Customer Profiling in Banking Data

Salma Moustafa [G09BEI] , Sarmad Shaikh [I1I8G4], Tameem Alshomari [H9WHAV]

January , 2021

1 Introduction and Overview

In this project, customer profiling on banking data is implemented. Different datasets provided concerned with banking data related to bank customers are used.

2 Customer Profiling

The objective of our project was to implement a baseline and/or novel algorithm for(offline) customer profiling (i.e.clustering) based on customer behavior in financial systems. Customers are small and medium enterprises with behavior descriptors: logins in certain banking systems, transfers to partners, invoices issued.

3 Data

The data is classified into four tables: static-data, invoice-history, transfer-history and customer-login-history. Table invoice-history has features/columns describing invoices issued by customers (identified by PARTY-ID). transfer-history table includes the features describing transfers done by customers to other entities. For static-data, it includes generic data describing the customer (i.e. Region, date of account creation). Last, customer-login-history includes login times for some customers.

In order to do meaningful customer profiling, we applied data transformations for the features in each table to end up with aggregated features, describing customers' behavior in each of the tables in our data. The transformations pipeline is done in pyspark as described in the following sections.

4 System Architecture

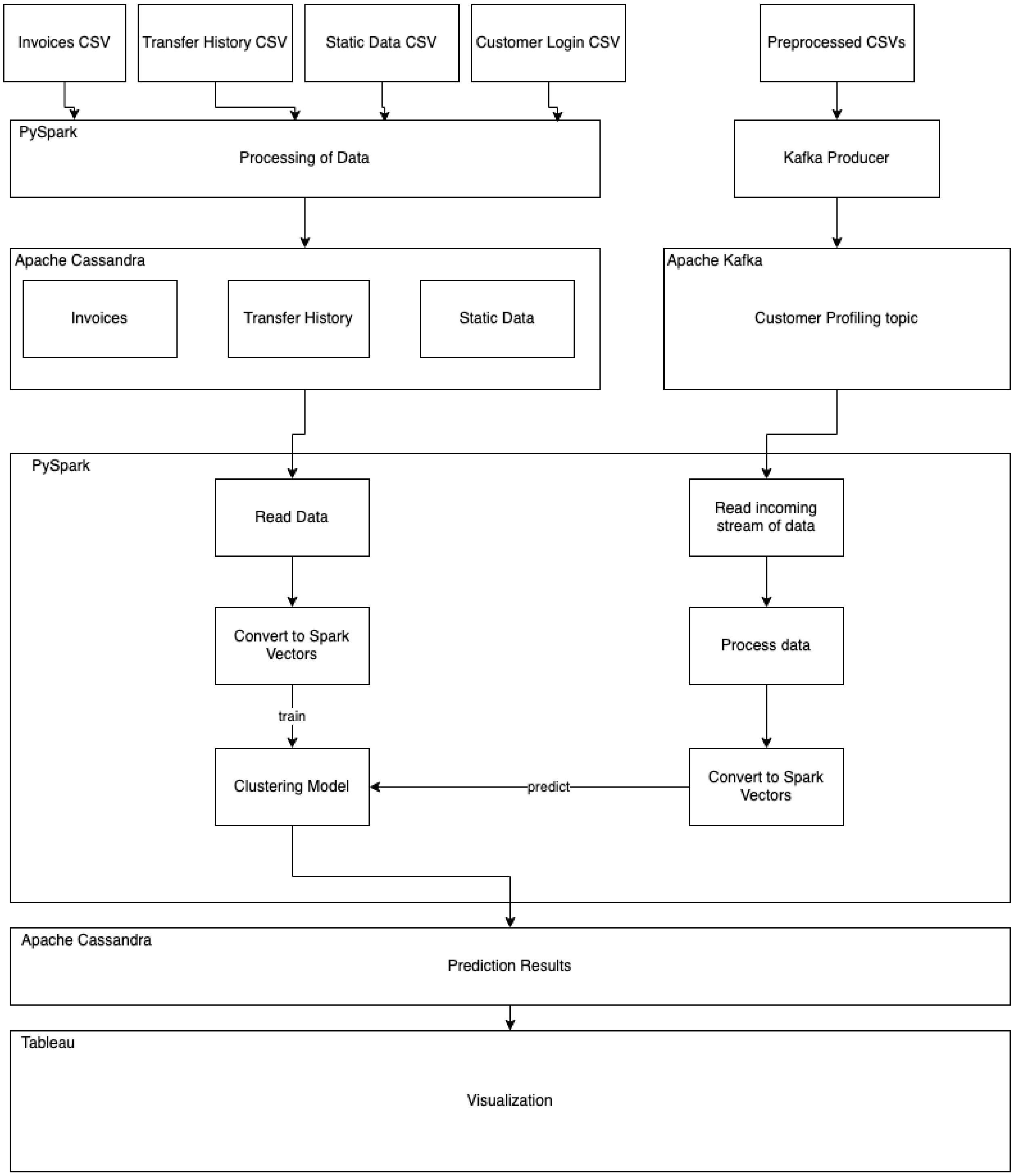


Fig 1. Implemented architecture of the project

5 Pyspark

For all the following, a batch pipeline was established for the data to transform it via a series of applied aggregation functions to the features. The final table in each case has PARTY-ID as the primary key and the aggregated features associated with each PARTY-ID.

5.1 transfer-history

The transfer-history table has the following features:

- PARTY-ID
- TRANSFER-ID
- AMOUNT
- CURRENCY
- VALUE-DATE

Instances were grouped by PARTY-ID. The sum, count, min, max and mean were applied to AMOUNT feature to get the total transfer amounts, average transfer amounts and min/max transfer amounts per customer. CURRENCY was changed to One-Hot-Encoding vector indicating the currencies each customer had their transfers in. Finally, count and countDistinct aggregated functions were used on TRANSFER-ID to count number of transfers and number of unique transfers per customer. The final table have the following features:

- PARTY-ID
- unique-transfers-count
- transfers-count
- currency-vector
- sum(AMOUNT)
- max(AMOUNT)
- min(AMOUNT)
- avg(AMOUNT)

5.2 invoice-history

The invoice-history table has the following features:

- PARTY-ID
- INVOICE-ID
- TAXINCLUSIVEAMOUNT
- CURRENCY
- TAXEXCLUSIVEAMOUNT
- PAYABLEAMOUNT
- ISSUEDATE
- TAXPOINTDATE

- DUEDATE

Instances were grouped by PARTY-ID. The sum, count, min, max and mean were applied to TAXINCLUSIVEAMOUNT and TAXEXCLUSIVEAMOUNT features to get the total, average, min and max amounts (per type of amount) per customer. PAYABLE AMOUNT was excluded from such transformations because PAYABLEAMOUNT column is equal to TAXINCLUSIVEAMOUNT column. CURRENCY was changed to One-Hot-Encoding vector indicating the currencies each customer had their transfers in. INV-PERIOD feature was created as the difference between ISSUEDATE and DUEDATE. Min/max and average invoice periods are calculated per customer afterwards. The final table have the following features:

- PARTY-ID
- min(INV-PERIOD)
- max(INV-PERIOD)
- avg(INV-PERIOD)
- sum(TAXINCLUSIVEAMOUNT)
- max(TAXINCLUSIVEAMOUNT)
- min(TAXINCLUSIVEAMOUNT)
- avg(TAXINCLUSIVEAMOUNT)
- sum(TAXEXCLUSIVEAMOUNT)
- max(TAXEXCLUSIVEAMOUNT)
- min(TAXEXCLUSIVEAMOUNT)
- avg(TAXEXCLUSIVEAMOUNT)
- invoice-count
- currency-vector

5.3 customer-login-history

this table has only one column, with login instances of customers. login-count was created from this table.

- PARTY-ID
- LAST-LOGIN

5.4 static-data

The static-data table has the following features:

- PARTY-ID
- CREATED-ON
- LAST-LOGIN
- LOB-CODE
- SIZE
- PACKAGE
- REGION
- COUNTY

First, missing value for the SIZE , PACKAGE and LOB-CODE Were fixed and processed , then label encoding was applied on COUNTY feature

6 Apache Kafka

We used Apache Kafka as the event streaming system. We created a topic called 'customer-profiling' where we'd publish rows from preprocessed datasets using the Kafka Connect Python API. These rows would be consumed by the streaming pipeline in PySpark, which would then be converted to features and fed to the model for analysis.

7 Cassandra

We use Cassandra in this project to store the processed data from their respective PySpark processing pipelines. From Cassandra, we ingest them into our model training pipeline, combine all the datasets into a single dataset, convert it into features and feed it into our model for training. We also store the output predictions from the model into our database. These tables are used for visualization purpose later.

8 Tableau

8.1 connect Cassandra to Tableau

Connect to Cassandra as an ODBC Data Source :
To create a data source or workbook in Tableau Desktop and publish the data source or workbook to Tableau server, you will need to configure a DSN on each machine (Desktop and Server), specifying connection properties and creating DSNs using the same name on each machine. Information for connecting to Cassandra follows, along with different instructions for configuring a DSN in Windows and Linux environments. Set the Server, Port, and Database connection properties to connect to Cassandra. Additionally, to use internal authentication set the User and Password connection properties. When you configure the DSN, you may also want to set the Max Rows connection property. This will limit the number of rows returned, which is especially helpful for improving performance when designing reports and visualizations. Windows If you are installing the CData ODBC Driver for Cassandra on Windows, DSN configuration is the last step of the driver installation. If you already have the driver installed, or you wish to configure new DSNs, you can use the Microsoft ODBC Data Source Administrator.
Publish the Cassandra Data Source to Tableau Server : With the connections to Cassandra data configured, you are ready to publish a Cassandra data source on Tableau, ready to be leveraged by users in your organization to create workbooks based on Cassandra data. .
Create and Publish a Data Source :

- In the Connect pane, click More -¿ Other Databases (ODBC). Select CData Cassandra Sys, the system DSN.
- The driver installation automatically creates matching user and system DSNs: The system DSN is needed to connect from Tableau Server.
- In the Database menu, select CData.
- In the Table box, enter a table name or click New Custom SQL to enter an SQL query.
- Drag the table onto the join area.
- From the Server menu, click Publish Data Source -¿ (YOUR DATA SOURCE).

8.2 Tableau Software

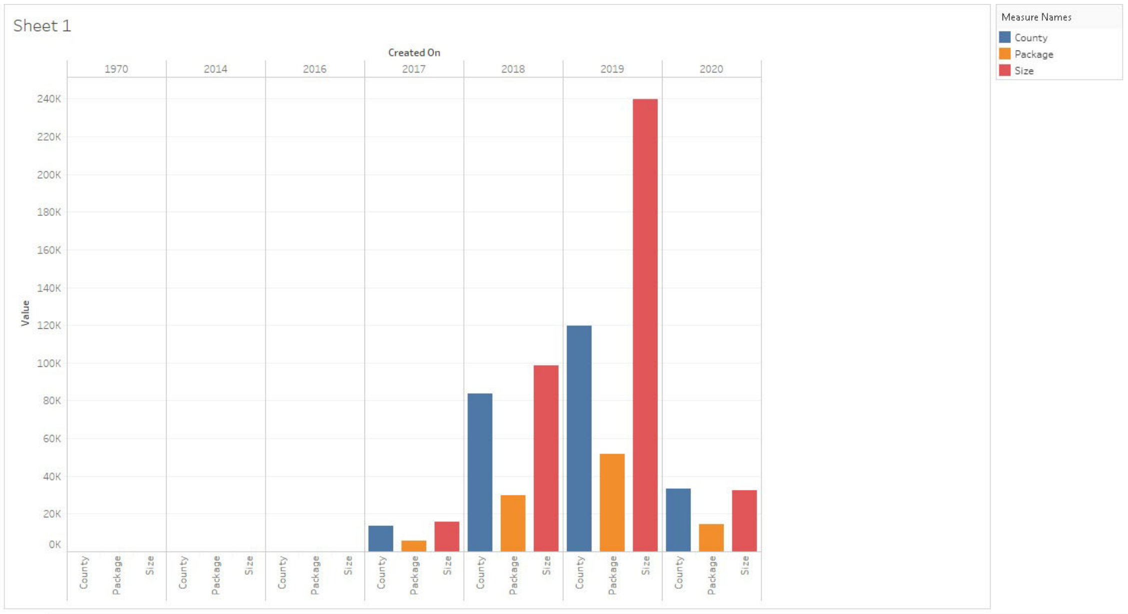
Open Tableau and connect to the Apache Spark server with following settings from the Connect panel: The server IP is the ip address of sparksql thriftserver which may also change depending of your installation. You may also change authentication settings depending of your configuration.

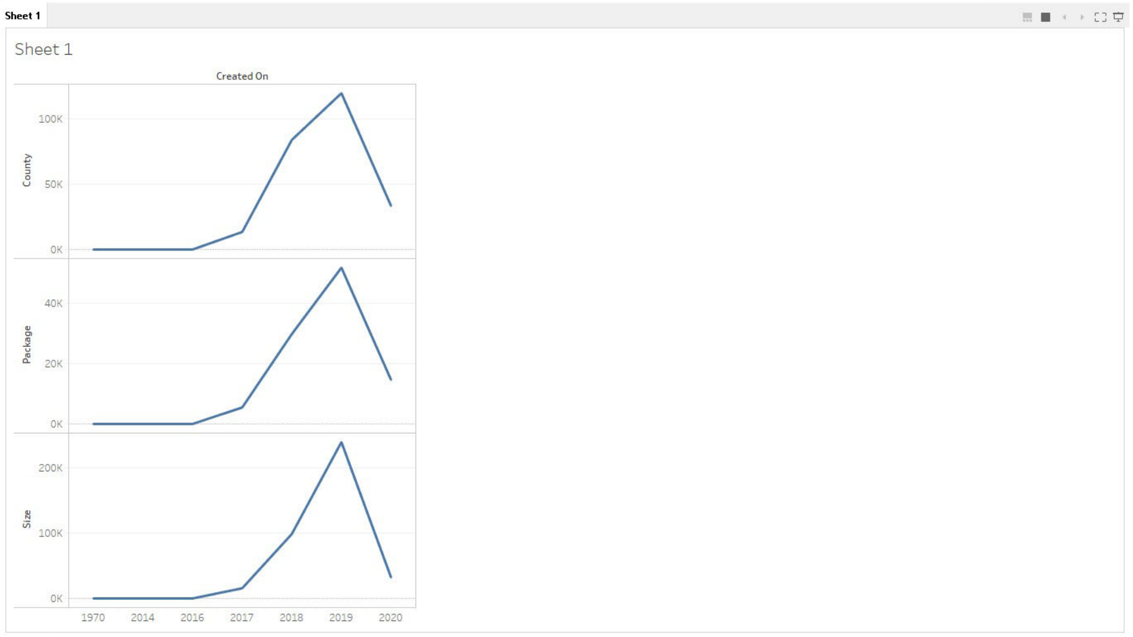
8.3 Cassandra Connection

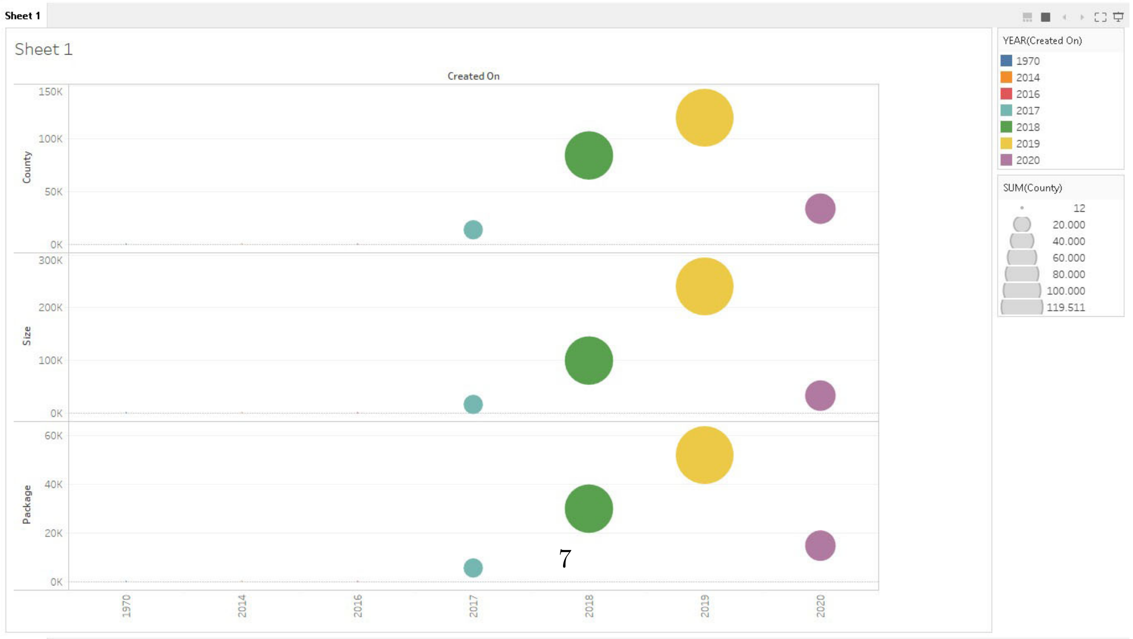
Then you should be able to see all Apache Cassandra keyspaces (named Schema in Tableau interface) and tables (click enter in Schema and Table inputs to see all available Cassandra keyspaces and tables).

9 Tableau Visualization

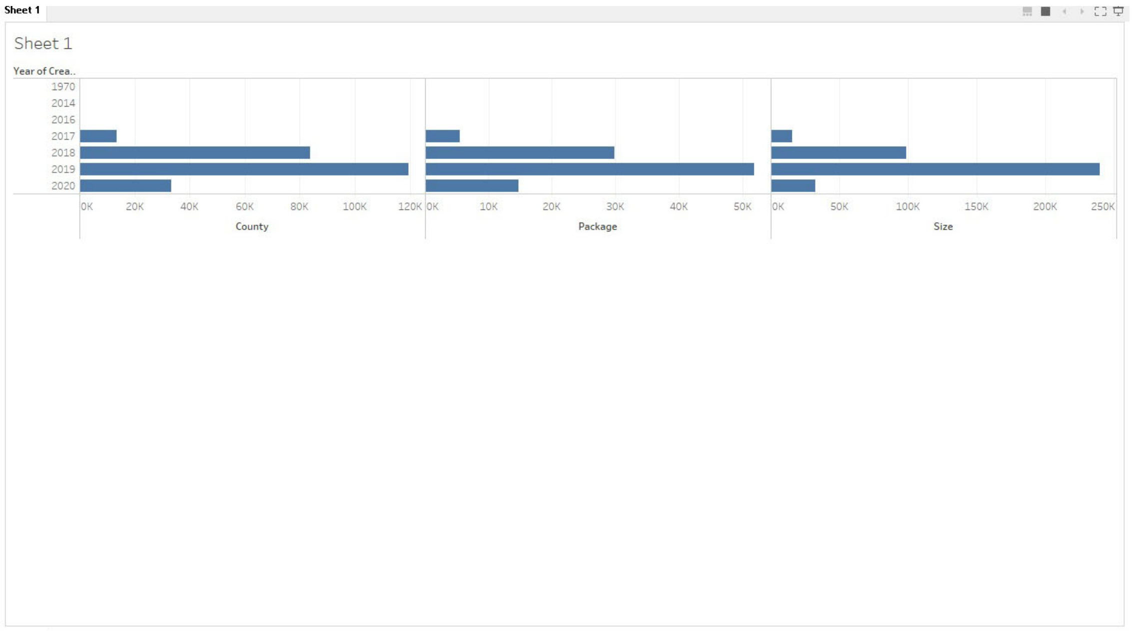
9.1 visualizing data from tables


Sheet 1
Created On
1970
2014
2016
2017
2018
2019
2020
Value
240K
220K
200K
180K
160K
140K
120K
100K
80K
60K
40K
20K
0K
County
Package
Size
Measure Names
🟦 County
🟧 Package
🟥 Size

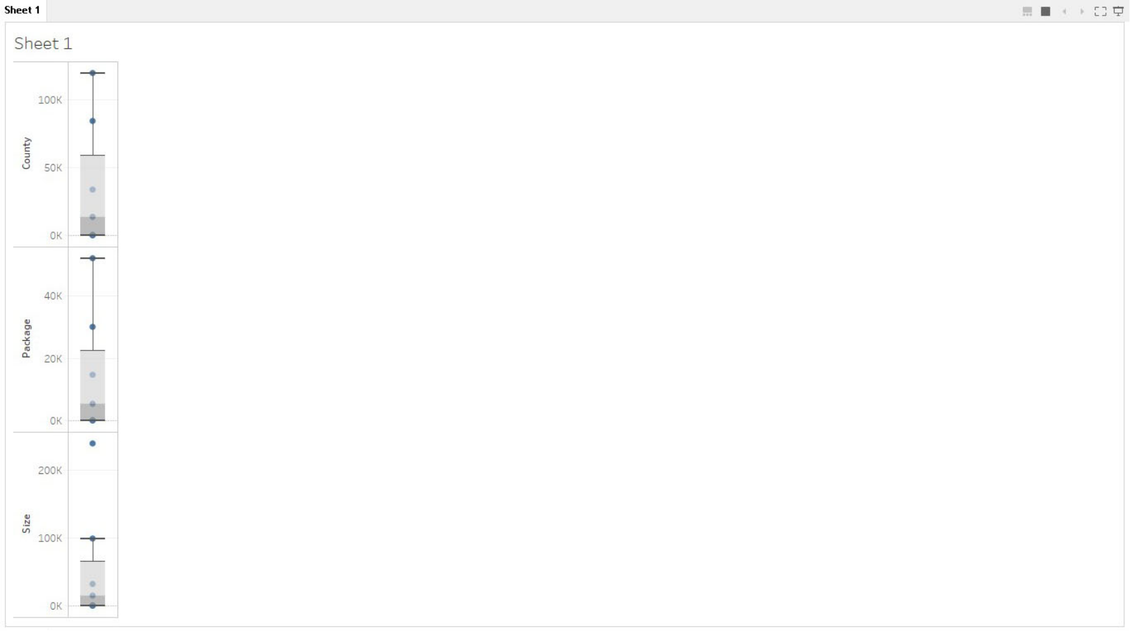

Sheet 1
Sheet 1
Created On
County
100K
50K
0K
Package
40K
20K
0K
Size
200K
100K
0K
1970
2014
2016
2017
2018
2019
2020


Sheet 1
Sheet 1
Created On
County
150K
100K
50K
0K
Size
300K
200K
100K
0K
Package
60K
40K
20K
0K
1970
2014
2016
2017
2018
2019
2020
YEAR(Created On)
🟦 1970
🟧 2014
🟥 2016
🟦 2017
🟩 2018
🟨 2019
🟪 2020
SUM(County)
12
20.000
40.000
60.000
80.000
100.000
119.511

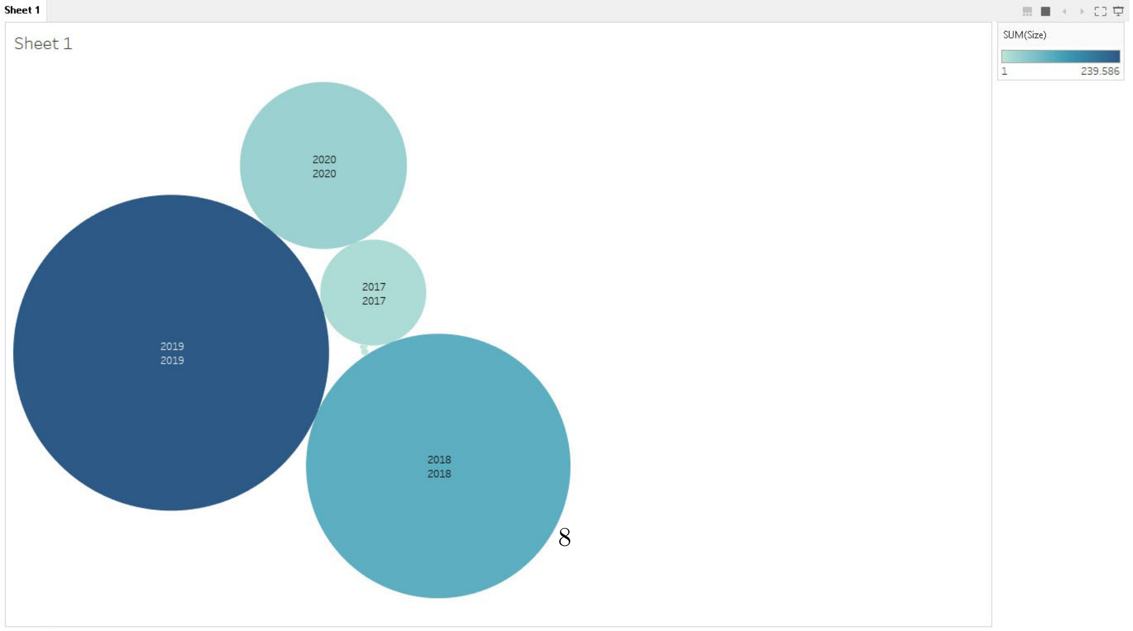
Sheet 1



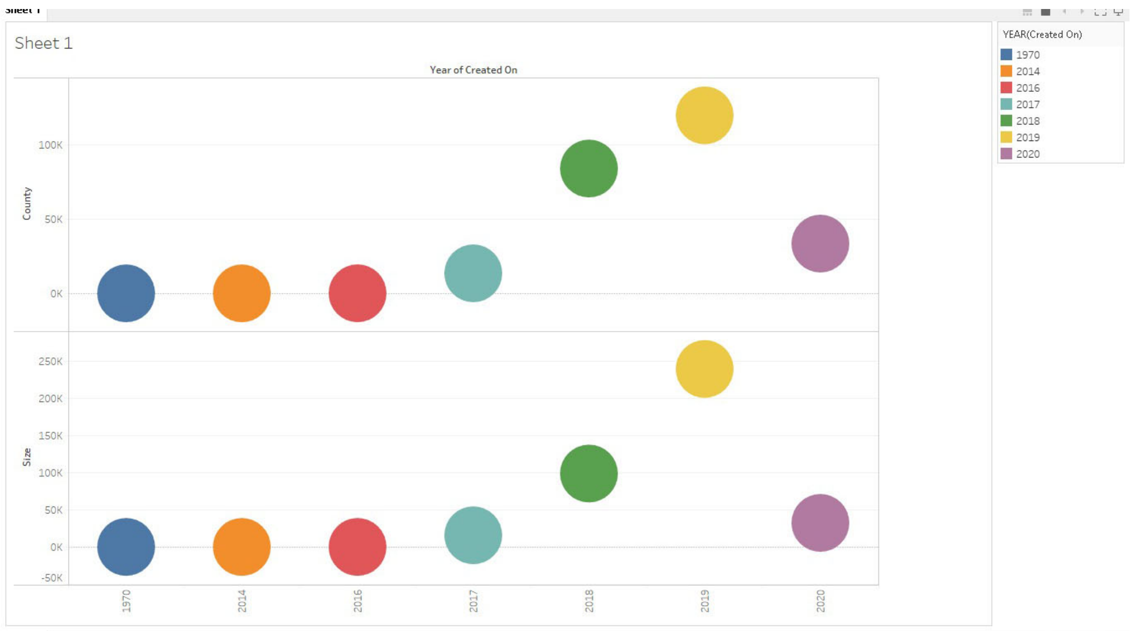
Sheet 1

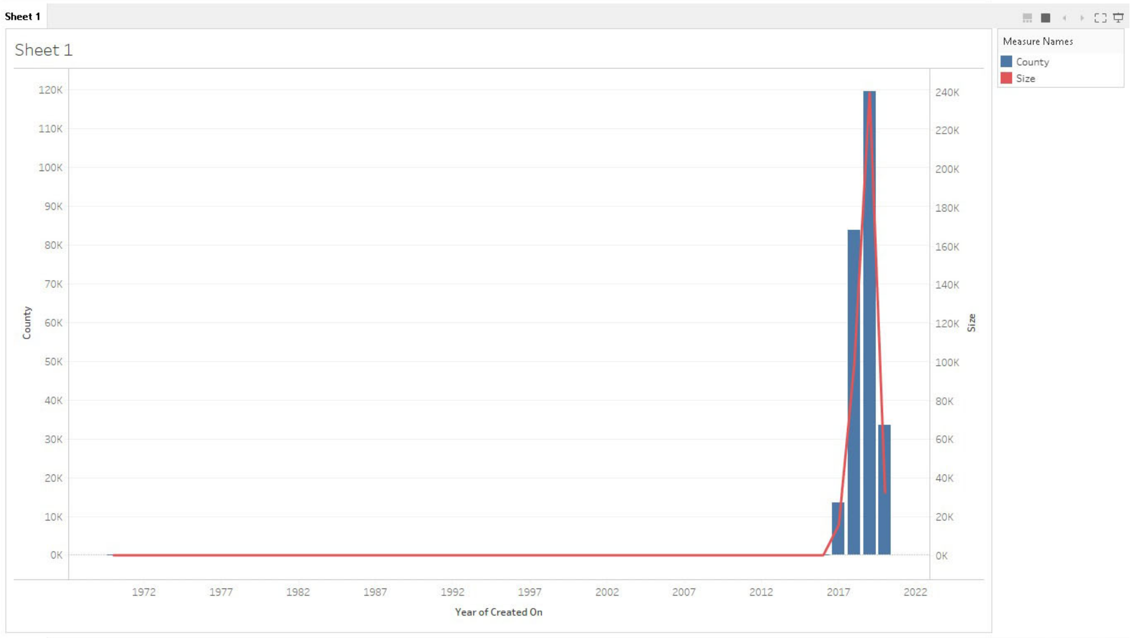


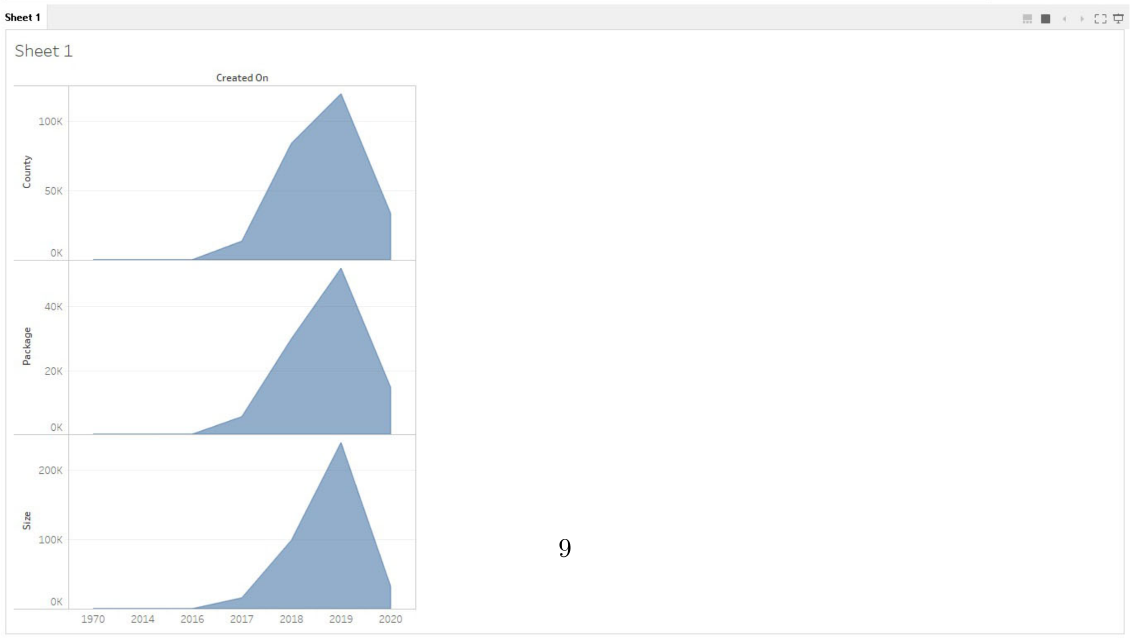
Sheet 1

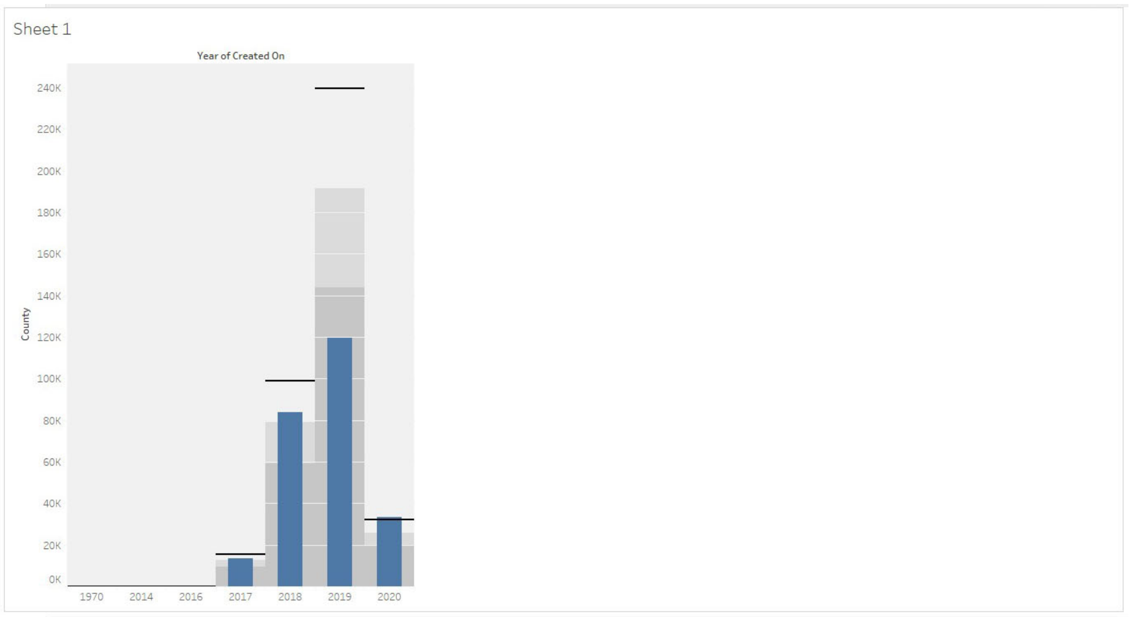


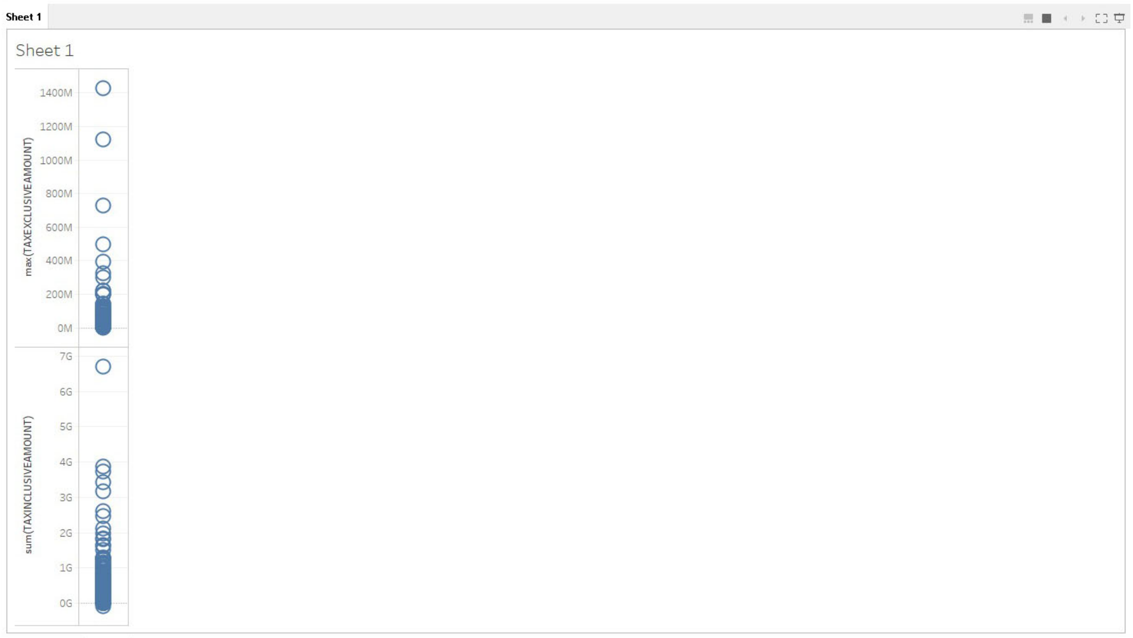
Sheet 1

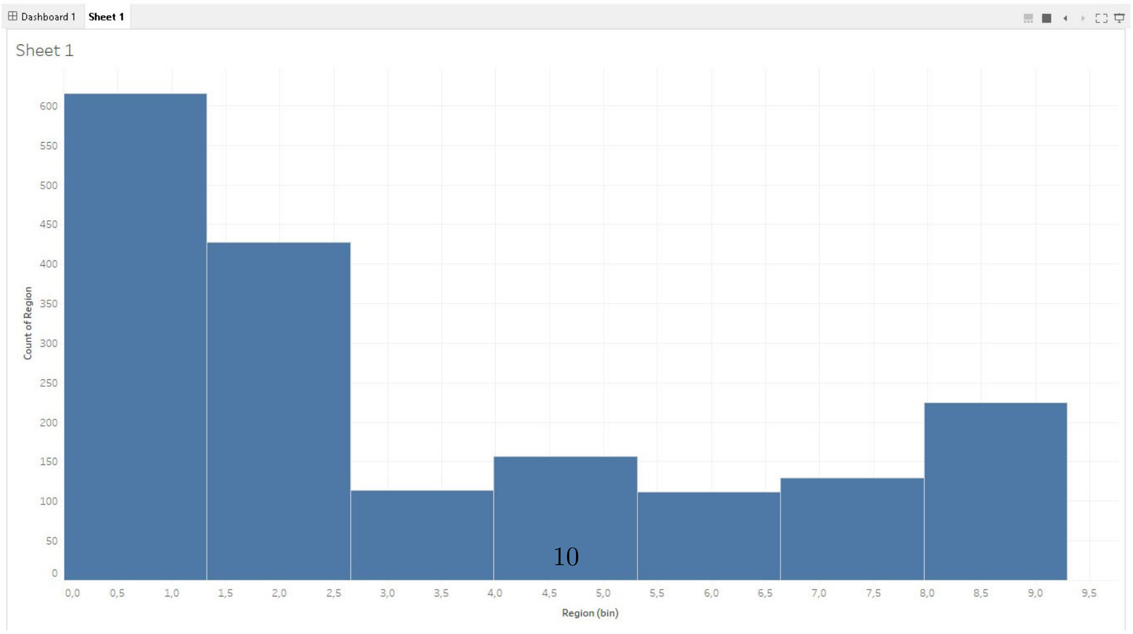

Sheet 1
Sheet 1
Year of Created On
County
Size
100K
50K
0K
250K
200K
150K
100K
50K
0K
-50K
1970
2014
2016
2017
2018
2019
2020
YEAR(Created On)
1970
2014
2016
2017
2018
2019
2020

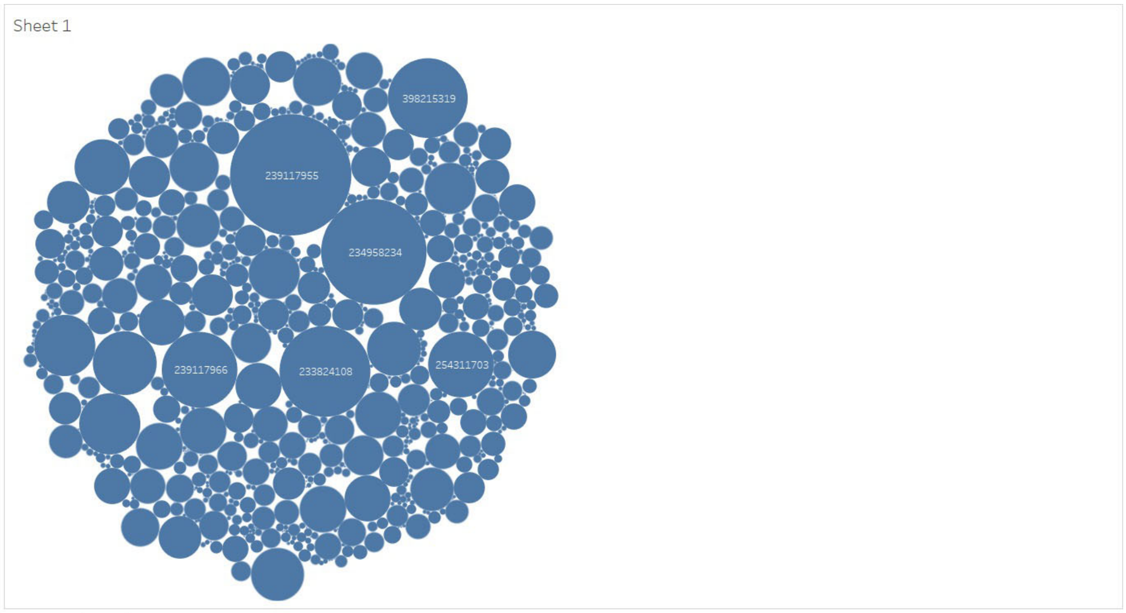

Sheet 1
Sheet 1
County
Size
120K
110K
100K
90K
80K
70K
60K
50K
40K
30K
20K
10K
0K
240K
220K
200K
180K
160K
140K
120K
100K
80K
60K
40K
20K
0K
1972
1977
1982
1987
1992
1997
2002
2007
2012
2017
2022
Year of Created On
Measure Names
County
Size

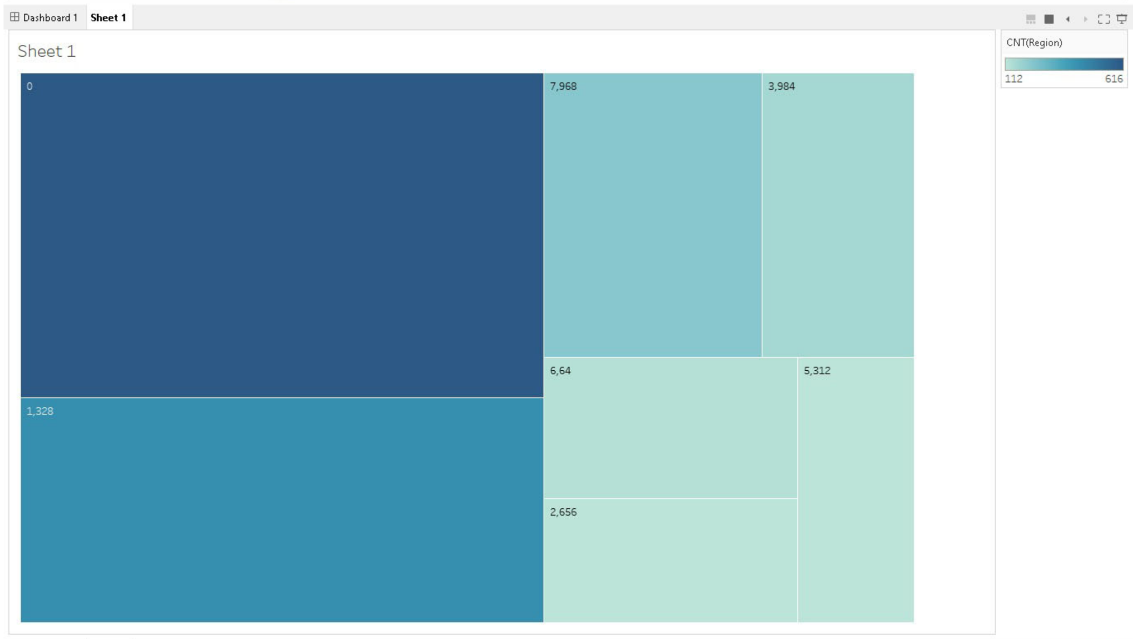

Sheet 1
Sheet 1
Created On
County
Package
Size
100K
50K
0K
40K
20K
0K
200K
100K
0K
1970
2014
2016
2017
2018
2019
2020

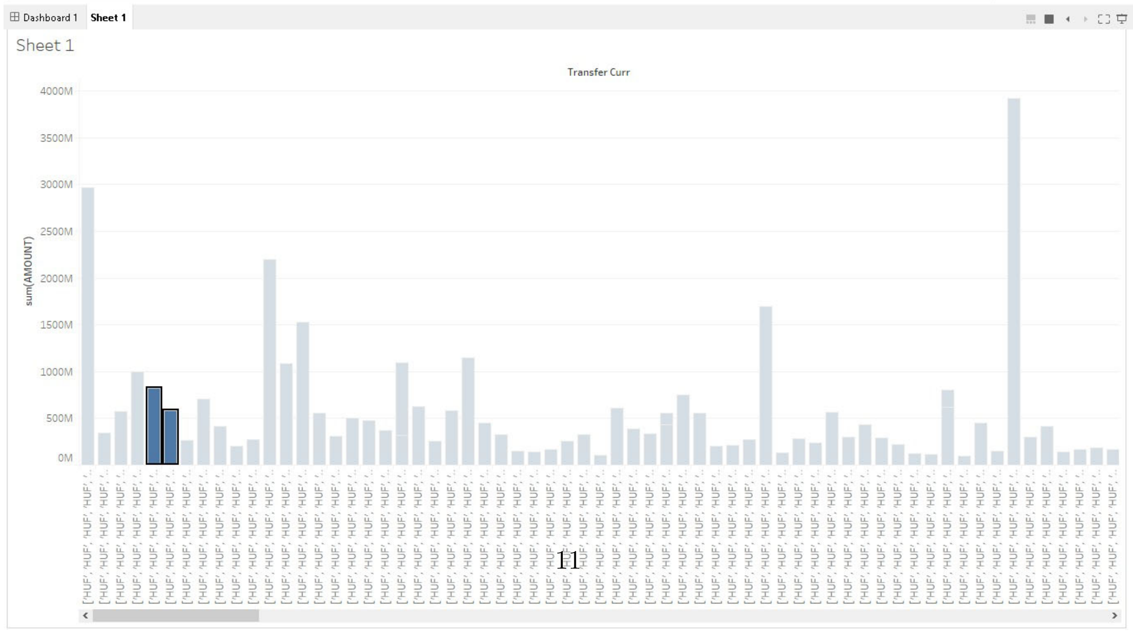

Sheet 1
Year of Created On
County
240K
220K
200K
180K
160K
140K
120K
100K
80K
60K
40K
20K
0K
1970
2014
2016
2017
2018
2019
2020

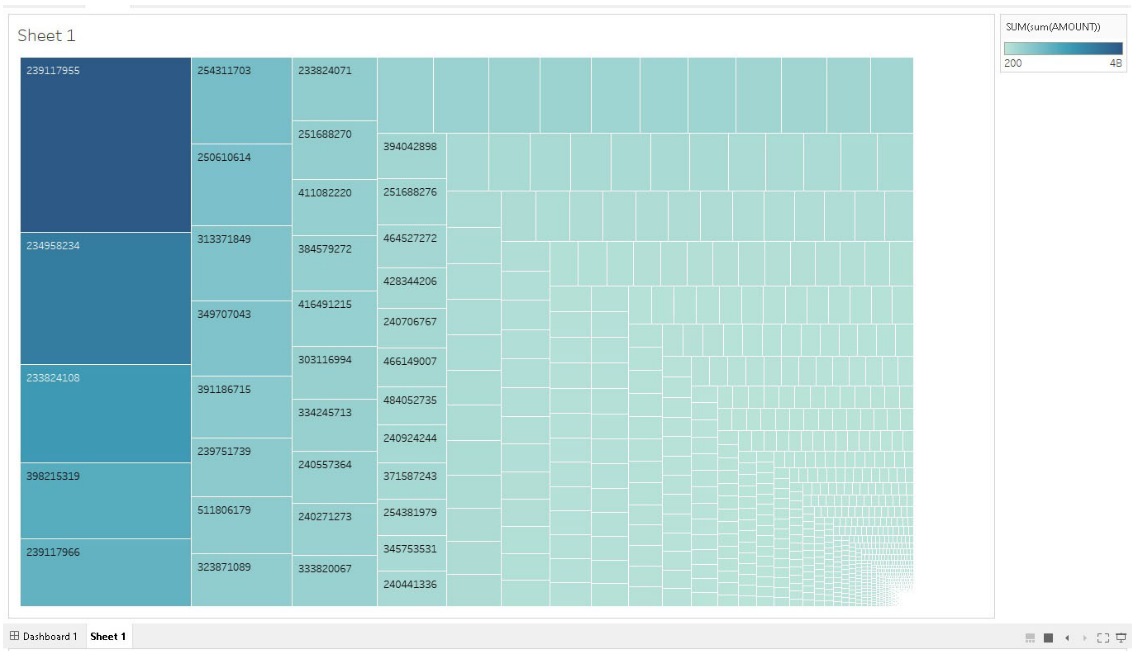

Sheet 1
Sheet 1
max(TAXEXCLUSIVEAMOUNT)
1400M
1200M
1000M
800M
600M
400M
200M
0M
sum(TAXINCLUSIVEAMOUNT)
7G
6G
5G
4G
3G
2G
1G
0G

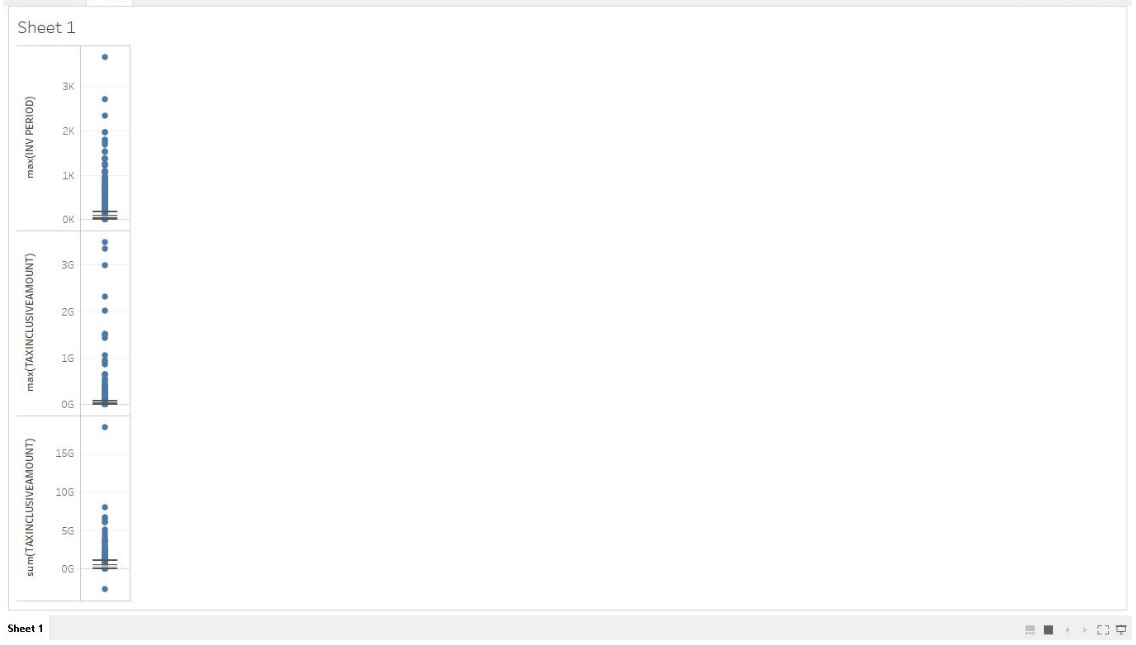

Dashboard 1
Sheet 1
Sheet 1
Count of Region
600
550
500
450
400
350
300
250
200
150
100
50
0
0,0
0,5
1,0
1,5
2,0
2,5
3,0
3,5
4,0
4,5
5,0
5,5
6,0
6,5
7,0
7,5
8,0
8,5
9,0
9,5
Region (bin)


Sheet 1
398215319
239117955
234958234
239117966
233824108
254311703

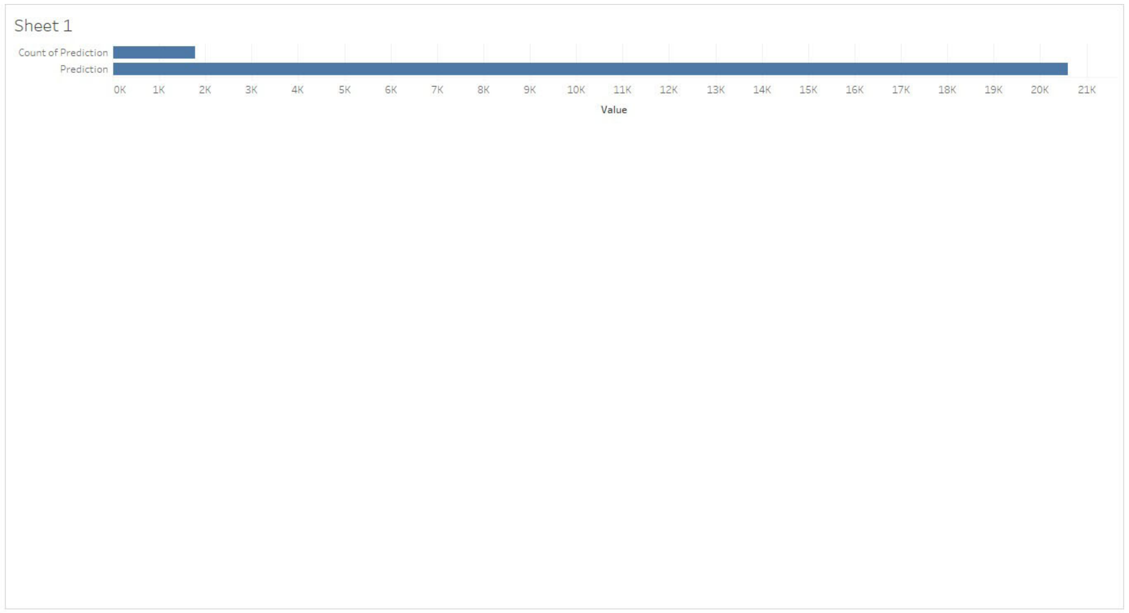

Dashboard 1
Sheet 1
Sheet 1
0
7,968
3,984
1,328
6,64
5,312
2,656
CNT(Region)
112
616

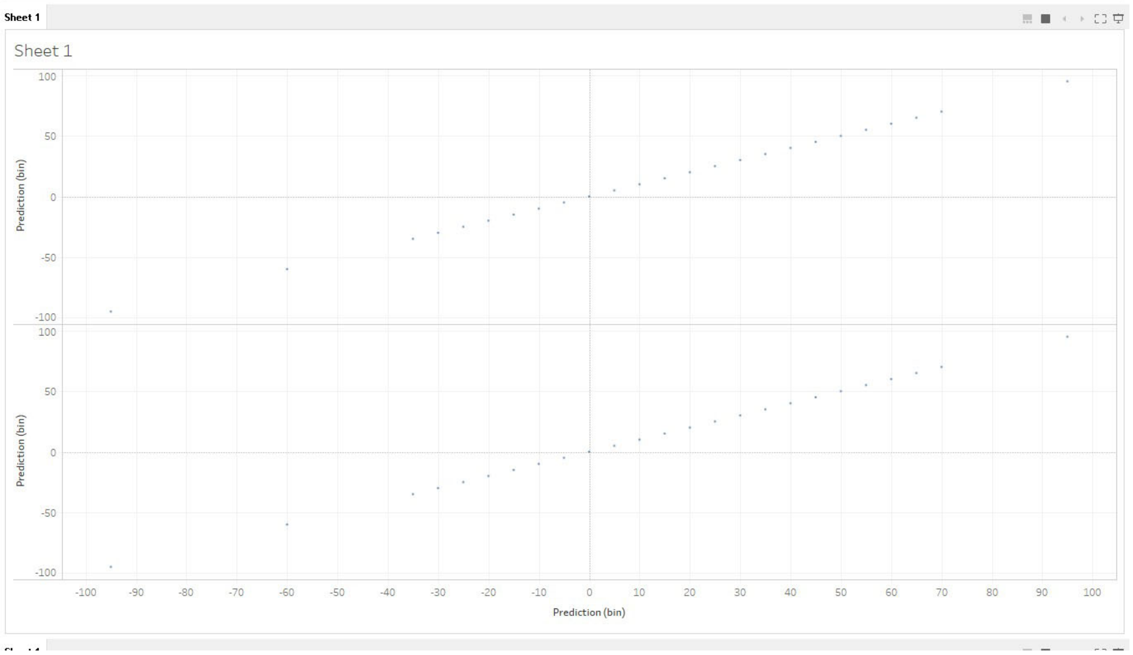

Dashboard 1
Sheet 1
Sheet 1
Transfer Curr
sum(AMOUNT)
4000M
3500M
3000M
2500M
2000M
1500M
1000M
500M
0M
['HUF', 'HUF', ...

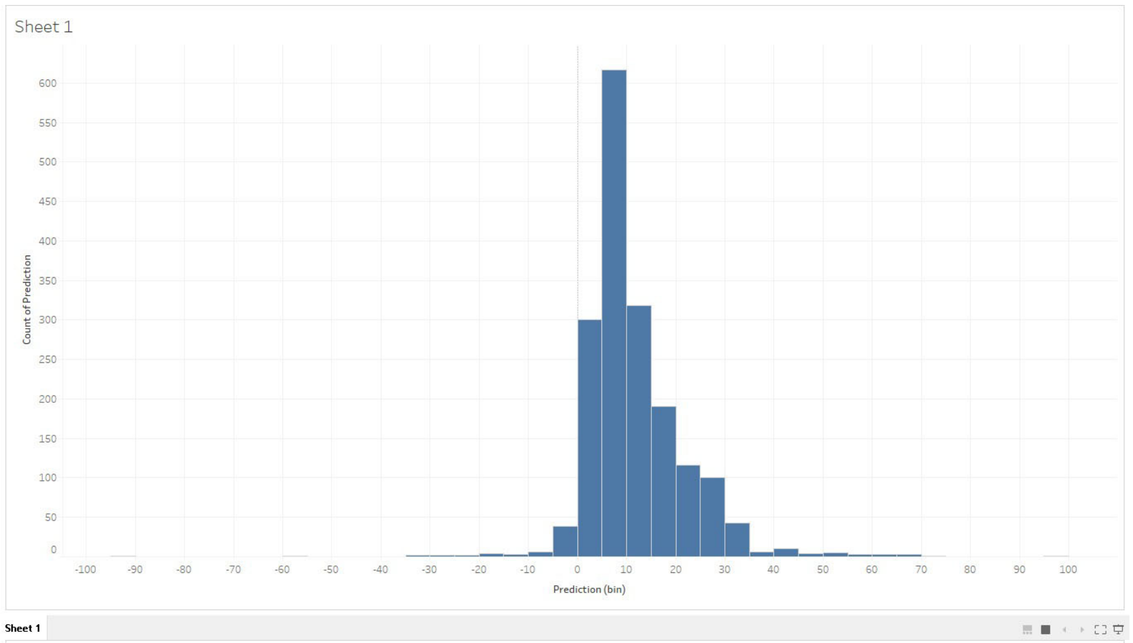

Sheet 1
SUM(sum(AMOUNT))
200
4B
239117955
234958234
233824108
398215319
239117966
254311703
250610614
313371849
349707043
391186715
239751739
511806179
323871089
233824071
251688270
411082220
384579272
416491215
303116994
334245713
240557364
240271273
333820067
394042898
251688276
464527272
428344206
240706767
466149007
484052735
240924244
371587243
254381979
345753531
240441336
Dashboard 1
Sheet 1

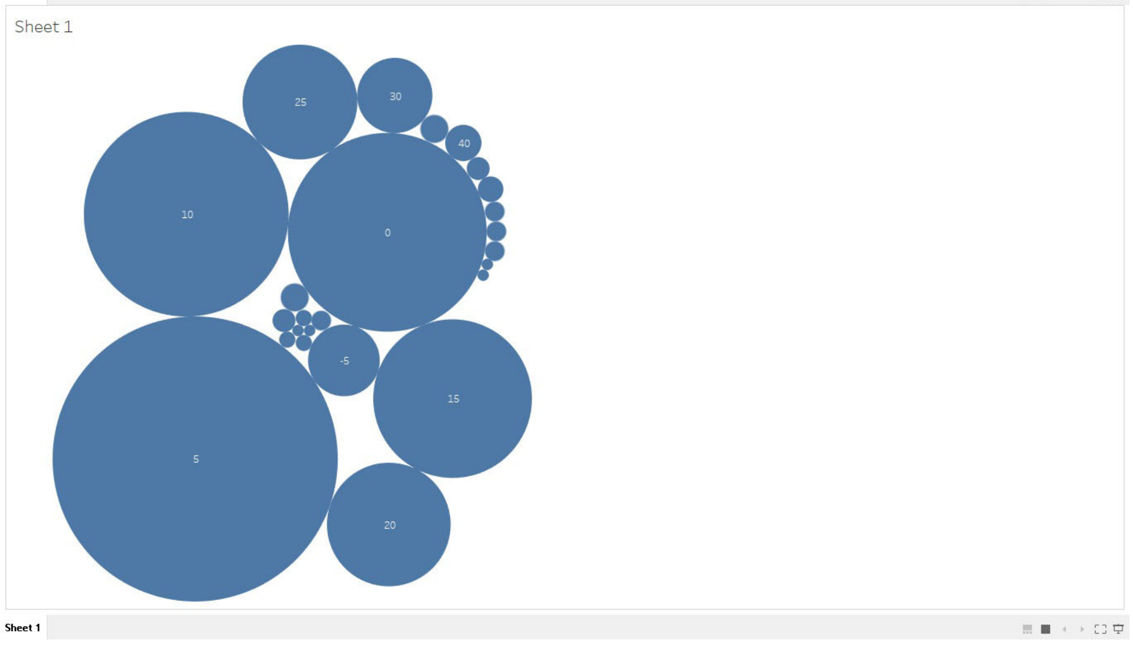

Sheet 1
max(INV PERIOD)
3K
2K
1K
0K
max(TAXINCLUSIVEAMOUNT)
3G
2G
1G
0G
sum(TAXINCLUSIVEAMOUNT)
15G
10G
5G
0G
Sheet 1

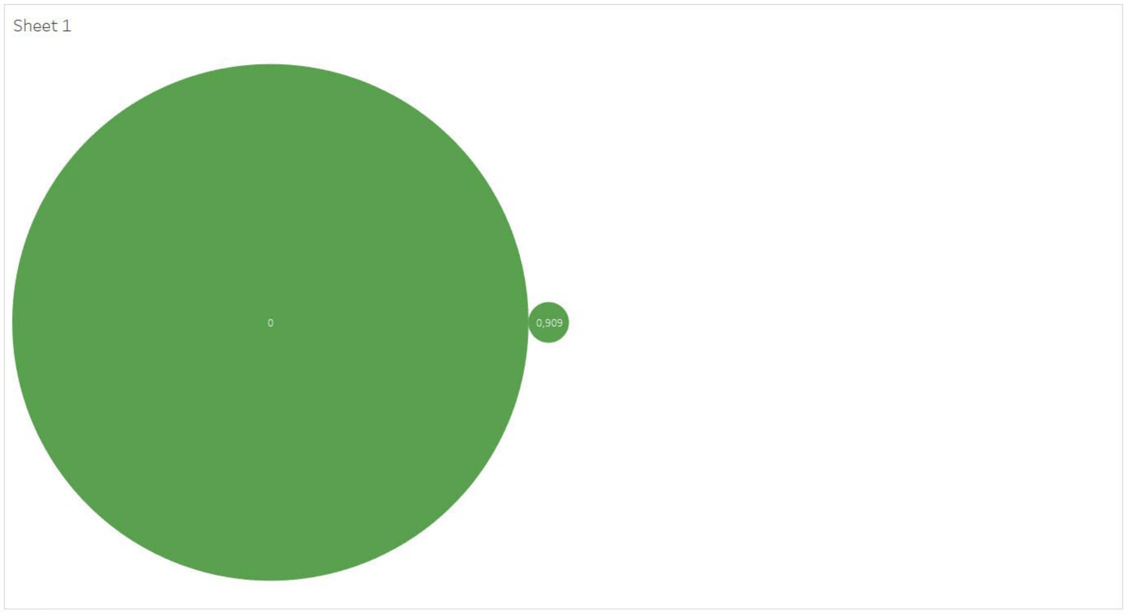
9.2 visualizing Model Results

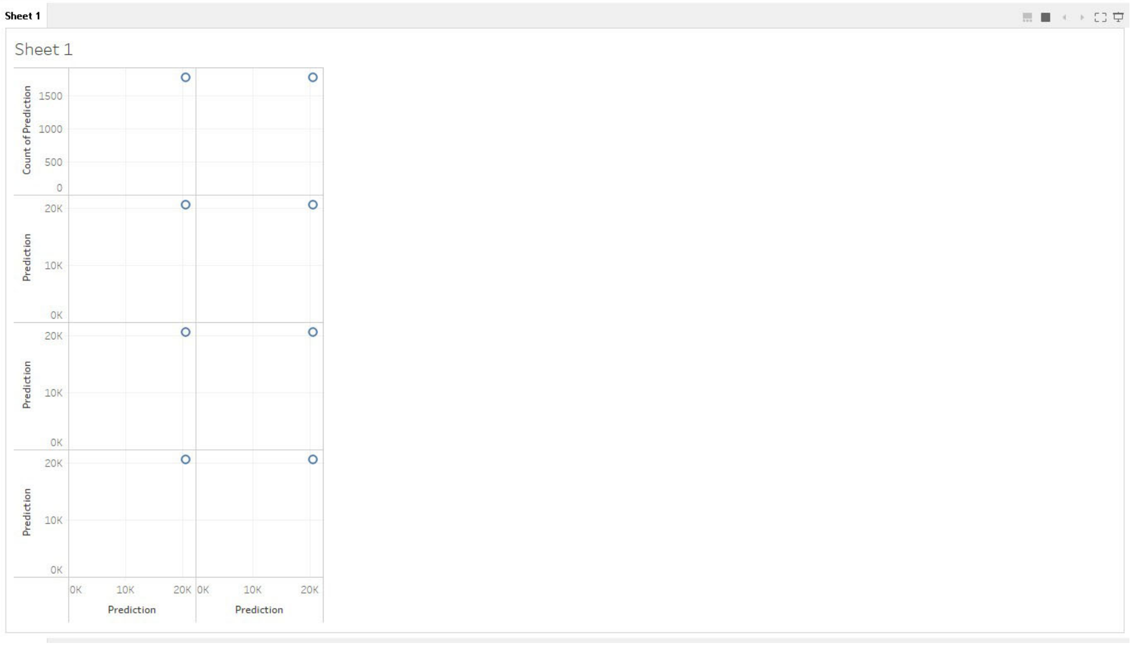


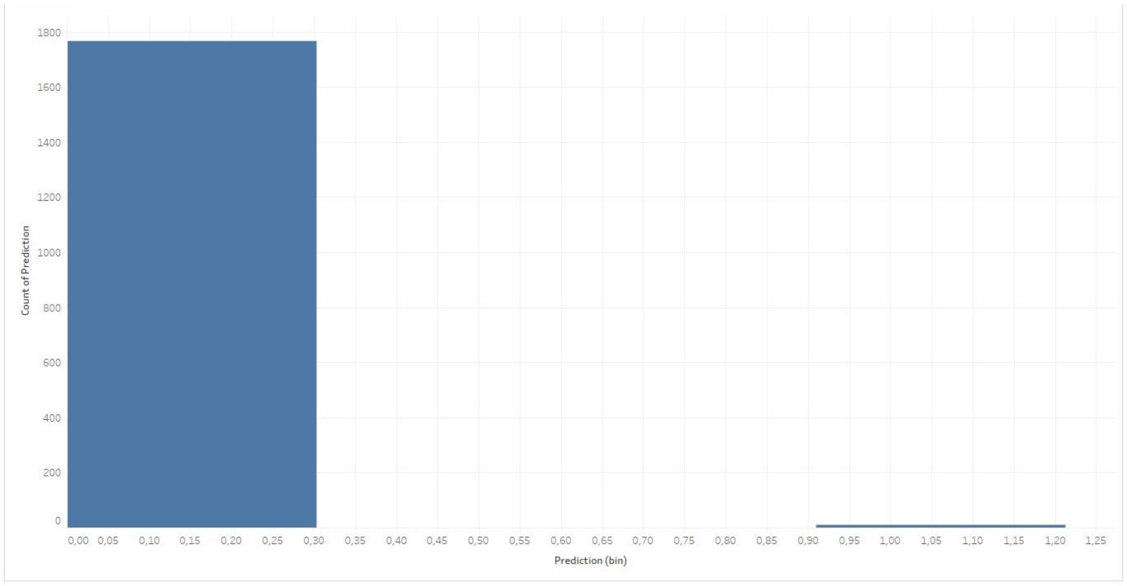



Sheet 1
Count of Prediction
600
550
500
450
400
350
300
250
200
150
100
50
0
-100
-90
-80
-70
-60
-50
-40
-30
-20
-10
0
10
20
30
40
50
60
70
80
90
100
Prediction (bin)
Sheet 1

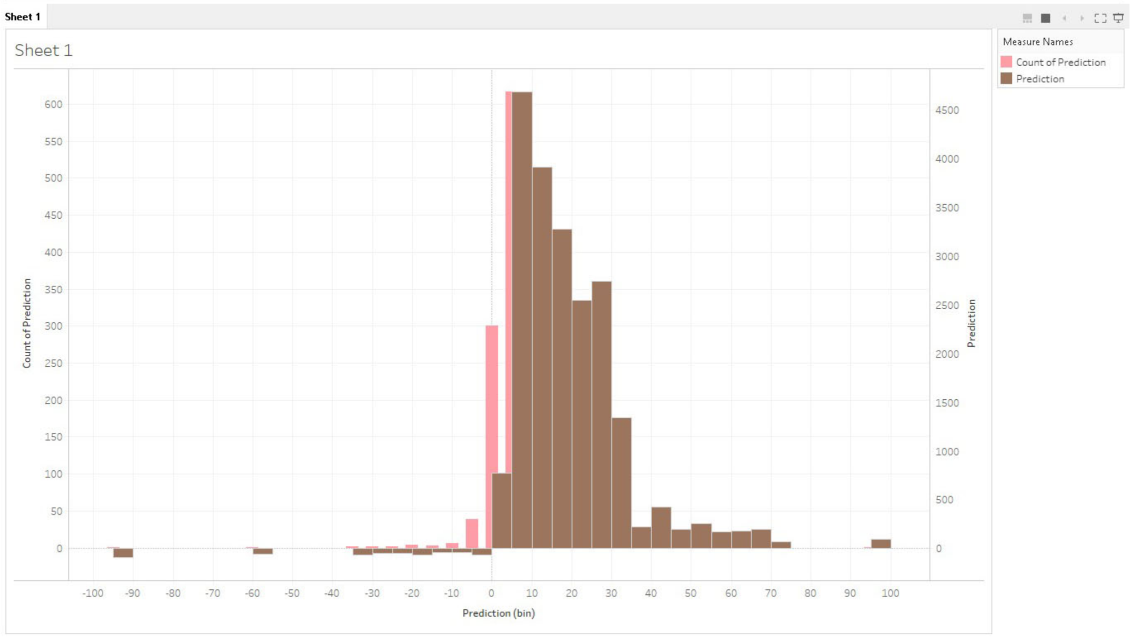

Sheet 1
25
30
40
10
0
-5
15
5
20
Sheet 1


Sheet 1
0
0,909


Sheet 1
Sheet 1
Count of Prediction
1500
1000
500
0
Prediction
20K
10K
0K
Prediction
20K
10K
0K
Prediction
20K
10K
0K
0K
10K
20K
Prediction
0K
10K
20K
Prediction



Count of Prediction
1800
1600
1400
1200
1000
800
600
400
200
0
0,00 0,05 0,10 0,15 0,20 0,25 0,30 0,35 0,40 0,45 0,50 0,55 0,60 0,65 0,70 0,75 0,80 0,85 0,90 0,95 1,00 1,05 1,10 1,15 1,20 1,25
Prediction (bin)



Sheet 1
Sheet 1
Measure Names
Count of Prediction
Prediction
Count of Prediction
600
550
500
450
400
350
300
250
200
150
100
50
0
Prediction
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-100 -90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 50 60 70 80 90 100
Prediction (bin)

10 Contributions

Sarmad Shaikh:

- Contributed to the design of the architecture of the system
- Contributed to the streaming pipeline and the model training pipeline of the project
- Contributed to the Kafka producer and the consumer part in the streaming part
- Connected PySpark pipelines to Cassandra tables for storage
- Generated CQL script for replication of Cassandra database
- Contribution to the setup of Tableau with Cassandra
- Contribution to creation of the report and presentation with the other members of the team.
- Participated in all team meetings

Salma Moustafa:

- Contributed to setting up Apache Kafka and working on Kafka producer.
- Setting up Pyspark on Docker.
- Worked on invoices and transfers in Pyspark pipeline for preprocessing and training.
- Contributed to designing System Architecture.
- Worked on setting up Cassandra and connecting Cassandra to Pyspark as part of the pipeline.
- Contribution to configuration and setting up Tableau.
- Collaborated on writing report and making presentation with the rest of the team members.
- Participated in all team meetings.

Tameem Alshomari

- Data preprocessing on Jupyter Notebook: For static data and customer login history
- Setting up Pyspark on Docker.
- Worked on Static Data and customer login history in Pyspark pipeline for preprocessing and training.
- Contributed to designing System Architecture.
- Worked on setting up Cassandra and connecting Cassandra to Pyspark as part of the pipeline.
- Contributed to setting up Apache Kafka and Contributed to the Kafka producer and the consumer part in the streaming part
- Contribution to configuration and setting up Tableau.
- Contribution to the visulization part on Tableau for the visualizing of tables data and model results
- Collaborated on writing report and making presentation with the rest of the team members.
- Participated in all team meetings.
- Code repository management on Github: creating repository, reviewing pull requests and access management
- Participated in preparing presentation material

11 Github Repository

https://github.com/TameemElshoumari/explainable-customer-profiling-in-financial-systems

12 References

1. F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer,and G. Bontempi, "Scarff: a scalable framework for streaming credit card fraud detection with Spark,"Information fusion, vol. 41, pp. 182–194, 2018.
2. Apache Spark Structured Streaming Programming Guide, https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
3. Apache Spark, https://spark.apache.org/
4. Spark Cassandra Connector, https://github.com/datastax/spark-cassandra-connector
5. PySpark Kafka Integration Guide, https://spark.apache.org/docs/latest/structured-streaming-kafka-integr html#deploying
6. Apache Kafka, https://kafka.apache.org/intro