, and Low-Value consumers after RFM ratings are determined. This enables firms to adjust their marketing plans and product offerings appropriately. In order to better facilitate the implementation of customized marketing campaigns aiming at enhancing consumer engagement and retention, Azure Data Factory may further coordinate the analysis of each segment's attributes and behavior. By using this strategy, companies may maximize client lifetime value and provide individualized experiences that improve overall marketing and sales effectiveness.

Team Members

- **Siddhant** Alhat Rajendra
- **Veda Samohitha** Chaganti
- **Krishna Chaitanya** Vutukuru
- **Sathish Gandhi** Parasuram Reddy

Title

Title of Project

Smart Segmentation

Harnessing Data Engineering for Customer Segmentation

Highlights of Project

Using both Azure and Databricks to develop our data pipeline, the RFM (Recency, Frequency, Monetary) analysis project was essential to improving our understanding of customer behavior and preferences. With the help of Databricks' potent analytics platform and Azure's sturdy architecture, we were able to gather, prepare, and analyze transactional data with ease, enabling thorough insights. Using the scalable computing power of Databricks, we were able to determine important aspects of consumer behavior by calculating RFM scores. Future development prospects encompass investigating sophisticated segmentation strategies and incorporating supplementary data sources to achieve a more all-encompassing perspective of client conduct. In conclusion, the RFM analysis project—which is made possible by Azure and Databricks—represents a crucial step in the direction of maximizing marketing and customer relationship management tactics, which will promote company expansion and success.

Submitted on: 22nd April 2024

Introduction

Customer segmentation helps with focused marketing and customer satisfaction by grouping customers according to common characteristics. This project collects, saves, and analyzes consumer data at scale using Apache Spark distributed data processing and Azure cloud architecture. Businesses are able to improve customer experiences, optimize resources, and refine marketing tactics through the use of Spark's clustering algorithms, which segment customers.

What is RFM?

Three metrics are used by RFM, a customer segmentation technique, to assess the behavior of its customers: monetary, frequency, and recency. It considers the amount, frequency, and recentness of a customer's purchases. Through the integration of these variables, companies can discern advantageous consumer groups, facilitating focused advertising, customized correspondence, and efficient resource distribution to enhance customer involvement and allegiance.

Importance of RFM Analysis

- **Widely adopted:** Major companies across industries use RFM analysis for marketing campaigns, customer relationship management, and business decision-making.
- **Benefits for businesses:** RFM analysis helps businesses identify high-value customers, tailor marketing strategies, improve customer retention, and optimize resource allocation.

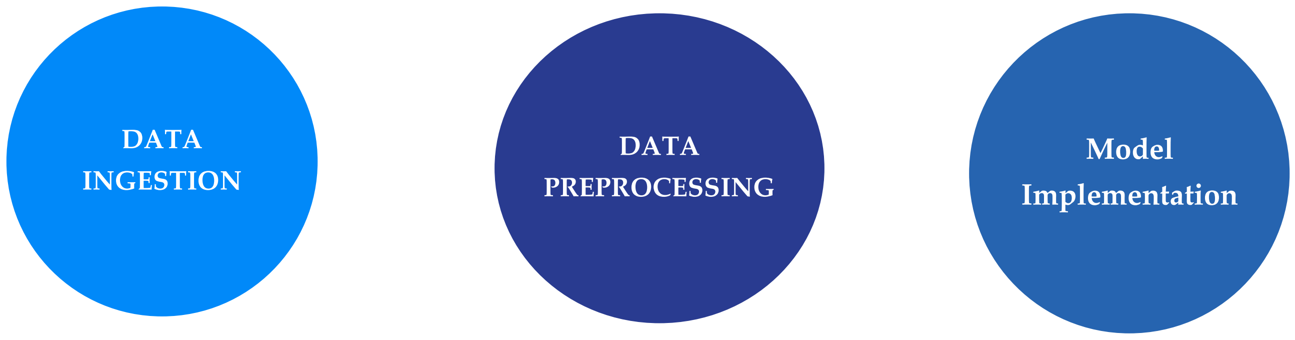
Data Pipeline

Dataset

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company sells unique all-occasion gifts. Many customers of the company are wholesalers.

Data Pipeline

We used Azure to create our data pipeline since it provided a stable platform that met our demands. Utilizing Azure's services improved our productivity and gave us dependable resources for efficient data management. We could leverage Azure's scalability and integration features to make sure our data pipeline was effective and flexible enough to change as our needs did.

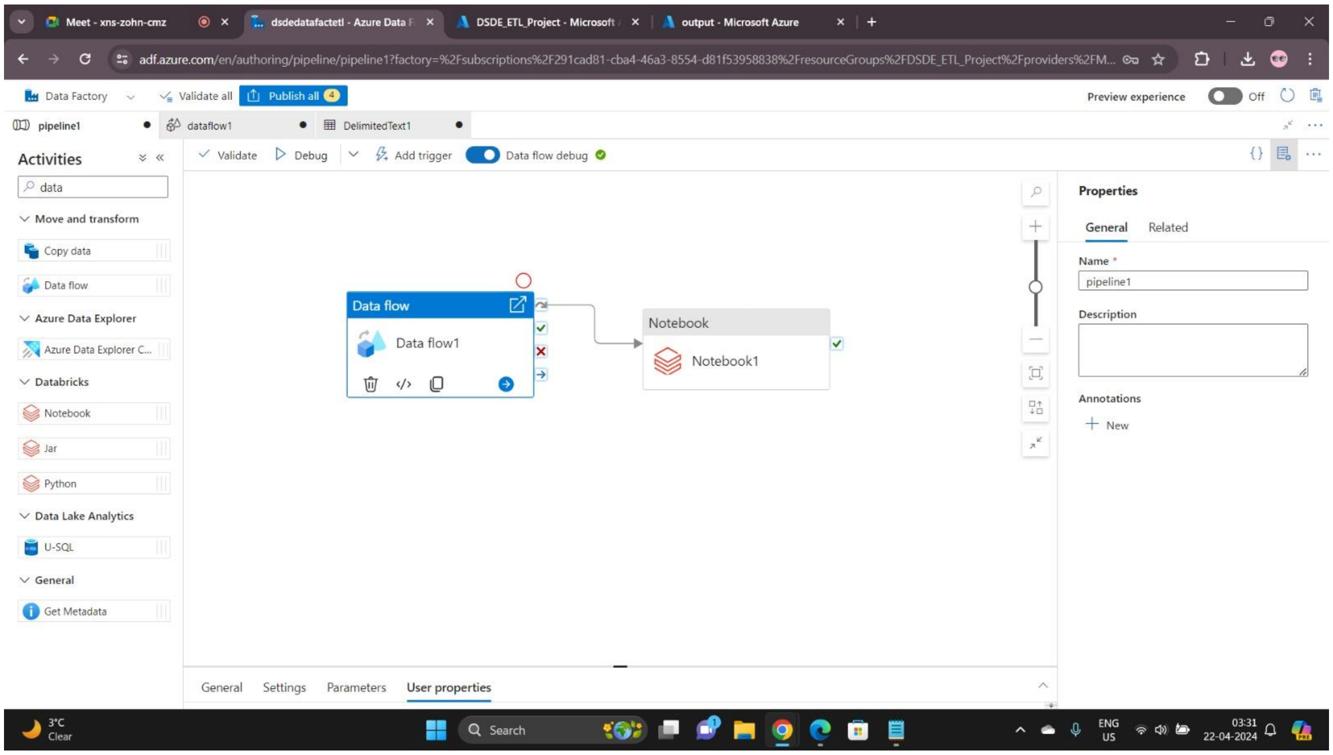


Data ingestion through GA tags (Extracting the data)

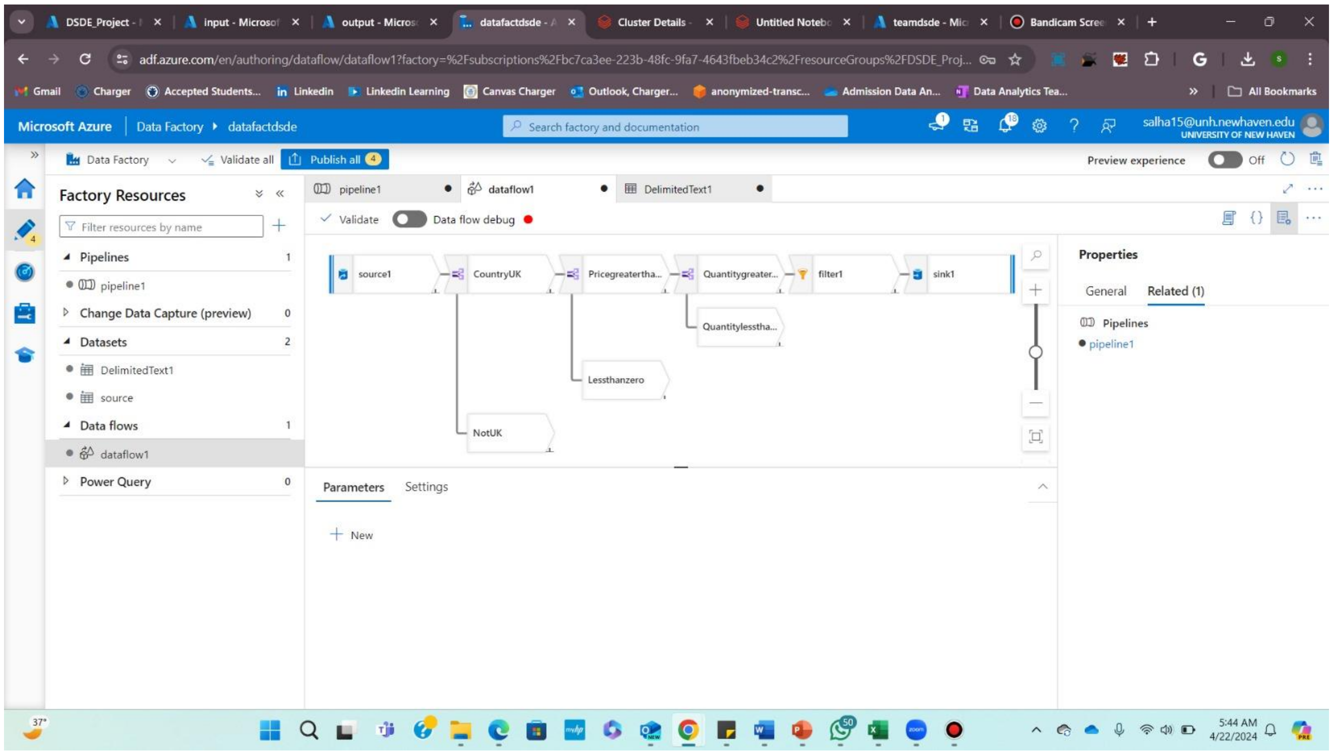
Data preprocessing in Azure

Calculating RFM score in Databricks and implementing spark

Data Pipeline



Name: pipeline1



Methodology

CRISP - DM

We used the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology for our study since it offered an organized way to direct our analysis. We used CRISP-DM in the following ways for our RFM study project.

Business Understanding: During the first stage, we concentrated on fully comprehending the dataset as well as the goals of our study. we carefully examined the dataset to understand its composition, organization, and insights. Our goal was to provide significant results that advance the field of marketing analytics by comprehending the context of the information and its applicability to the analysis of consumer behavior.

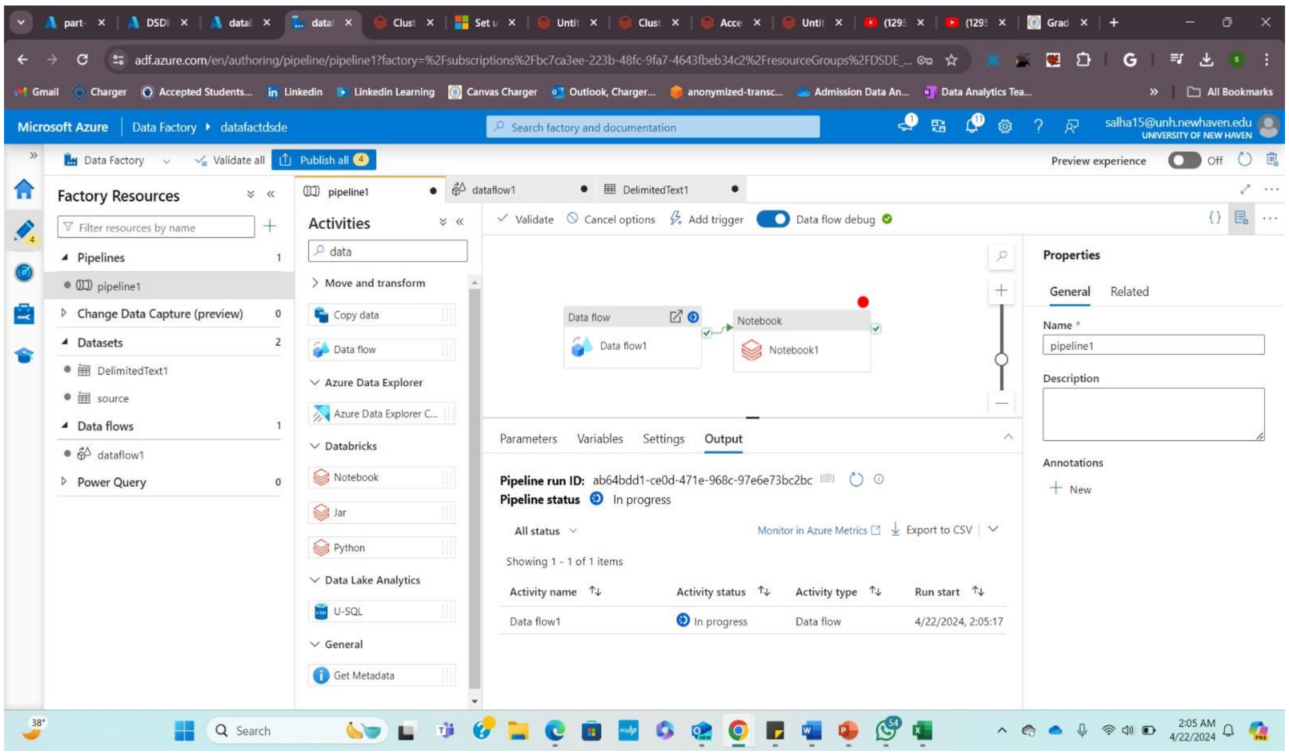
Data Understanding: We then went into detail to comprehend the dataset. To do this, we had to go through the dataset and find its features, their distributions, and any obvious patterns. Additionally, we evaluated the quality of the data, looking for anomalies, missing numbers, and consistency issues. Our choices for data pretreatment and modeling techniques were informed by this in-depth knowledge of the dataset.

Data Preparation: After gaining a thorough grasp of the dataset, we moved on to the data's preparation for analysis. This included fixing mistakes or inconsistencies in the data, changing variables as necessary, and, where relevant, merging data from several sources. Furthermore, to ensure reliable and significant results, we formatted the data in a way that made it ideal for RFM analysis.

Modeling: To begin the modeling stage, we determined each customer's RFM scores. We used methods and algorithms to calculate frequency, recency, and monetary value metrics based on transactional data by utilizing tools like Spark. Effective client segmentation and the discovery of useful insights depended on this stage.

Azure Data Factory

- **Role in basic data processing:** Azure Data Factory was used to construct a pipeline for basic data processing and transformations.
- **Construction of pipeline:** The pipeline included steps for data ingestion, cleansing, and transformation using Azure Data Flow.
- **Key steps in data transformation:** Data was filtered, aggregated, and prepared for further analysis in Databricks.



Pipeline run ID: ab64bdd1-ce0d-471e-968c-97e6e73bc2bc
Pipeline status In progress

All status

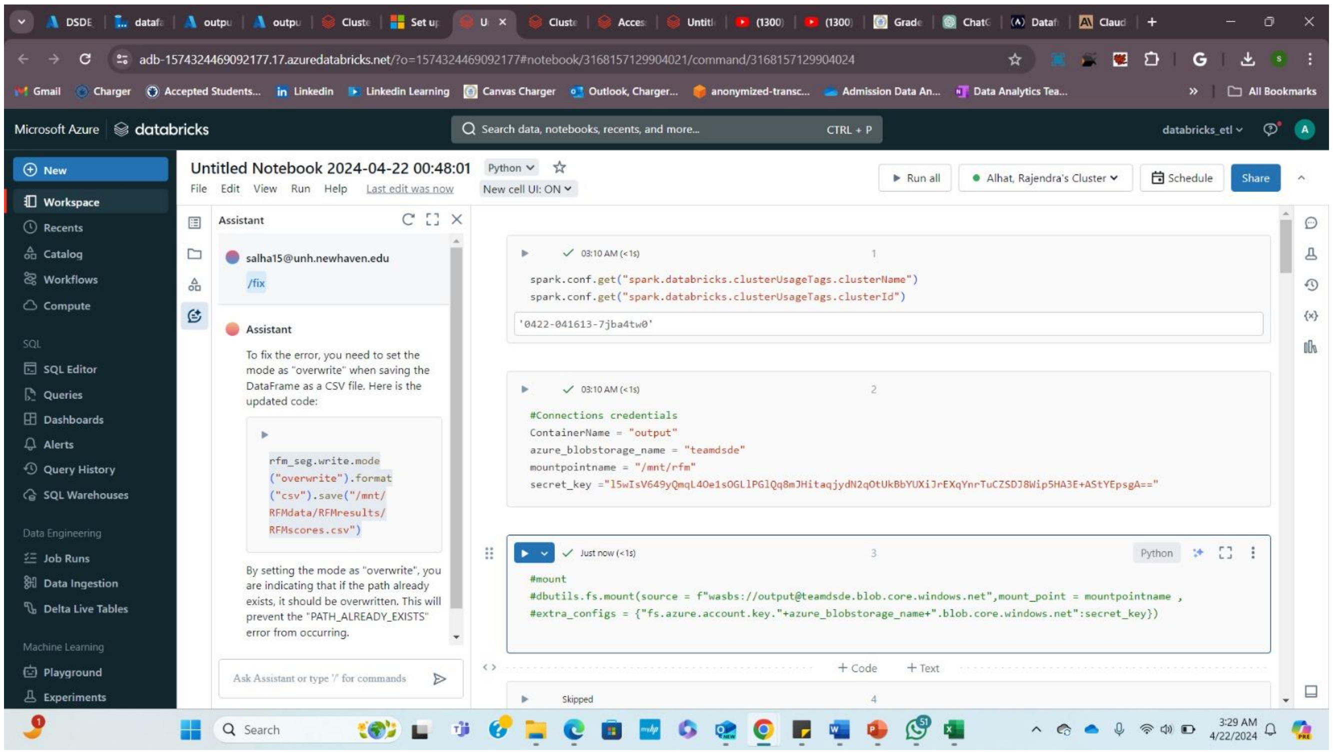
Monitor in Azure Metrics   Export to CSV

Showing 1 - 1 of 1 items

Activity name	Activity status	Activity type	Run start
Data flow1	In progress	Data flow	4/22/2024, 2:05:17

Databricks Integration

- **Utilization of Spark code:** Spark code was implemented in Databricks for advanced data processing tasks.
- **Filtering null values:** Null values were filtered out from the dataset to ensure data quality.
- **Calculating RFM scores:** RFM scores were calculated using Spark code to evaluate the recency, frequency, and monetary value of customer transactions.



Assistant

salha15@unh.newhaven.edu

/fix

Assistant

To fix the error, you need to set the mode as "overwrite" when saving the DataFrame as a CSV file. Here is the updated code:

```
rfm_seg.write.mode
("overwrite").format
("csv").save("/mnt/
RFMdata/RFMresults/
RFMscores.csv")
```

By setting the mode as "overwrite", you are indicating that if the path already exists, it should be overwritten. This will prevent the "PATH_ALREADY_EXISTS" error from occurring.

```
spark.conf.get("spark.databricks.clusterUsageTags.clusterName")
spark.conf.get("spark.databricks.clusterUsageTags.clusterId")
```

```
'0422-041613-7jba4tw0'
```

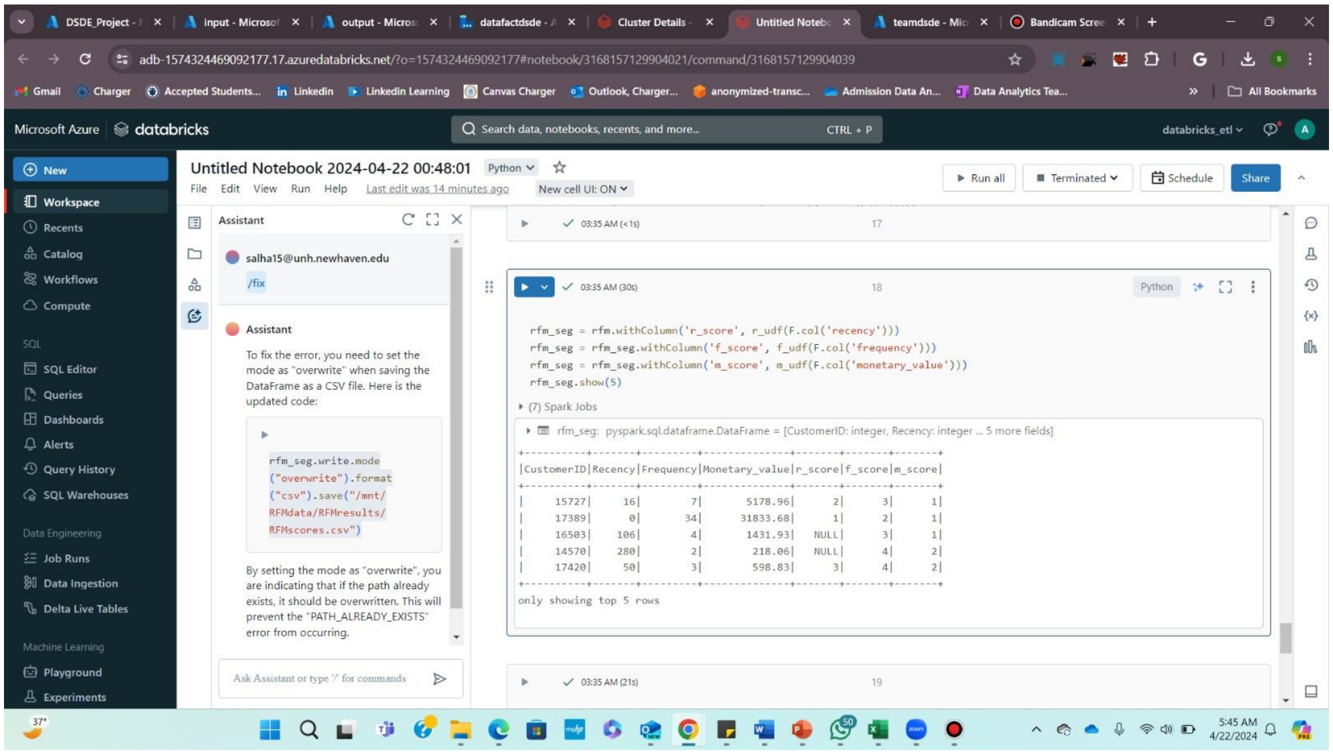
```
#Connections credentials
ContainerName = "output"
azure_blobstorage_name = "teamdsde"
mountpointname = "/mnt/rfm"
secret_key ="l5wIsV649yQmqL4Oe1sOGLlPGlQq8mJHitaqjydN2qOtUkBbYUXiJrEXqYnrTuCZSDJ8Wip5HA3E+AStYEpsgA=="
```

```
#mount
#dbutils.fs.mount(source = f"wasbs://output@teamdsde.blob.core.windows.net",mount_point = mountpointname ,
#extra_configs = {"fs.azure.account.key."+azure_blobstorage_name+".blob.core.windows.net":secret_key})
```

RFM Score

RFM Score Calculation

- **Explanation of RFM score components:** Recency, Frequency, and Monetary (RFM) scores are calculated based on the customer's recent purchase, frequency of purchases, and monetary value of transactions.
- **How RFM scores are calculated:** Each customer is assigned an RFM score based on their transaction history, with higher scores indicating higher value or engagement.



Untitled Notebook 2024-04-22 00:48:01

Assistant

salha15@unh.newhaven.edu

/fix

Assistant

To fix the error, you need to set the mode as "overwrite" when saving the DataFrame as a CSV file. Here is the updated code:

```
rfm_seg.write.mode
("overwrite").format
("csv").save("/mnt/
RFMdata/RFMresults/
RFMscores.csv")
```

By setting the mode as "overwrite", you are indicating that if the path already exists, it should be overwritten. This will prevent the "PATH_ALREADY_EXISTS" error from occurring.

```
rfm_seg = rfm.withColumn('r_score', r_udf(F.col('recency')))
rfm_seg = rfm_seg.withColumn('f_score', f_udf(F.col('frequency')))
rfm_seg = rfm_seg.withColumn('m_score', m_udf(F.col('monetary_value')))
rfm_seg.show(5)
```

```
(7) Spark Jobs
rfm_seg:  pyspark.sql.dataframe.DataFrame = [CustomerID: integer, Recency: integer ... 5 more fields]

+----------+-------+---------+--------------+-------+-------+-------+
|CustomerID|Recency|Frequency|Monetary_value|r_score|f_score|m_score|
+----------+-------+---------+--------------+-------+-------+-------+
|     15727|     16|        7|       5178.96|      2|      3|      1|
|     17389|      0|       34|      31833.68|      1|      2|      1|
|     16503|    106|        4|       1431.93|   NULL|      3|      1|
|     14570|    280|        2|        218.06|   NULL|      4|      2|
|     17420|     50|        3|        598.83|      3|      4|      2|
+----------+-------+---------+--------------+-------+-------+-------+
only showing top 5 rows
```

Code Implementation

This section describes the Databricks code implementation used to conduct RFM (Recency, Frequency, Monetary) analysis. We have streamlined the data ingestion, preparation, and analysis process to extract useful insights from our transactional data by utilizing the capabilities of Databricks. The procedures for determining RFM scores. Our goal with using Databricks is to maximize the effectiveness and scalability of our RFM analysis, which will improve our comprehension of customer segmentation and allow for more focused marketing approaches.

```
#Connections credentials
ContainerName = "output"
azure_blobstorage_name = "teamdsde"
mountpointname = "/mnt/rfm"
secret_key
="l5wIsV649yQmqL4Oe1sOGLlPGlQq8mJHitaqjydN2qOtUkBbYUXiJrEXqYnrTuCZSDJ8Wip5H
A3E+AStYEpsgA=="
```

```
#mount
dbutils.fs.mount(source =
f"wasbs://output@teamdsde.blob.core.windows.net",mount_point =
mountpointname ,extra_configs =
{"fs.azure.account.key."+azure_blobstorage_name+".blob.core.windows.net":se
cret_key})
```

```
DF = spark.read.format('csv').options(
    header='true', inferschema='true').load("/mnt/rfm/*.csv")
```

```
rowscount=DF.count()
colcount=len(DF.columns)
print("Dataframe rows:",rowscount)
print("Dataframe cols:",colcount)
```

```
#Null count
from pyspark.sql.functions import col,isnan, when, count
```

Code Implementation

```
DF.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in
DF.columns]
   ).show()
```

```
DF1=DF.na.drop(subset=["CustomerID"])
DF1.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in
DF1.columns]
   ).show()
```

```
DF1.printSchema()
DF1.show(10)
```

```
#Imports
# #def square(x):
#    return x**2

import pandas as pd
pandasDF = DF1.toPandas()
pandasDF['InvoiceDate']= pd.to_datetime(pandasDF['InvoiceDate'])
pandasDF["Date"]=pd.DatetimeIndex(pandasDF["InvoiceDate"]).date
pandasDF["Month"]=pandasDF["InvoiceDate"].dt.month
pandasDF["Year"]=pandasDF["InvoiceDate"].dt.year
DF2 = spark.createDataFrame(pandasDF)
DF2.show()
```

```
from pyspark.sql.functions import *
from pyspark.sql.types import *
import pyspark.sql.functions as F
from pyspark.sql.types import *
DF3=DF2.withColumn("Totalsales",col("UnitPrice")*col("Quantity"))
Totalsalesbymonth=DF3.groupBy('Description').sum('Totalsales')
Totalsalesbymonth.show()
```

```
date_max = DF3.select(F.max(DF3.InvoiceDate).alias('max_date')).toPandas()
DF4 = DF3.withColumn('Duration', F.datediff(F.lit(date_max.iloc[0][0]),
'InvoiceDate')) #'2011-12-04 13:15:00'
DF4.show(5)
```

Code Implementation

```
recency = DF4.groupby('CustomerID').agg(F.min('Duration').alias('Recency'))
recency.sort(col('Recency')).show()
```

```
frequency = DF4.groupby('CustomerID', 'InvoiceDate').count()\
                        .groupby('CustomerID')\
                        .agg(F.count("*").alias("Frequency"))
frequency.orderBy(frequency.Frequency.desc()).show(5)
```

```
monetary = DF4.groupby('CustomerID').agg(F.round(F.sum('Totalsales'),
2).alias('Monetary_value'))
monetary.orderBy(F.col('Monetary_value').desc()).show(5)
```

```
rfm = recency.join(frequency, on = 'CustomerID', how = 'inner')\
             .join(monetary, on = 'CustomerID', how = 'inner')
rfm.show(5)
```

```
rfm.where(rfm.Recency.contains(0)).show()
```

```
def r_score(r_value):
    """recency with lower value ranks the least, i.e the
    more recent a user transaction, the lower the rank"""

    if r_value <= 14:    # 2 weeks
        return 1
    elif r_value <= 31:   # 1 month
        return 2
    elif r_value <= 93:    # 3 month
        return 3
    else:
        return

def f_score(f_value):
    """the more a customer transact with the business,
    the more their rank is closer to 1, the fewer the number of
    times they have made transaction, the higher the rank"""

    if f_value <= 3:
        return 4
```

Code Implementation

```
    elif f_value <= 18:
        return 3
    elif f_value <= 36:
        return 2
    else:
        return 1


def m_score(m_value):
    """the more the value contributed to the business,
    the higher their rank to 1"""

    if m_value < 10:
        return 4
    elif m_value <= 100:
        return 3
    elif m_value <= 1000:
        return 2
    else:
        return 1

r_udf = F.udf(lambda r_value: r_score(r_value), StringType())
f_udf = F.udf(lambda f_value: f_score(f_value), StringType())
m_udf = F.udf(lambda m_value: m_score(m_value), StringType())
```

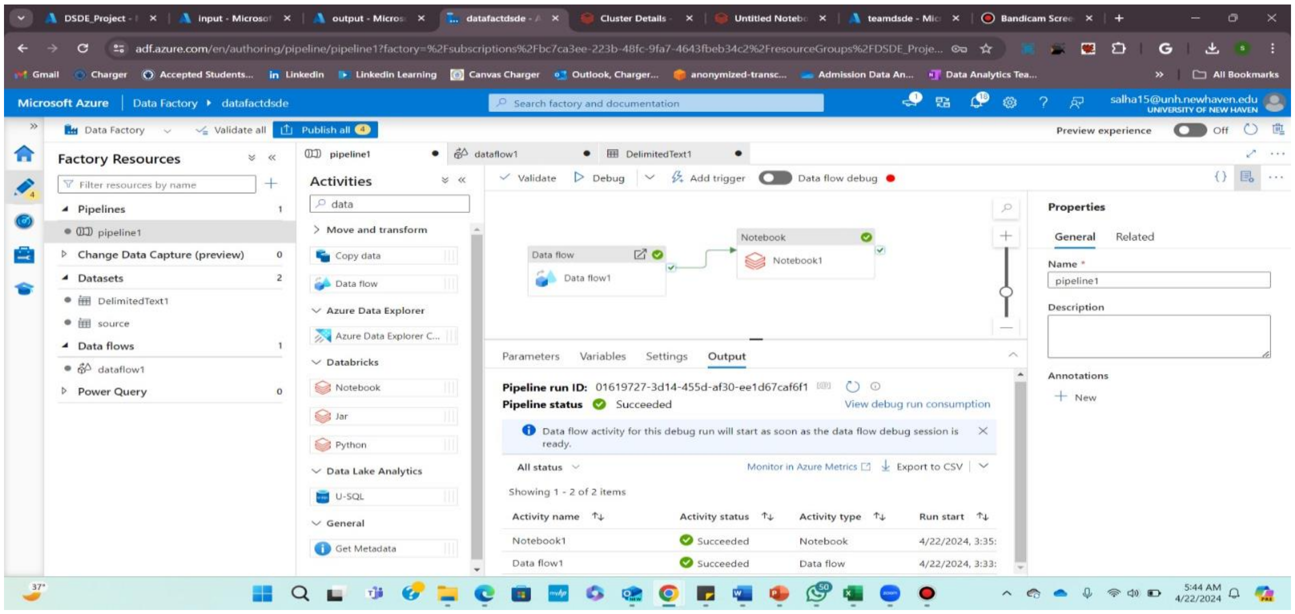
```
rfm_seg = rfm.withColumn('r_score', r_udf(F.col('recency')))
rfm_seg = rfm_seg.withColumn('f_score', f_udf(F.col('frequency')))
rfm_seg = rfm_seg.withColumn('m_score', m_udf(F.col('monetary_value')))
rfm_seg.show(5)
```

```
rowscount=rfm_seg.count()
colcount=len(rfm_seg.columns)
print("Dataframe rows:",rowscount)
print("Dataframe cols:",colcount)
```

```
rfm_seg.write.mode("overwrite").format("csv").save("/mnt/RFMdata/RFMresults
/RFMscores.csv")
```

Results

The use of Azure to create our data pipeline produced important insights, most notably RFM scores. Through improved data analysis, we produced RFM (Recency, Frequency, Monetary) scores that offer a more comprehensive view of consumer behavior and preferences. One important outcome of our Azure-based data pipeline solution is the ability to compute RFM scores, which shows the real-world advantages of using advanced data analytics methods in our project.



Parameters | Variables | Settings | Output

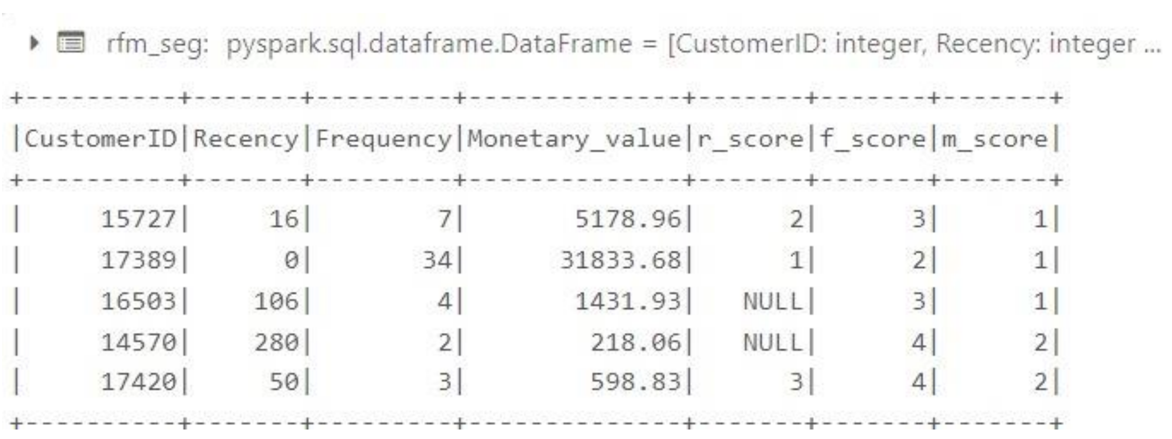
Pipeline run ID: 01619727-3d14-455d-af30-ee1d67caf6f1
Pipeline status Succeeded

Data flow activity for this debug run will start as soon as the data flow debug session is ready.

All status — Monitor in Azure Metrics — Export to CSV

Showing 1 - 2 of 2 items

Activity name	Activity status	Activity type	Run start
Notebook1	Succeeded	Notebook	4/22/2024, 3:35:
Data flow1	Succeeded	Data flow	4/22/2024, 3:33:



```
rfm_seg:  pyspark.sql.dataframe.DataFrame = [CustomerID: integer, Recency: integer ...

+----------+-------+---------+--------------+-------+-------+-------+
|CustomerID|Recency|Frequency|Monetary_value|r_score|f_score|m_score|
+----------+-------+---------+--------------+-------+-------+-------+
|     15727|     16|        7|       5178.96|      2|      3|      1|
|     17389|      0|       34|      31833.68|      1|      2|      1|
|     16503|    106|        4|       1431.93|   NULL|      3|      1|
|     14570|    280|        2|        218.06|   NULL|      4|      2|
|     17420|     50|        3|        598.83|      3|      4|      2|
+----------+-------+---------+--------------+-------+-------+-------+
```

Insights

Customer Segmentation and Insights

- **Application of RFM scores:** RFM scores are used to segment customers into diverse groups based on their buying behavior.
- **Interpretation of RFM scores:** Businesses can gain insights into customer behavior and preferences, enabling targeted marketing campaigns and personalized customer experiences.
- **Business insights and strategies:** RFM analysis helps businesses identify high-value customers, re-engage dormant customers, and optimize marketing strategies for better ROI.

Conclusion

Our project's goals, which are to improve customer experiences and optimize marketing strategies, depend heavily on consumer segmentation utilizing Spark and Azure. We aim to explore the complexities of RFM analysis, a popular technique for classifying clients based on recency, frequency, and monetary value, by utilizing the power of these innovative technologies. Our project's main goal is to effectively handle large amounts of transactional data by utilizing Azure's cloud infrastructure, which has strong data processing and storage capabilities. In the meantime, real-time analysis and faster calculation times are made possible by Spark's distributed computing framework, which allows for parallel processing across clusters. Our goal is to discover high-value client groupings and adjust marketing campaigns accordingly by using the synergy between Spark and Azure. ultimately generating prospects for corporate growth and profitability.

Through our project, we hope to demonstrate how Azure and Spark can revolutionize the fields of marketing analytics and customer segmentation. Through resource optimization and the utilization of innovative data processing skills, our goal is to show how companies can extract meaningful insights from massive datasets, which can enhance customer happiness and better target marketing campaigns. We aim to enable organizations to uncover new revenue streams, improve competitiveness, and prosper in today's changing marketplace through our thorough analysis and strategic execution.