1.1 Dataset –Definition of ML Task

Predicting whether a newly IPO'd company will be delisted within the next five quarters.

Initial Public Offering (IPO): Process in which a privately owned company lists its shares on a stock exchange to raise capitals from investors

Binary Classification

1.2 Dataset – Use of Different File

File Name	Source	Number of Rows	Number of Columns	File Size (MB)	Main Entities	Usage
DS1-IPODataFull.csv	Kaggle	3673	1669	28.8MB	Name, LastSale, Market Cap, Sector, Industry, Summary Quote	Not Used
DS2-CompaniesRanked.csv	Companies MarketCap	8091	6	474KB	Rank, Name, Symbol, Employees Count, Price (USD), Country	Not Used
DS3-company_ipo.csv	Kaggle	1766	7	109KB	ID, IPO Date, Symbol, Company Name, IPO Price, Current Return	Not Used
DS4-final_merged_ipos.csv	StockAnalysis	2307	19	185KB	Company Name, Country, Current Deal Size, Employees, Exchange, Founded, IPO Date, IPO Price, Industry, Is SPAC, Market Cap, Open Price, Return, Return(Open), Sector, Shares Offered, Symbol, Volume	Used
DS5-delisted_companies.csv	AlphaVantage	7854	7	637KB	Symbol Name, Exchange, Asset Type, IPO Date, Delisting Date, Status	Used

1.2 Dataset – Data sources

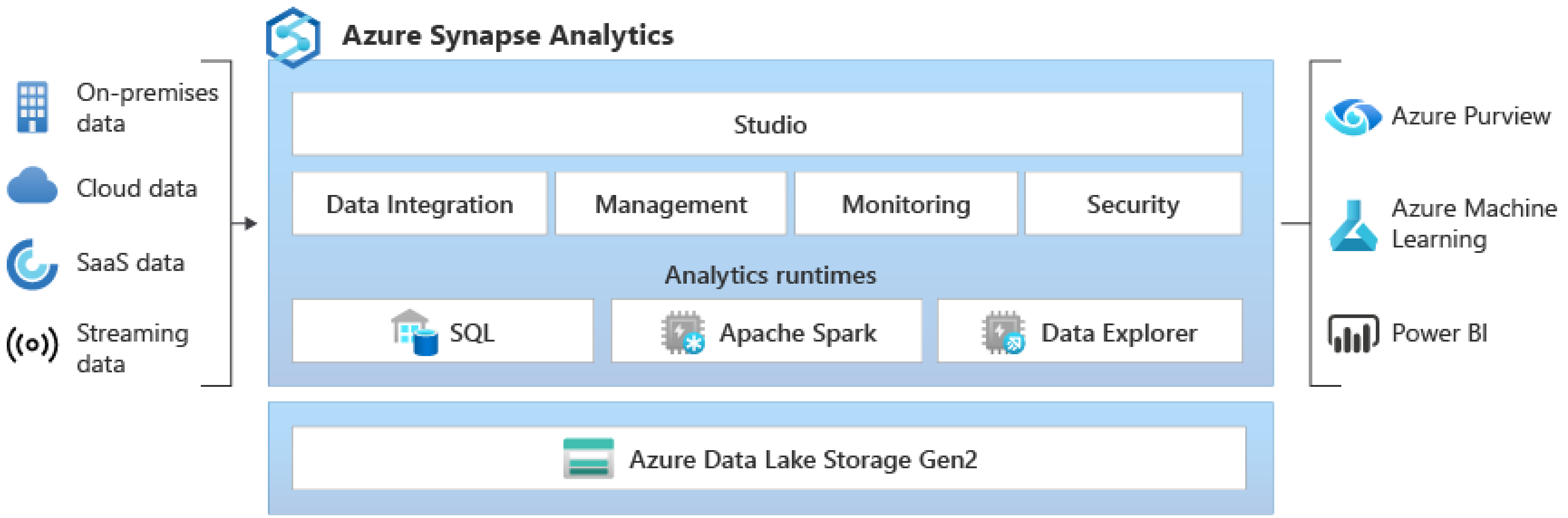
	Data source for listed companies	Data source for delisted companies
Company names & stock symbols	STOCK ANALYSIS	ALPHA VANTAGE
Financial Data	ALPHA VANTAGE	barchart

1.3 Dataset – Use of Storage Systems

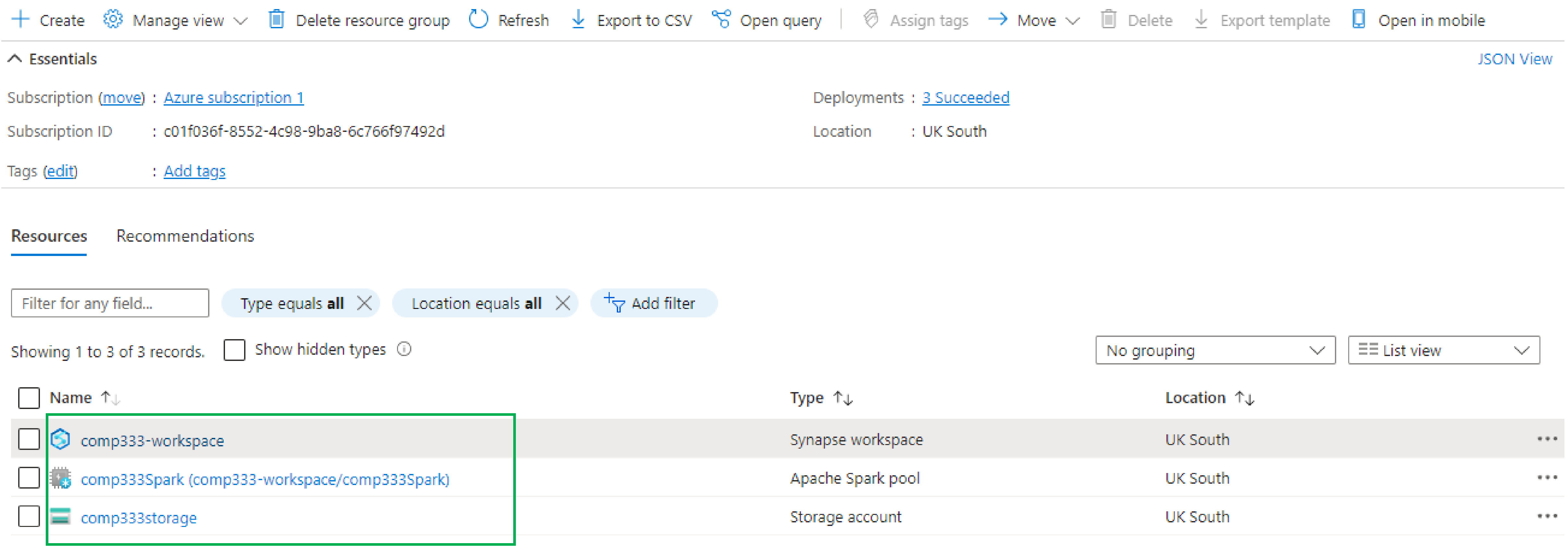
A mix of everything

Google colab

Azure Synapse Analytics



1.3 Dataset – Use of Storage Systems



1.3 Dataset – Example with Data Transformation

Data

Workspace Linked

Filter resources by name

Azure Data Lake Storage Gen2 3

DataTransformation

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

Run all | Undo | Publish | Outline | Attach to comp333Spark | Language PySpark (Python) | Variables

Session timed out. Run the notebook to start a new session.

+ Code + Markdown

```python
import pandas as pd
import csv
import re
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import normalize,MinMaxScaler,StandardScaler, RobustScaler
import sklearn
```

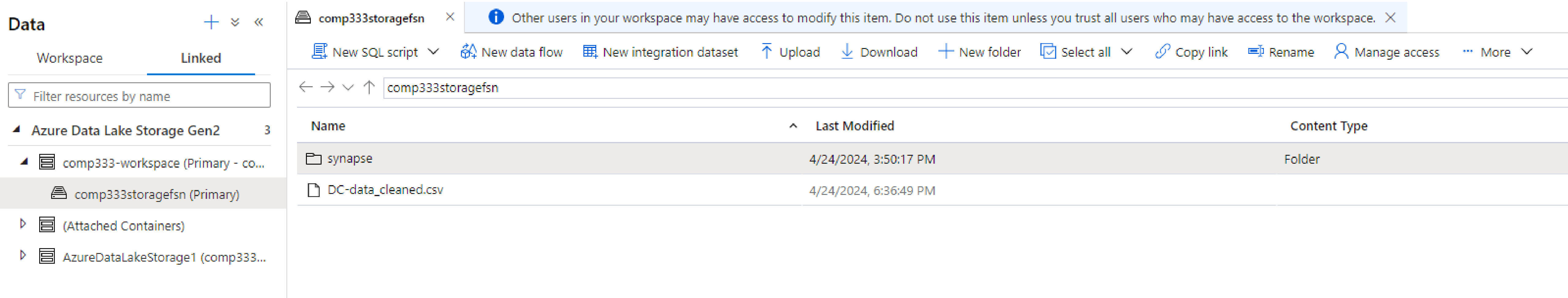
```python
df3_cleaned= pd.read_csv('abfss://comp333storagefsn@comp333storage.dfs.core.windows.net/DC-data_cleaned.csv')
df3_cleaned
```

Data Encoding

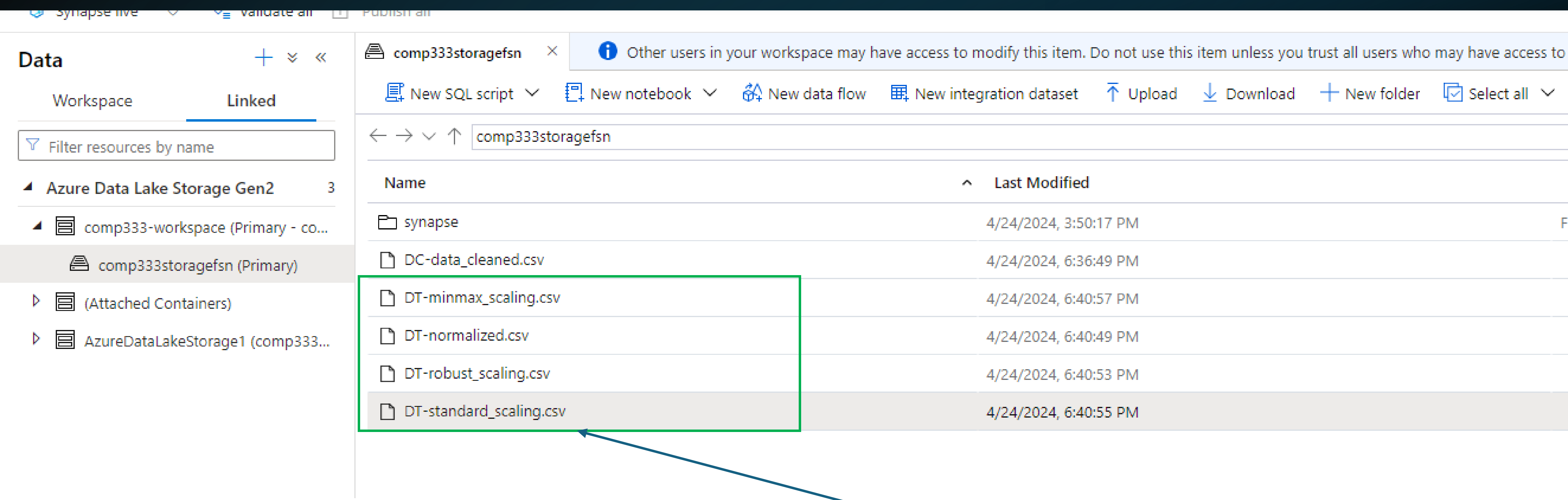
```python
encoder = LabelEncoder()

df3_cleaned['status'] = encoder.fit_transform(df3_cleaned['status'])
df3_cleaned
```

Data Normalization

1.3 Dataset – Data Lake Storage (Before)



1.3 Dataset – Data Lake Storage (After)



Outputted files from our DataTransformation code file can be seen directly in the data lake

1.3 Dataset – Quick Preview

DT-normalized.csv

Path https://comp333storage.dfs.core.windows.net/comp333storagefsn/DT-normalized.csv
Modified 4/24/2024, 6:40:49 PM

With column header On

STATUS	SALESQ5	SALESQ4	SALESQ3
0	0.004744900661...	0.004260704379...	0.00531580
0	0.454580712883...	0.408192746759...	0.50927520
0	0.454581864047...	0.408193780452...	0.50927649
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580
0	0.058761507889...	0.072576909431...	0.08715983
0	0.004744900661...	0.004260704379...	0.00531580
0	0.004744900661...	0.004260704379...	0.00531580

OK

DT-standard_scaling.csv

Path https://comp333storage.dfs.core.windows.net/comp333storagefsn/DT-standard_scaling.csv
Modified 4/24/2024, 6:40:55 PM

With column header On

STATUS	SALESQ5	SALESQ4	SALESQ3
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	1.660584917713...	1.672020090394...	1.72881015
0	1.660590972634...	1.672026203062...	1.72881636
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.42135425537...	-0.31261951678...	-0.30397716
0	-0.70547207776...	-0.71660237837...	-0.6981147
0	-0.70547207776...	-0.71660237837...	-0.6981147

OK

DT-minmax_scaling.csv

Path https://comp333storage.dfs.core.windows.net/comp333storagefsn/DT-minmax_scaling.csv
Modified 4/24/2024, 6:40:57 PM

With column header On

STATUS	SALESQ5	SALESQ4	SALESQ3
0	0.293188593162...	0.053360077968...	0.00532465
0	0.612653510540...	0.440620108310...	0.50929682
0	0.612654328075...	0.440621099338...	0.50929811
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465
0	0.331550169756...	0.118856579522...	0.08717076
0	0.293188593162...	0.053360077968...	0.00532465
0	0.293188593162...	0.053360077968...	0.00532465

OK

DT-robust_scaling.csv

Path https://comp333storage.dfs.core.windows.net/comp333storagefsn/DT-robust_scaling.csv
Modified 4/24/2024, 6:40:53 PM

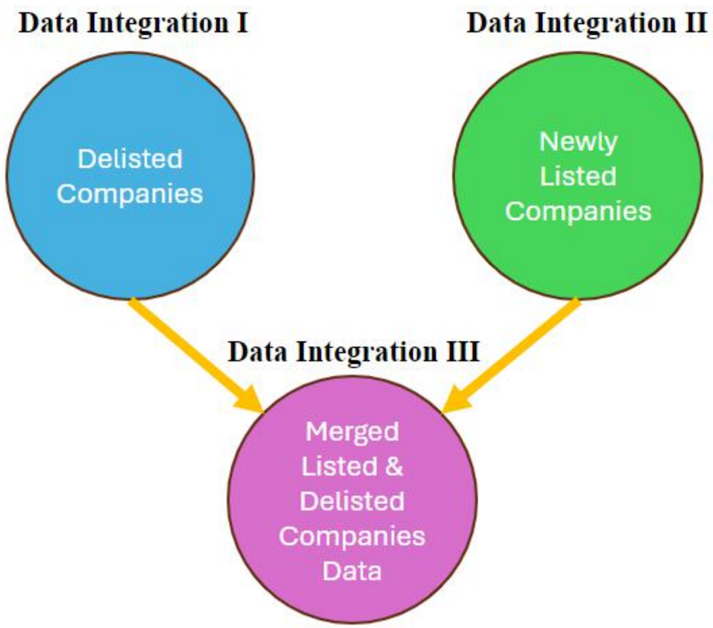
With column header On

STATUS	SALESQ5	SALESQ4	SALESQ3
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	1.421105690206...	1.426997887601...	1.52292285
0	1.421109504554...	1.427001737125...	1.52292689
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	0.109570695939...	0.177147743223...	0.19980502
0	-0.06941170248...	-0.07726521722...	-0.0567345
0	-0.06941170248...	-0.07726521722...	-0.0567345

OK

2. 1 Schema Integration- Overview

- Three main groups of Integration:



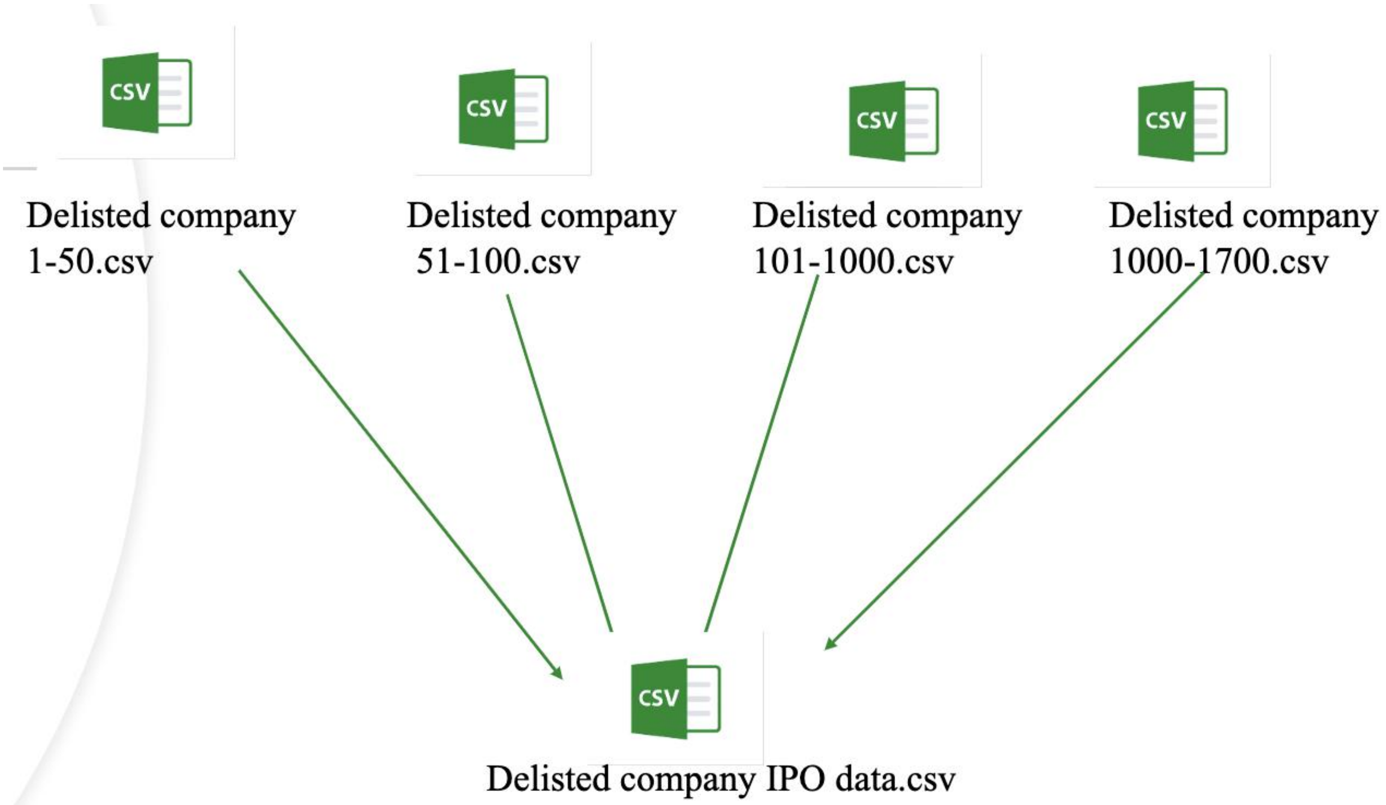
Schema Integration 1 and Mapping: Delisted IPO Data

- Key: **'symbol'**

- **Initial Aggregation**: Add monthly financial data from the first two files—'salesJan', 'salesFeb', 'salesMar' to calculate 'salesQ1'.

- **Subsequent Mapping**: Correspond 'salesQ1' from initial files with 'salesQ1' in subsequent datasets, where quarterly data already exists.

- **Uniform Naming Convention**: Consolidate figures under a standardized label 'XXQN' across all files for clarity and consistency in financial analysis.



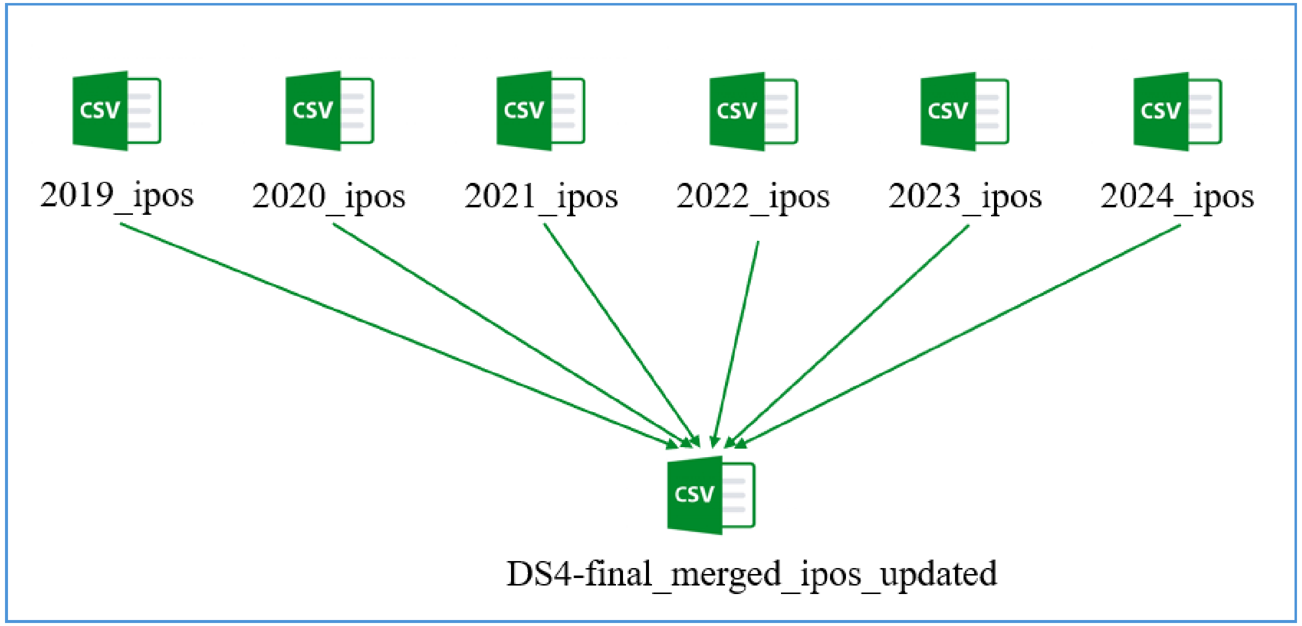
Schema Integration and Mapping 2: Listed IPO Data

-Key: **'Symbol'**

-**Date Harmonization**: Align **'IPO Date'** from 2019 and 2020 files with **'ipo date'** in subsequent datasets.

-**Renaming for Consistency**: Standardize field names by renaming all instances to **'IPO Date'.**

-**Format Unification**: Ensure date structure adheres to a consistent 'Year-Month-Date' format across all records.



Schema Integration 3 and Mapping :

Listed and Delisted IPO Data

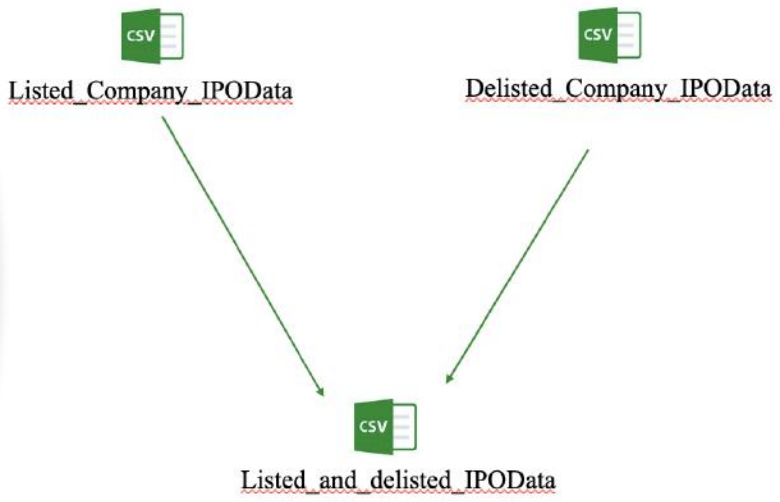
Key: **'symbol'**

Map **'Symbol'** in listed company file to **'symbol'** in delisted company, rename it as **'symbol'**

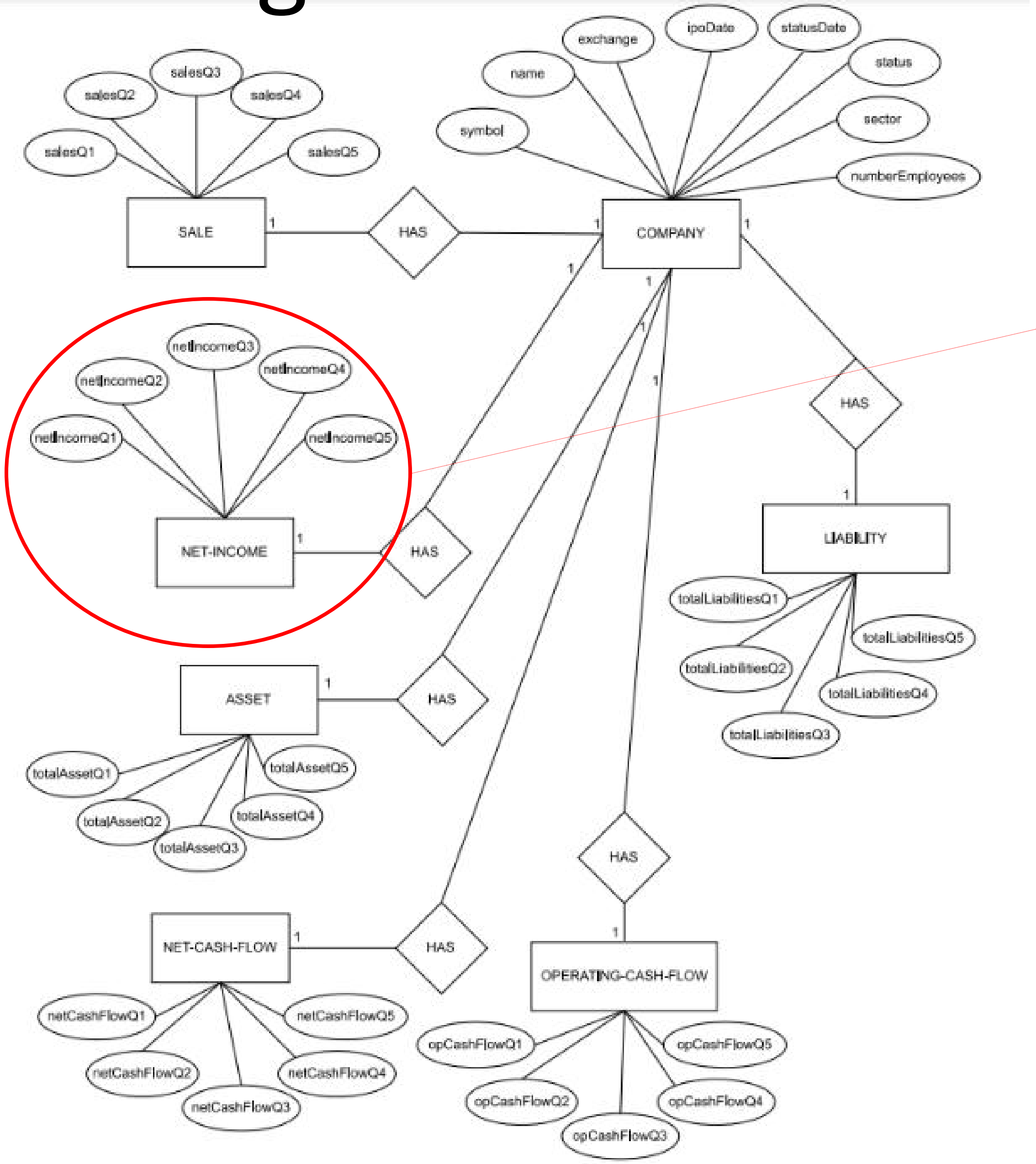
Map **'IPO date'** in listed company file to **'IPO delisted Date'** in delisted company file

-> rename it as **'IPO date'**

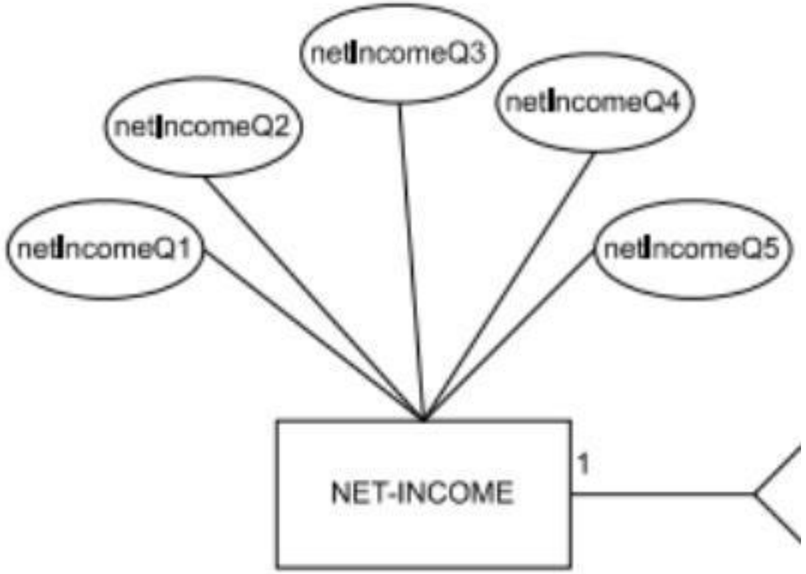
-> split it as **'IPOMonth'**. **'IPOYear'**, **'IPOIsWeekend'**



ER Diagram

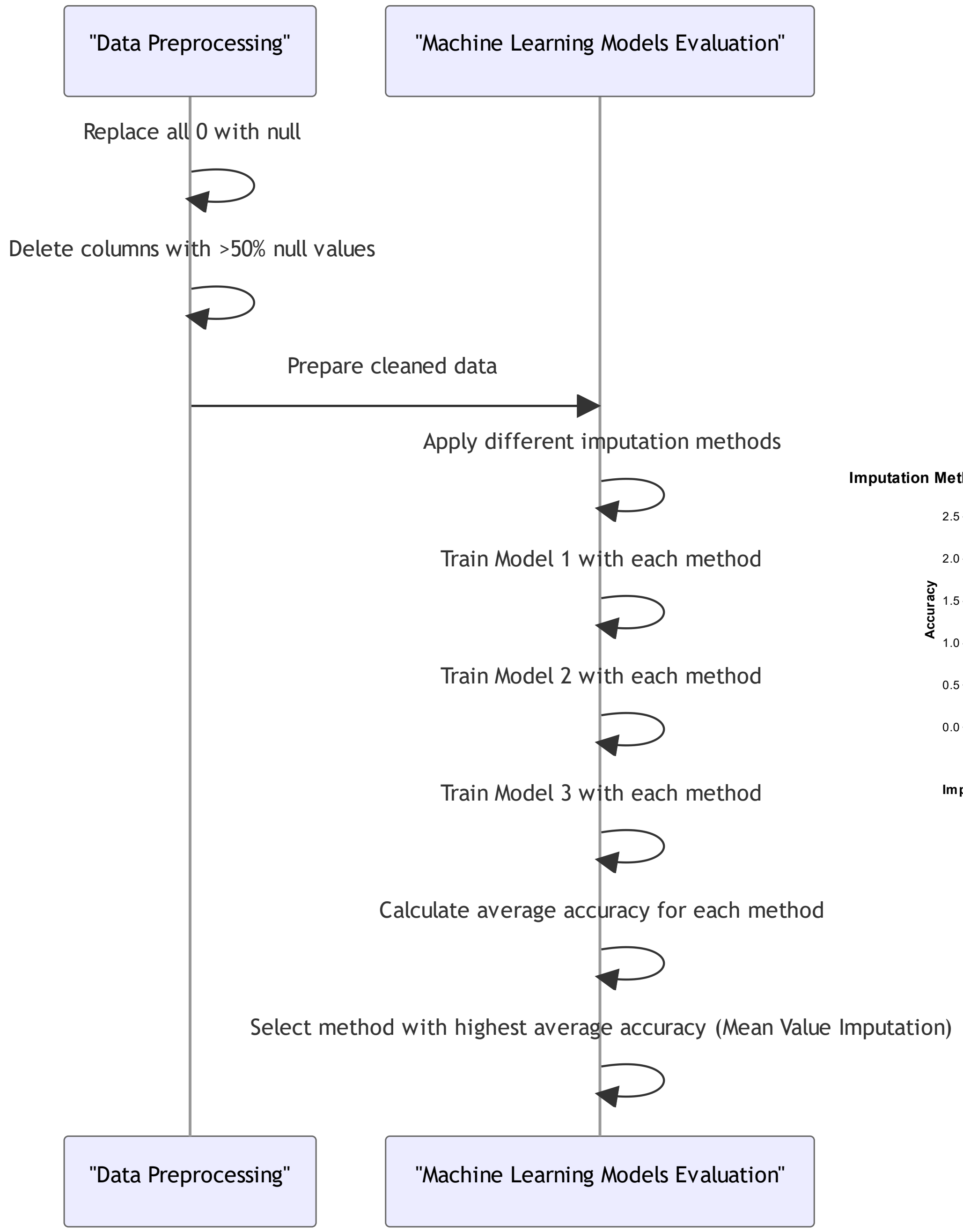


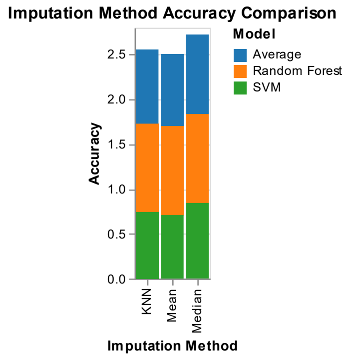
salesQ1
salesQ2
salesQ3
salesQ4
salesQ5
SALE
HAS
COMPANY
symbol
name
exchange
ipoDate
statusDate
status
sector
numberEmployees
netIncomeQ1
netIncomeQ2
netIncomeQ3
netIncomeQ4
netIncomeQ5
NET-INCOME
HAS
HAS
LIABILITY
totalLiabilitiesQ1
totalLiabilitiesQ2
totalLiabilitiesQ3
totalLiabilitiesQ4
totalLiabilitiesQ5
ASSET
HAS
totalAssetQ1
totalAssetQ2
totalAssetQ3
totalAssetQ4
totalAssetQ5
HAS
OPERATING-CASH-FLOW
opCashFlowQ1
opCashFlowQ2
opCashFlowQ3
opCashFlowQ4
opCashFlowQ5
NET-CASH-FLOW
HAS
netCashFlowQ1
netCashFlowQ2
netCashFlowQ3
netCashFlowQ4
netCashFlowQ5



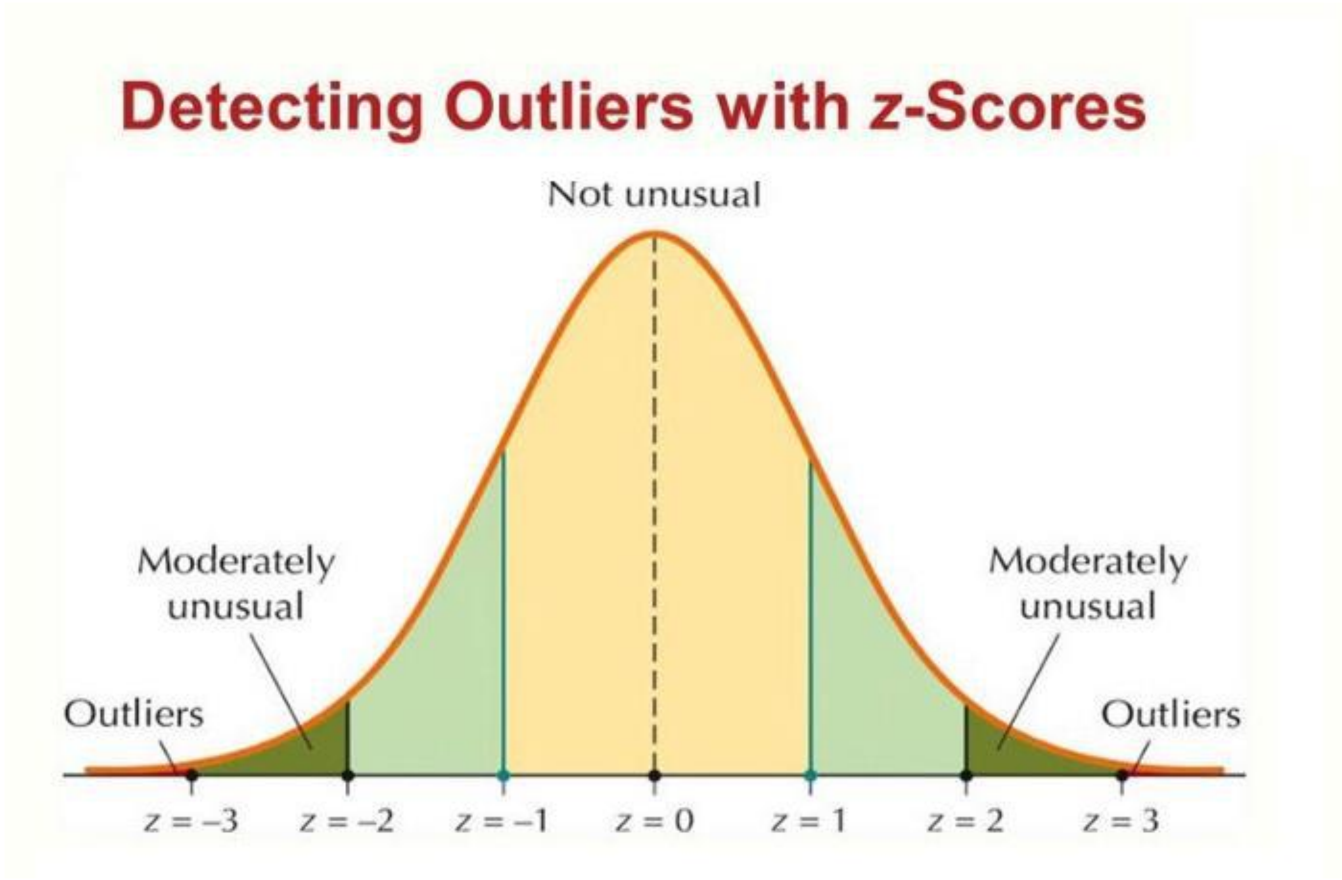
netIncomeQ1
netIncomeQ2
netIncomeQ3
netIncomeQ4
netIncomeQ5
NET-INCOME
1

3. Data Cleaning: Support DC on Null values





3. Data Cleaning: Support DC for outliers



- After visualize our data, we notice it follows a **Gaussian normal distribution.**

-data points with a Z-Score greater than a **threshold=3** are considered outliers

Model	Description	Accuracy Score
SVM	Before Data Cleaning	0.5969
SVM	After Data Cleaning	0.7286

3. Data Cleaning: ML model Improvement

4. Data Transformation

Data Processing – Normalization

- API: sklearn.preprocessing.**Normalizer**
- This API preserves unit norm, since our ML model is SVM, it relies on the distance between support vectors and boundary that maximizes the margin

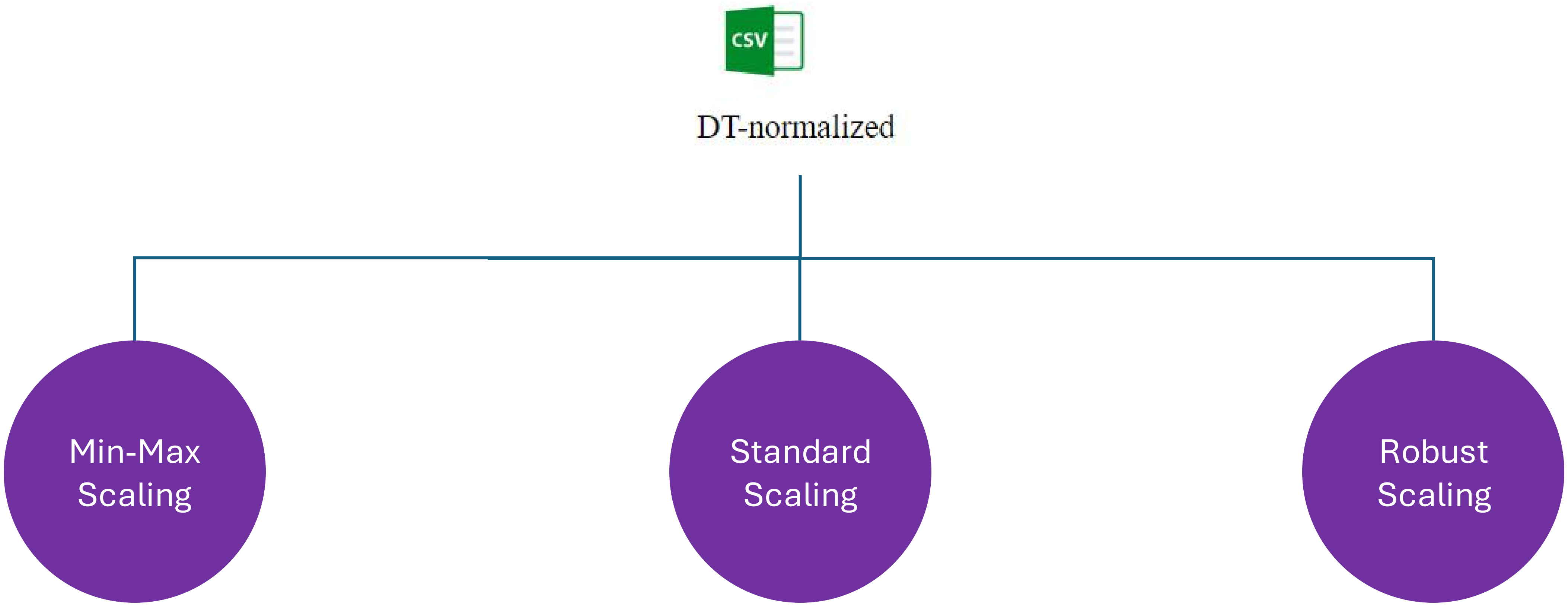
```
df_scaled.head()
```

	salesQ5	salesQ4	salesQ3	salesQ2	salesQ1	netIncomeQ5	netIncomeQ4	netIncomeQ3	netIncomeQ2	netIncomeQ1	...
0	0.004745	0.004261	0.005316	0.004351	0.004593	-0.000097	-8.839976e-05	-5.628035e-05	-1.165012e-04	-1.133295e-03	...
1	0.454581	0.408193	0.509275	0.416830	0.440023	0.000005	9.235170e-06	1.633772e-05	3.597637e-06	1.088563e-05	...
2	0.454582	0.408194	0.509276	0.416832	0.440024	0.000001	-1.854457e-07	-5.563370e-07	-2.781685e-07	-4.079804e-07	...
3	0.004745	0.004261	0.005316	0.004351	0.004593	-0.000097	-8.839976e-05	-5.628035e-05	-1.165012e-04	-1.133295e-03	...
4	0.004745	0.004261	0.005316	0.004351	0.004593	-0.000097	-8.839976e-05	-5.628035e-05	-1.165012e-04	-1.133295e-03	...

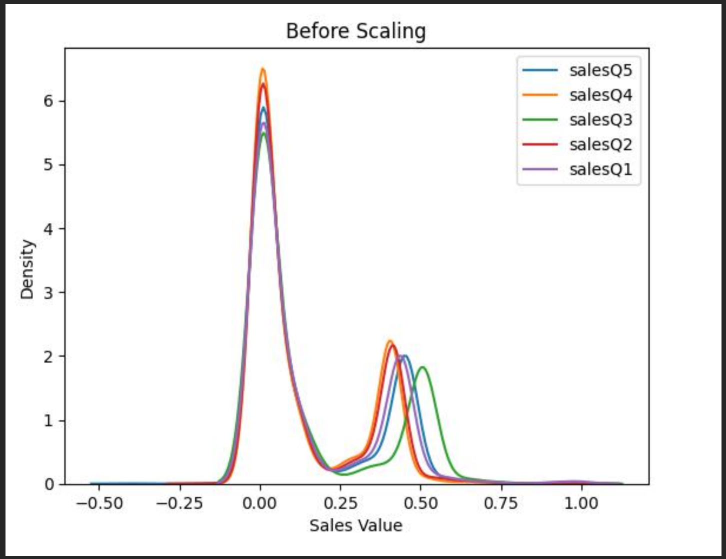
rows × 31 columns

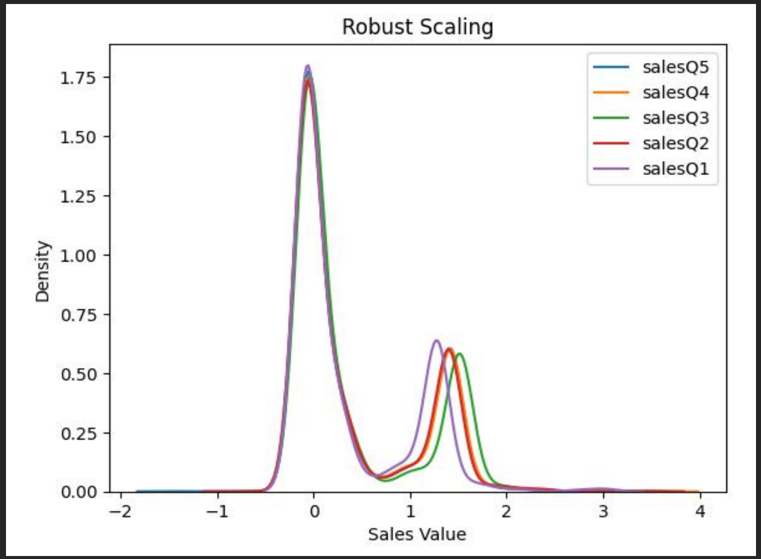
4. Data Transformation – Applying different scaling functions after normalization

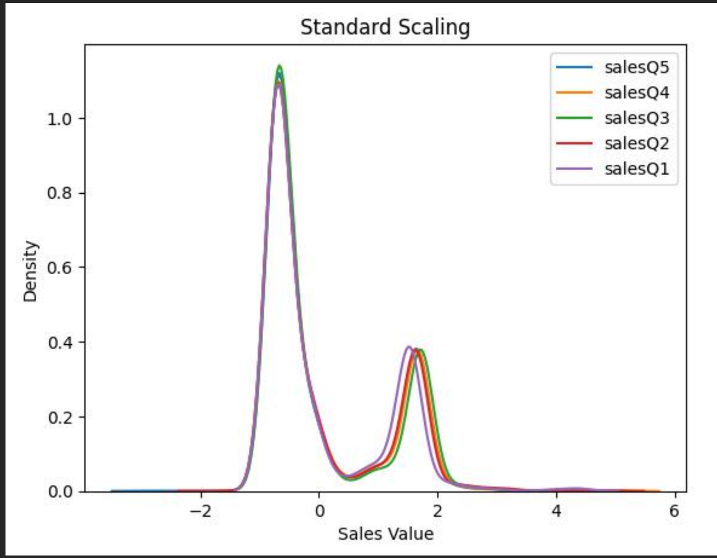
- We considered different scaling functions in our scaling process

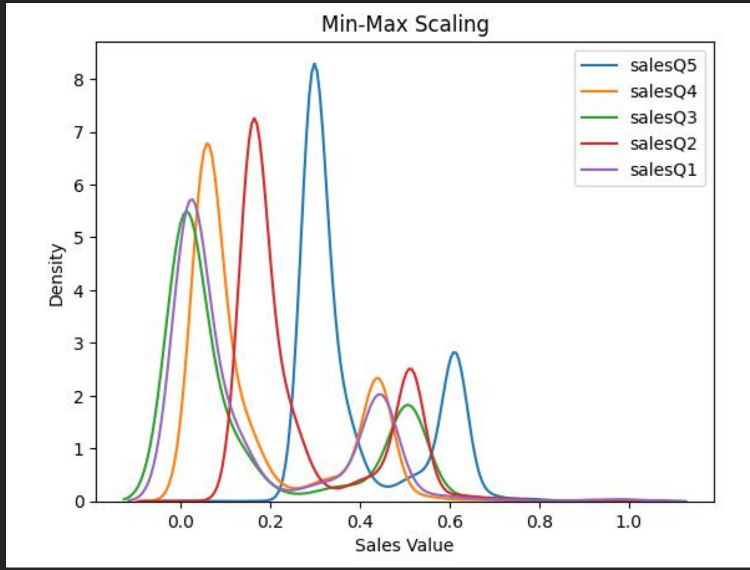


Sales

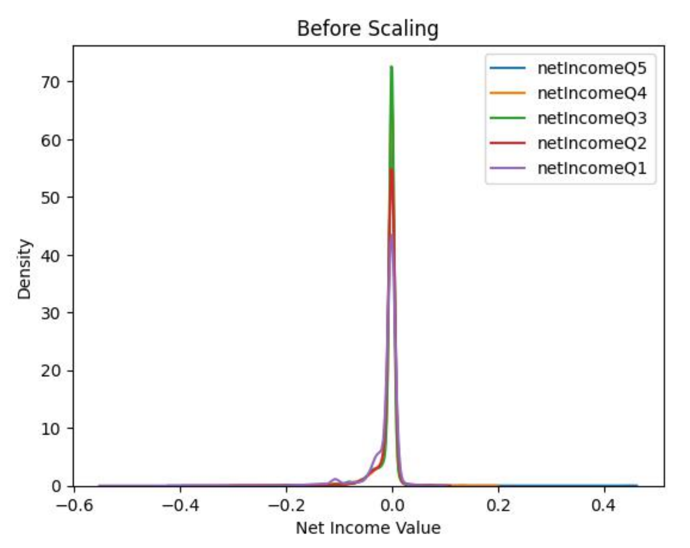

Before Scaling
salesQ5
salesQ4
salesQ3
salesQ2
salesQ1
Density
Sales Value
−0.50 −0.25 0.00 0.25 0.50 0.75 1.00
0 1 2 3 4 5 6


Robust Scaling
salesQ5
salesQ4
salesQ3
salesQ2
salesQ1
Density
Sales Value
−2 −1 0 1 2 3 4
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75

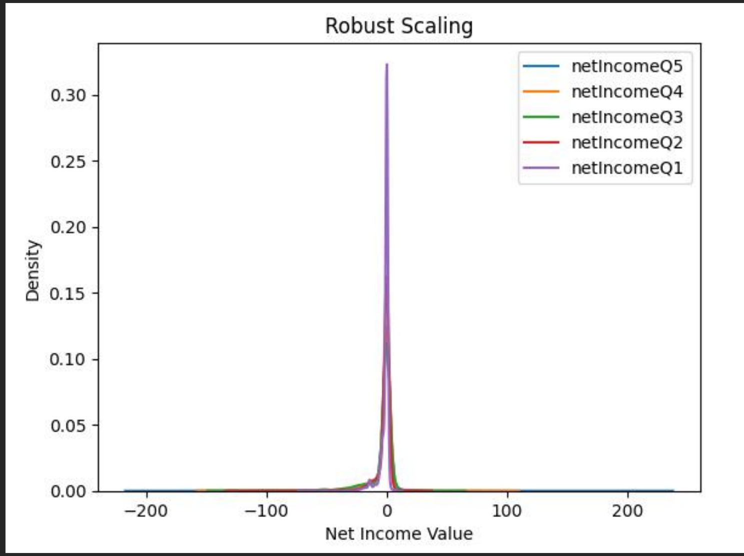

Standard Scaling
salesQ5
salesQ4
salesQ3
salesQ2
salesQ1
Density
Sales Value
−2 0 2 4 6
0.0 0.2 0.4 0.6 0.8 1.0


Min-Max Scaling
salesQ5
salesQ4
salesQ3
salesQ2
salesQ1
Density
Sales Value
0.0 0.2 0.4 0.6 0.8 1.0
0 1 2 3 4 5 6 7 8

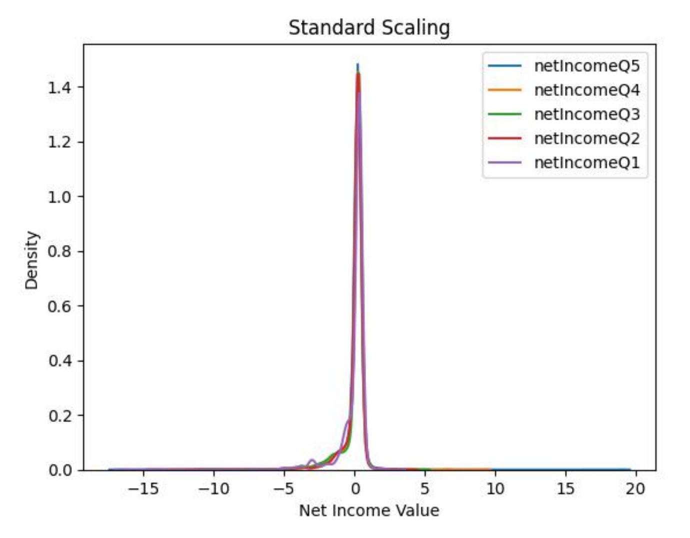
Net Income



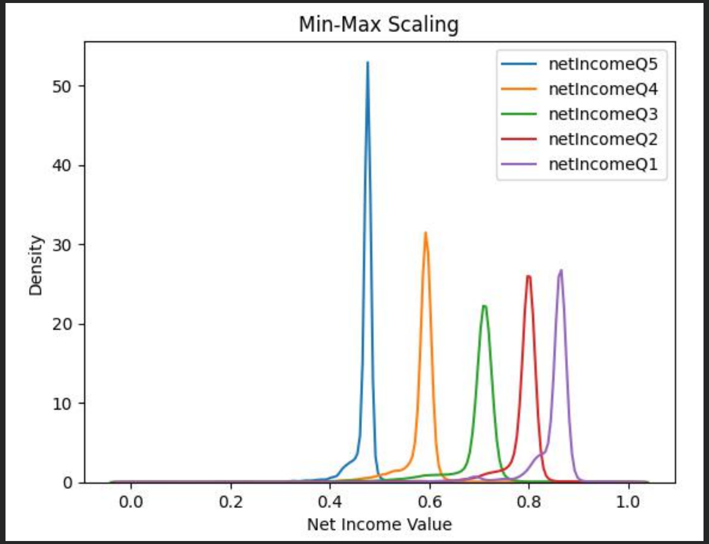
Before Scaling
netIncomeQ5
netIncomeQ4
netIncomeQ3
netIncomeQ2
netIncomeQ1
Density
Net Income Value



Robust Scaling
netIncomeQ5
netIncomeQ4
netIncomeQ3
netIncomeQ2
netIncomeQ1
Density
Net Income Value

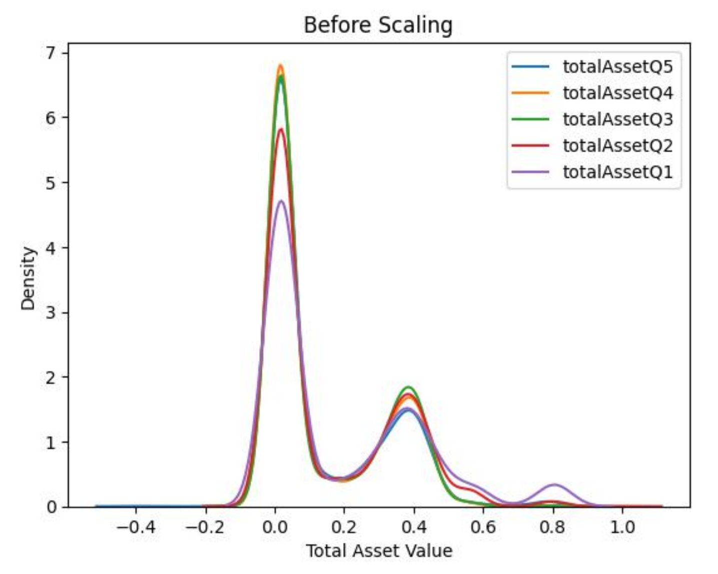


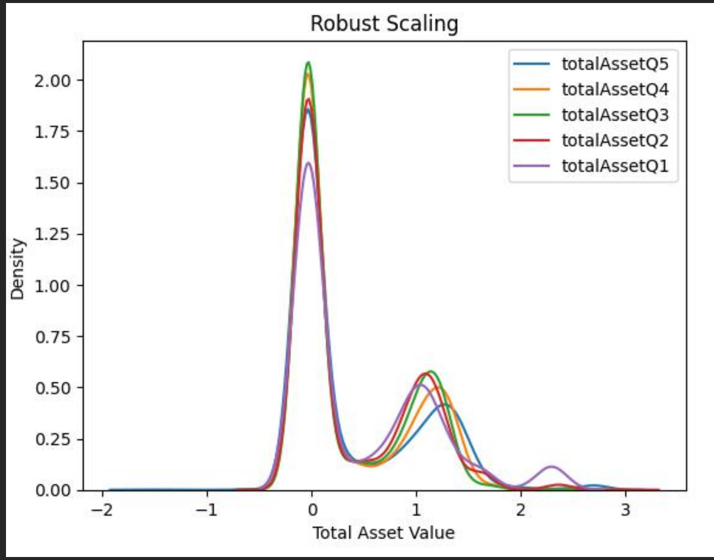
Standard Scaling
netIncomeQ5
netIncomeQ4
netIncomeQ3
netIncomeQ2
netIncomeQ1
Density
Net Income Value

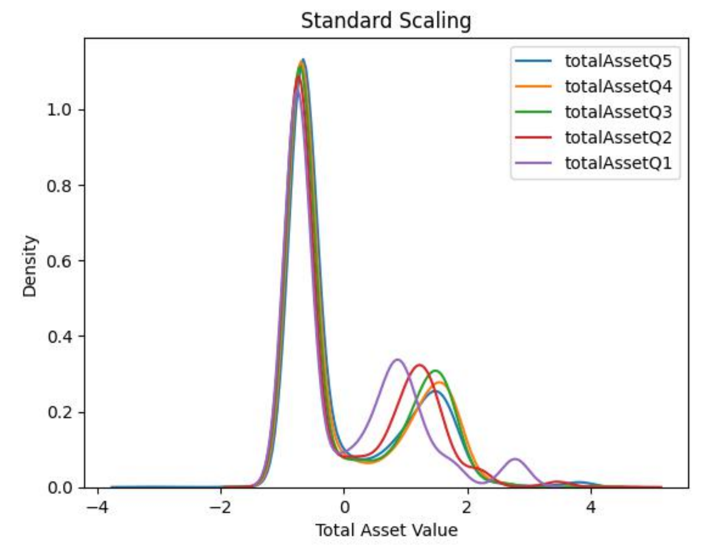


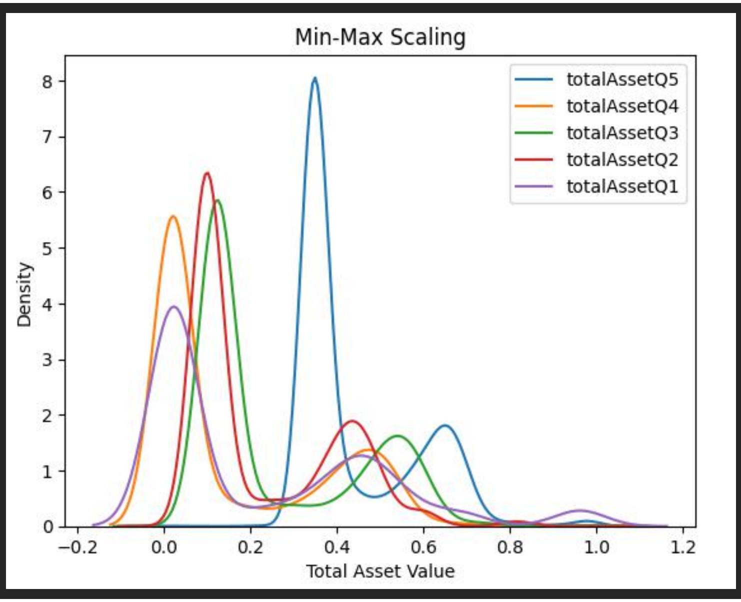
Min-Max Scaling
netIncomeQ5
netIncomeQ4
netIncomeQ3
netIncomeQ2
netIncomeQ1
Density
Net Income Value

Total Asset

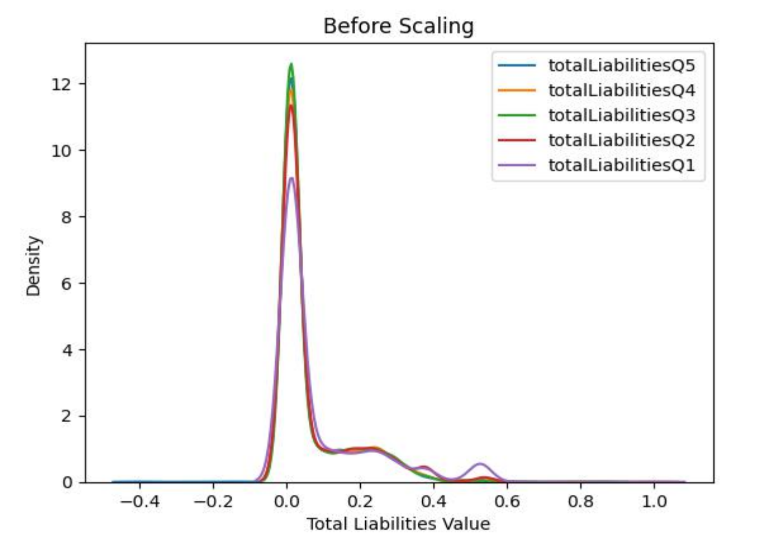

Before Scaling
totalAssetQ5
totalAssetQ4
totalAssetQ3
totalAssetQ2
totalAssetQ1
Density
Total Asset Value

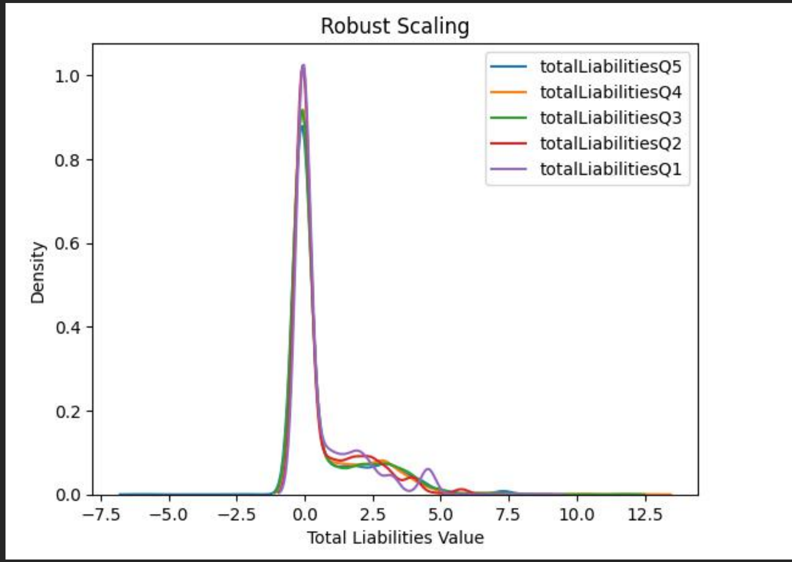

Robust Scaling
totalAssetQ5
totalAssetQ4
totalAssetQ3
totalAssetQ2
totalAssetQ1
Density
Total Asset Value

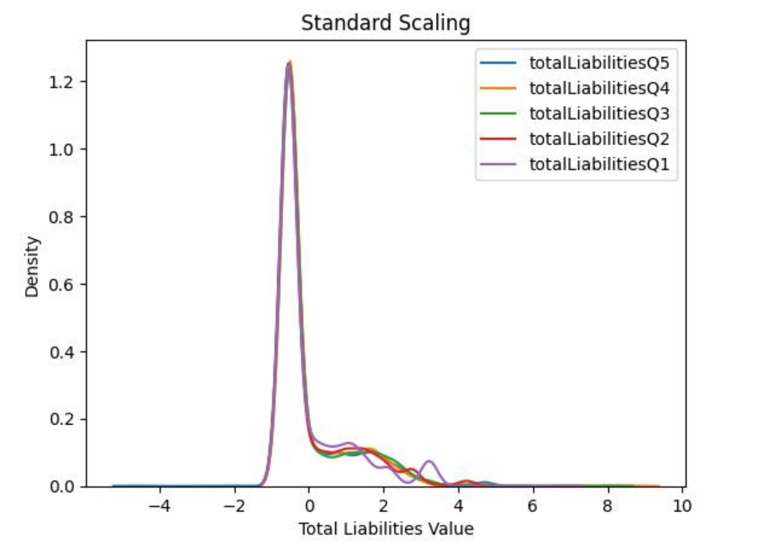

Standard Scaling
totalAssetQ5
totalAssetQ4
totalAssetQ3
totalAssetQ2
totalAssetQ1
Density
Total Asset Value

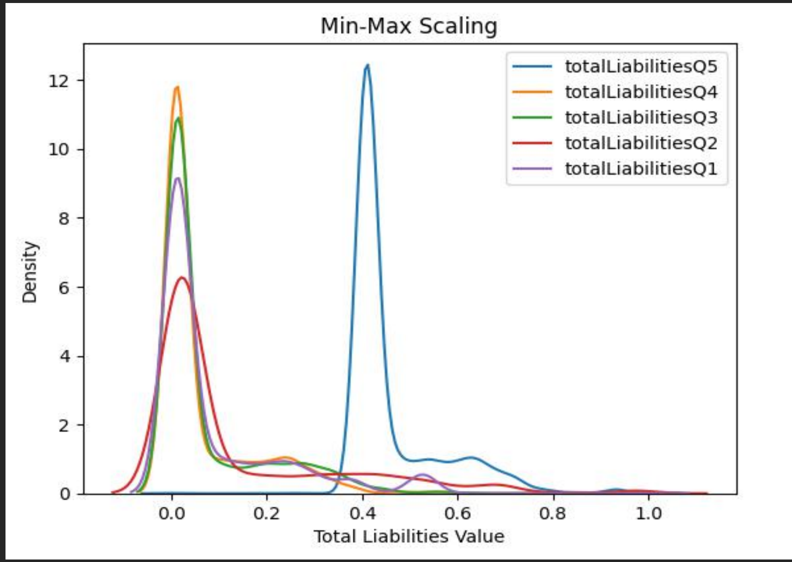

Min-Max Scaling
totalAssetQ5
totalAssetQ4
totalAssetQ3
totalAssetQ2
totalAssetQ1
Density
Total Asset Value

Total Liabilities

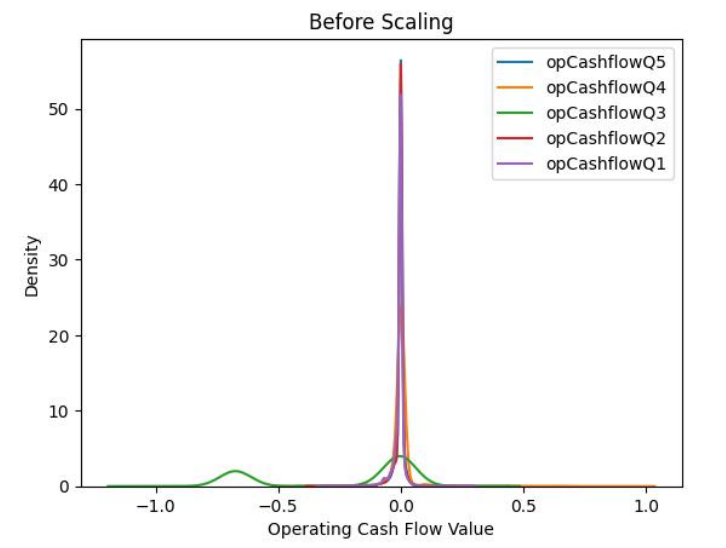


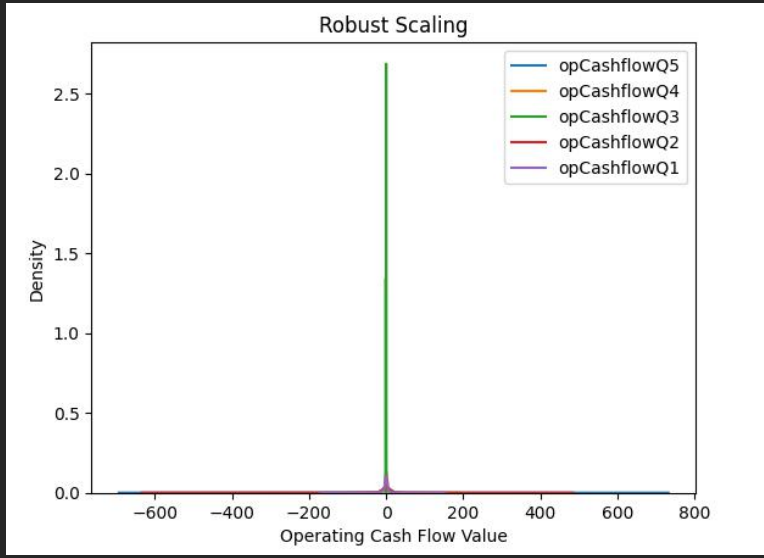


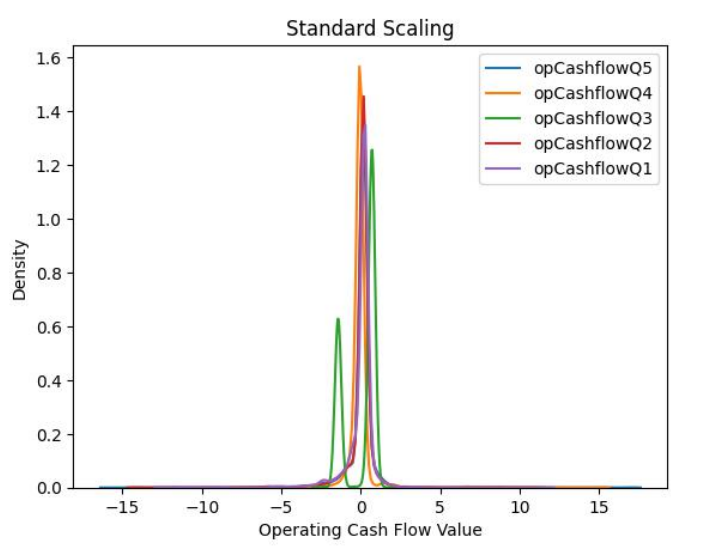


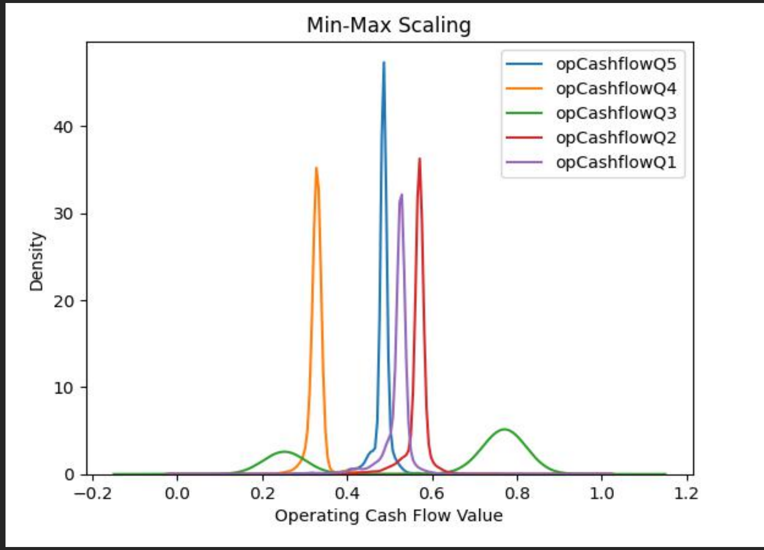


Operating Cash Flow

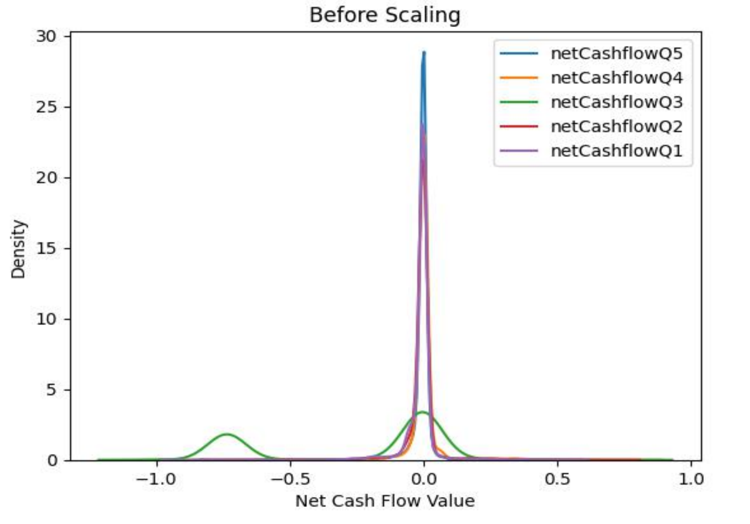


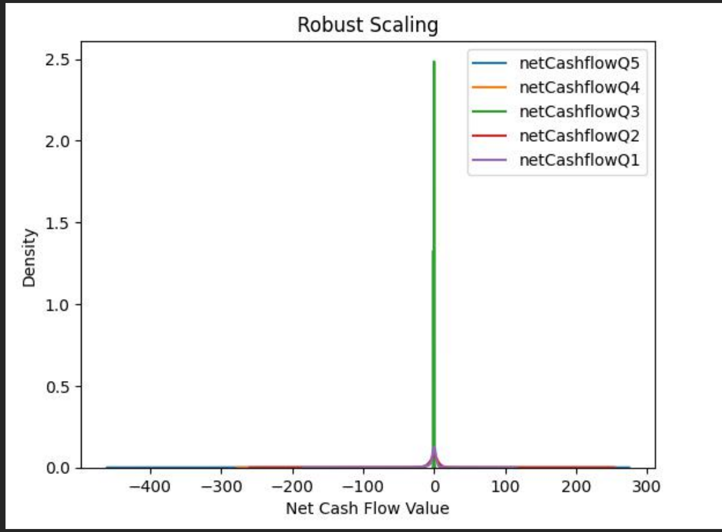


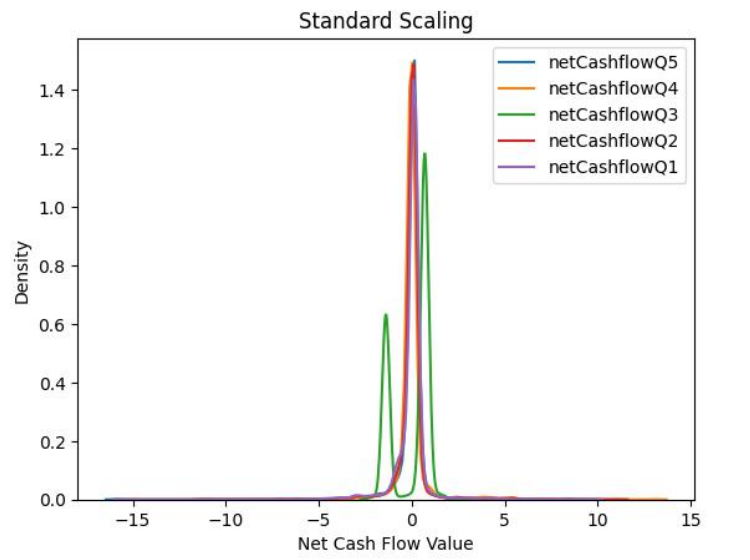


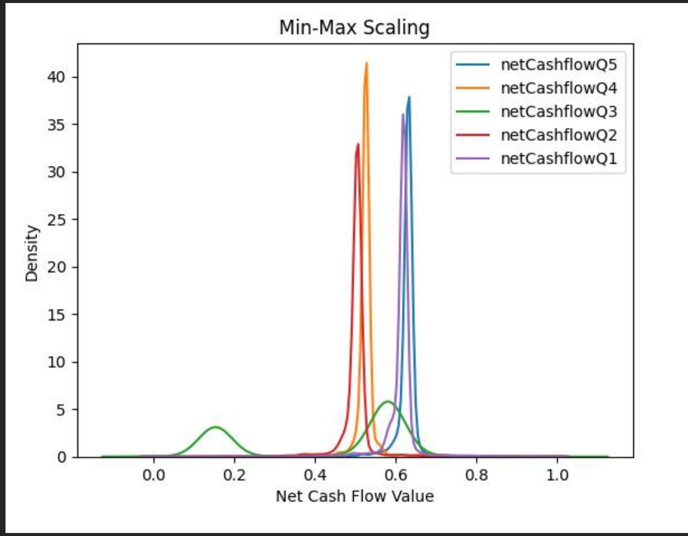


Net Cash Flow

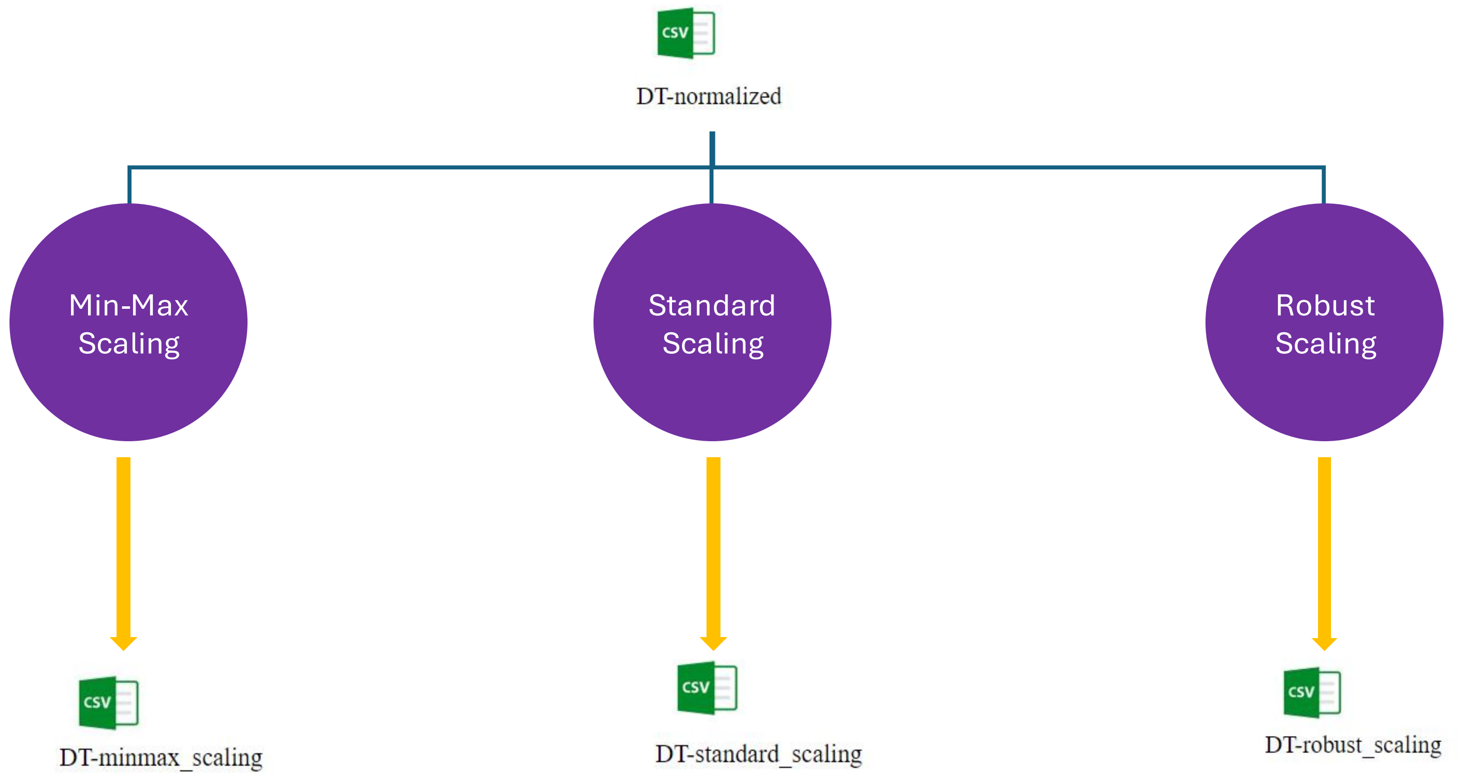








4. Data Transformation – Resulting output data



4. Data Transformation – Data Encoding

- We used the following to do our data encoding API: sklearn.preprocessing.**LabelEncoder**
- We transform status columns from "listed", "delisted" to 0 and 1

Data Encoding

```
encoder = LabelEncoder()

df3_cleaned['status'] = encoder.fit_transform(df3_cleaned['status'])
df3_cleaned
```

	status	salesQ5	salesQ4	salesQ3	salesQ2	salesQ1	netIncomeQ5	netIncomeQ4	netIncomeQ3	netIncomeQ2	...
0	0	4.744901e-03	4.260704e-03	5.315800e-03	0.004351	0.004593	-0.000097	-8.839976e-05	-5.628035e-05	-1.165012e-04	...
1	0	4.545807e-01	4.081927e-01	5.092752e-01	0.416830	0.440023	0.000005	9.235170e-06	1.633772e-05	3.597637e-06	...
2	0	4.545819e-01	4.081938e-01	5.092765e-01	0.416832	0.440024	0.000001	-1.854457e-07	-5.563370e-07	-2.781685e-07	...
3	0	4.744901e-03	4.260704e-03	5.315800e-03	0.004351	0.004593	-0.000097	-8.839976e-05	-5.628035e-05	-1.165012e-04	...
4	0	4.744901e-03	4.260704e-03	5.315800e-03	0.004351	0.004593	-0.000097	-8.839976e-05	-5.628035e-05	-1.165012e-04	...
...	...	...	...	...	...	...	...	...	...	...	...
2795	1	5.586174e-01	5.016129e-01	1.473739e-03	0.001158	0.540728	-0.001010	3.249664e-04	6.626467e-04	5.744568e-05	...
2796	1	1.895937e-05	1.004333e-06	7.850978e-08	0.332960	0.351486	0.004391	3.952301e-03	3.445666e-03	2.209753e-03	...
2797	1	2.108767e-03	2.479783e-03	1.443247e-03	0.000999	0.911613	0.000973	1.195231e-03	5.832293e-04	6.583507e-04	...
2798	1	7.998089e-09	2.132824e-08	9.331104e-08	0.599251	0.632594	-0.000194	3.176116e-03	1.404976e-03	-7.768811e-06	...
2799	1	1.532547e-03	2.321270e-03	2.031616e-03	0.001533	0.888784	-0.000010	7.094075e-04	-8.396840e-04	5.323506e-04	...

4. Data Transformation – ML Result

Condition	SVM Accuracy
Before Data Transformation	0.7976
After Data Transformation:	
MinMaxScaler	0.8619
StandardScaler	0.8810
RobustScaler	0.7976

5. Data Visualization and ETL-Use of Different Tools

- Python (Pandas, SQLite, BeautifulSoup, csv, requests, selenium), API, Google Collab.
- AWS Lambda, AWS SQS for creating multi-threading web scrapping tool.
- Delta Lake From Databricks (https://docs.databricks.com/en/delta/index.html).
- AWS Glue for ETL pipeline.

5.ETL to find Features-Feature Selection

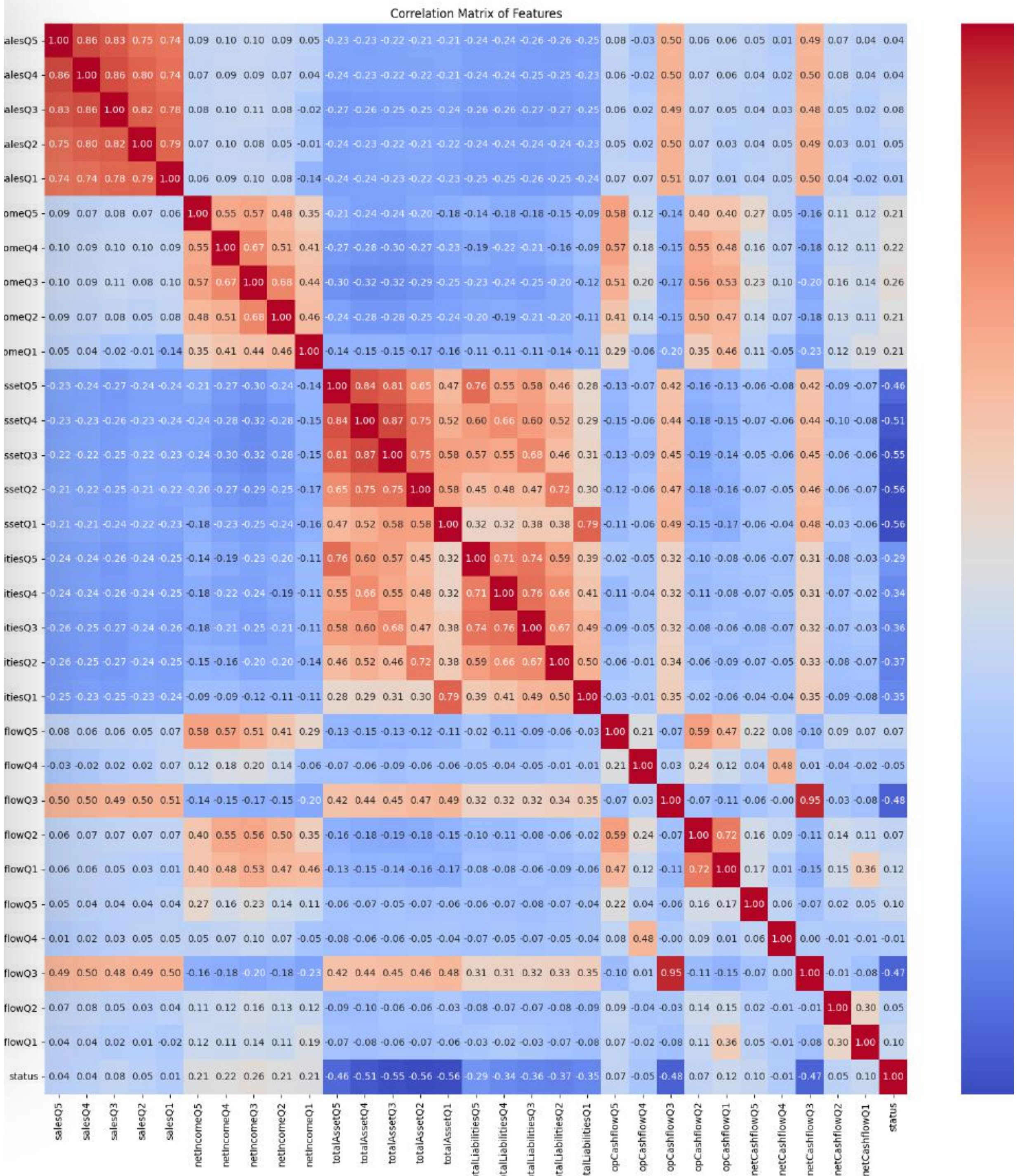
Removing irrelevant Features

We only keep Times Series data

- ~~Stock Symbol~~
- ~~Company Name~~
- ~~Exchange Center~~
- ~~IPO Delisted Date~~(data leakage, in a real world, you won't know when your company is delisted)

5.ETL to find Features-Feature Selection

- Set correlation threshold=0.85
- Drop columns:
- ~~'salesQ3', 'totalAssetQ1', 'totalLiabilitiesQ4', 'opCashflowQ3', 'salesQ5', 'totalAssetQ5', 'totalAssetQ4', 'salesQ4', 'salesQ2'~~



5.ETL to find Features- Feature Selection ML accuracy

Description	SVM Accuracy
Before Feature Selection	0.8821
After Feature Selection	0.8833

5.ETL to find Features-Features Transformation

We perform PCA for features Transformation

Why?

-It maximize the variance of the dataset.

-filter out noise and less significant details

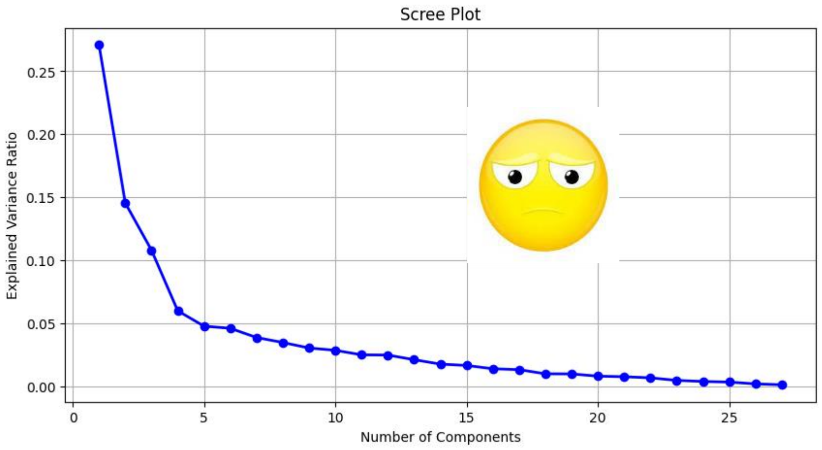
-decrease the feature correlation

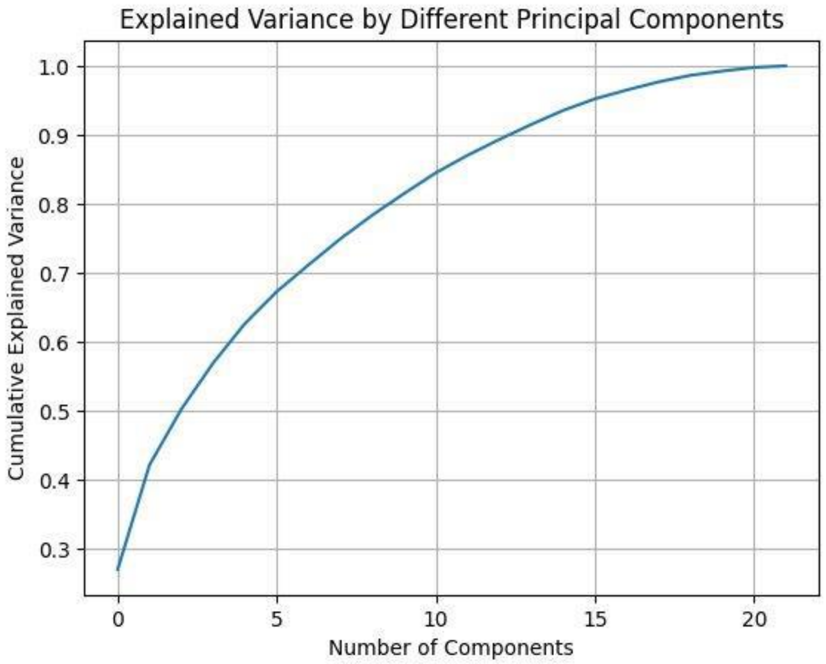
-easy for visualize as less columns remains

5.ETL to find Features-Hyperparameter Tuning

Objective: Let N =n of components, Select N to cover 95% of variance,   1<=N<=31

Plot the variance for each values:





Best hyperparameter N =18

5.ETL to find Features-ML Accuracy

```
principal_df.head()
```

	PCA1	PCA2	PCA3	PCA4	PCA5	PCA6	PCA7	PCA8	PCA9
0	-1.676521	-0.257513	2.018692	-0.043753	-0.208860	-0.133962	-0.007171	-0.106446	0.047713
1	-2.640882	1.392012	-2.090201	0.535550	-0.188258	-0.299712	-0.233052	-0.198329	-0.171183
2	-2.648911	1.397961	-2.087499	0.535232	-0.187373	-0.298361	-0.231196	-0.196152	-0.171992
3	-1.676521	-0.257513	2.018692	-0.043753	-0.208860	-0.133962	-0.007171	-0.106446	0.047713
4	-1.676521	-0.257513	2.018692	-0.043753	-0.208860	-0.133962	-0.007171	-0.106446	0.047713

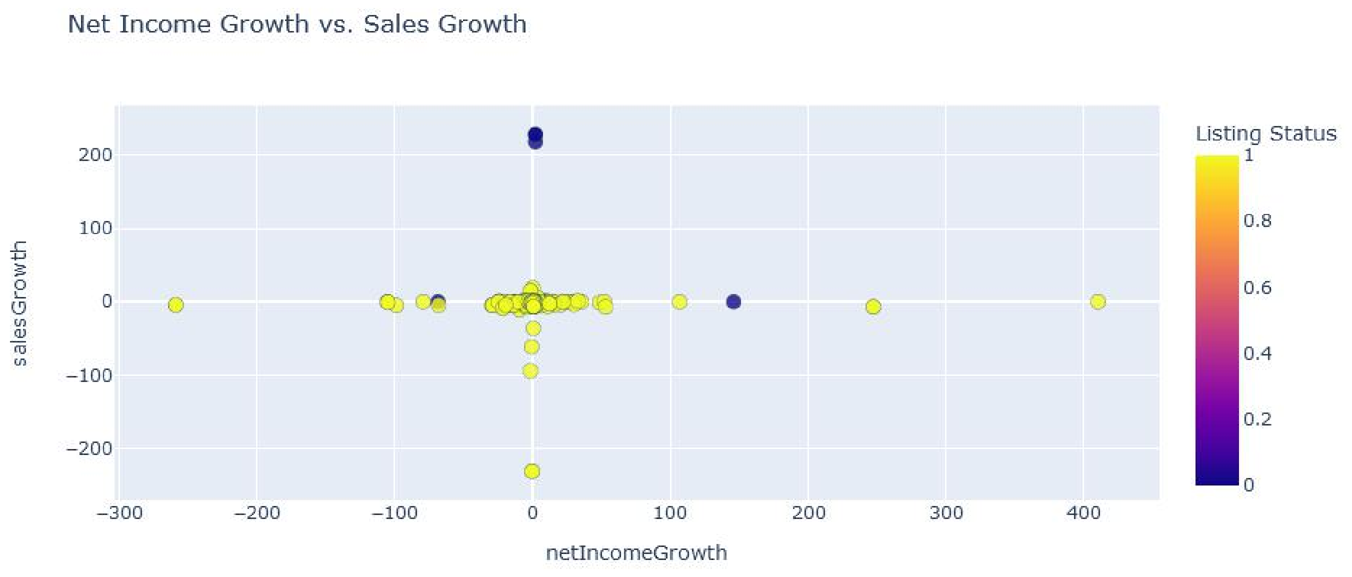
	PCA10	PCA11	PCA12	PCA13	PCA14	PCA15	PCA16	PCA17	PCA18
0	0.058022	-0.003011	-0.013878	0.019516	-0.026920	-0.007651	-0.019073	0.010205	-0.047502
1	-0.268602	-0.006329	0.018682	0.179673	0.095123	0.066136	0.057468	-0.029552	-0.073766
2	-0.266810	-0.006455	0.018430	0.179822	0.095117	0.066745	0.057600	-0.029385	-0.072753
3	0.058022	-0.003011	-0.013878	0.019516	-0.026920	-0.007651	-0.019073	0.010205	-0.047502
4	0.058022	-0.003011	-0.013878	0.019516	-0.026920	-0.007651	-0.019073	0.010205	-0.047502

Description	SVM Accuracy
Before Feature Selection and Transformation	0.8821
After Feature Selection and Transformation	0.9000

5.1. Data Visualization/Dashboard

- Used Python Dash + Plotly for visualization
- Some problems: Timing across time-series data only has 5 points(Q1-Q5).
- Making a few types of plots less reliable.

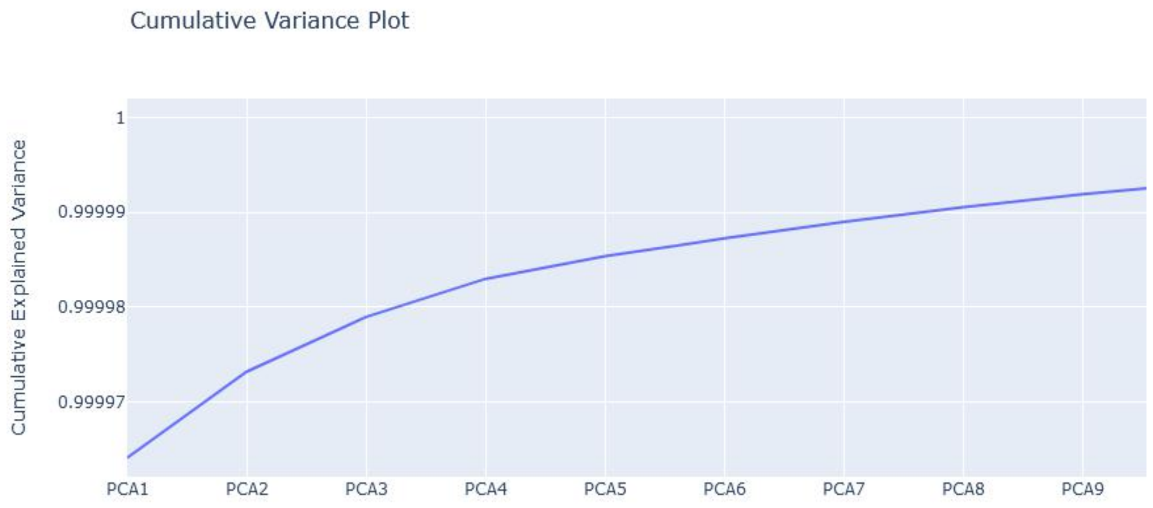
Net Income Growth vs. Sales Growth vs. Status



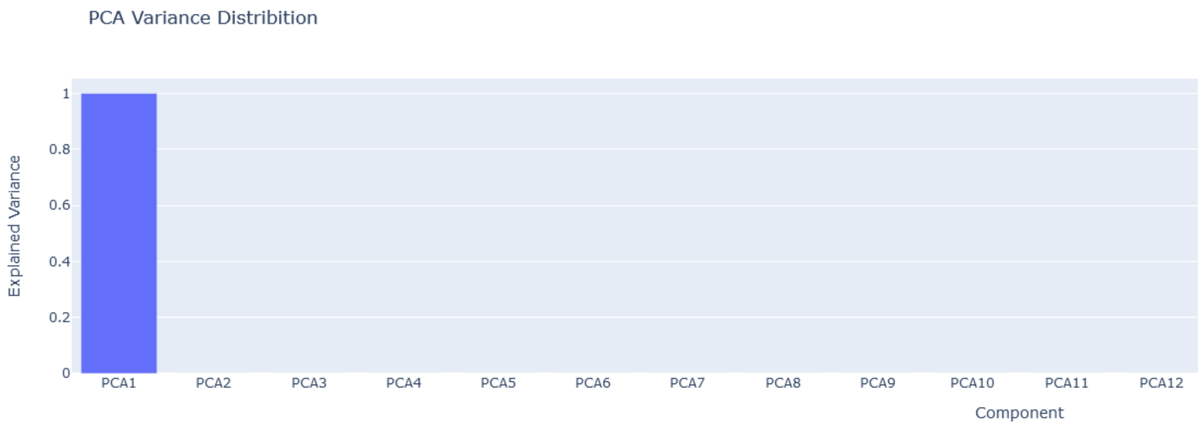
Scatter plot be looking like this 😭

5.1. Data Visualization/Dashboard

- Principal component analysis is used to help with this problem.

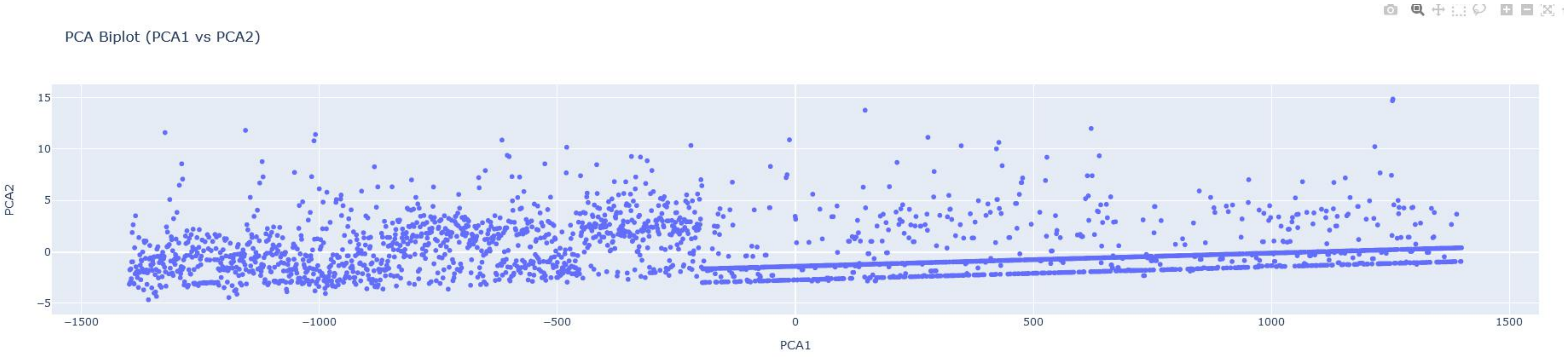


PCA Visualization Dashboard

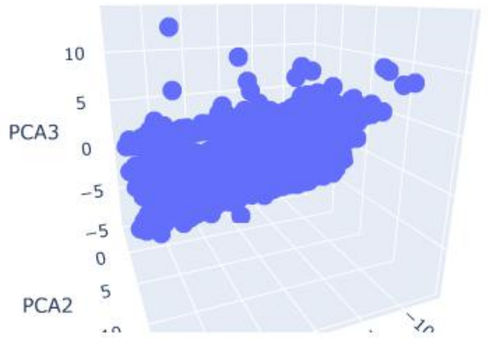


5.1. Data Visualization/Dashboard

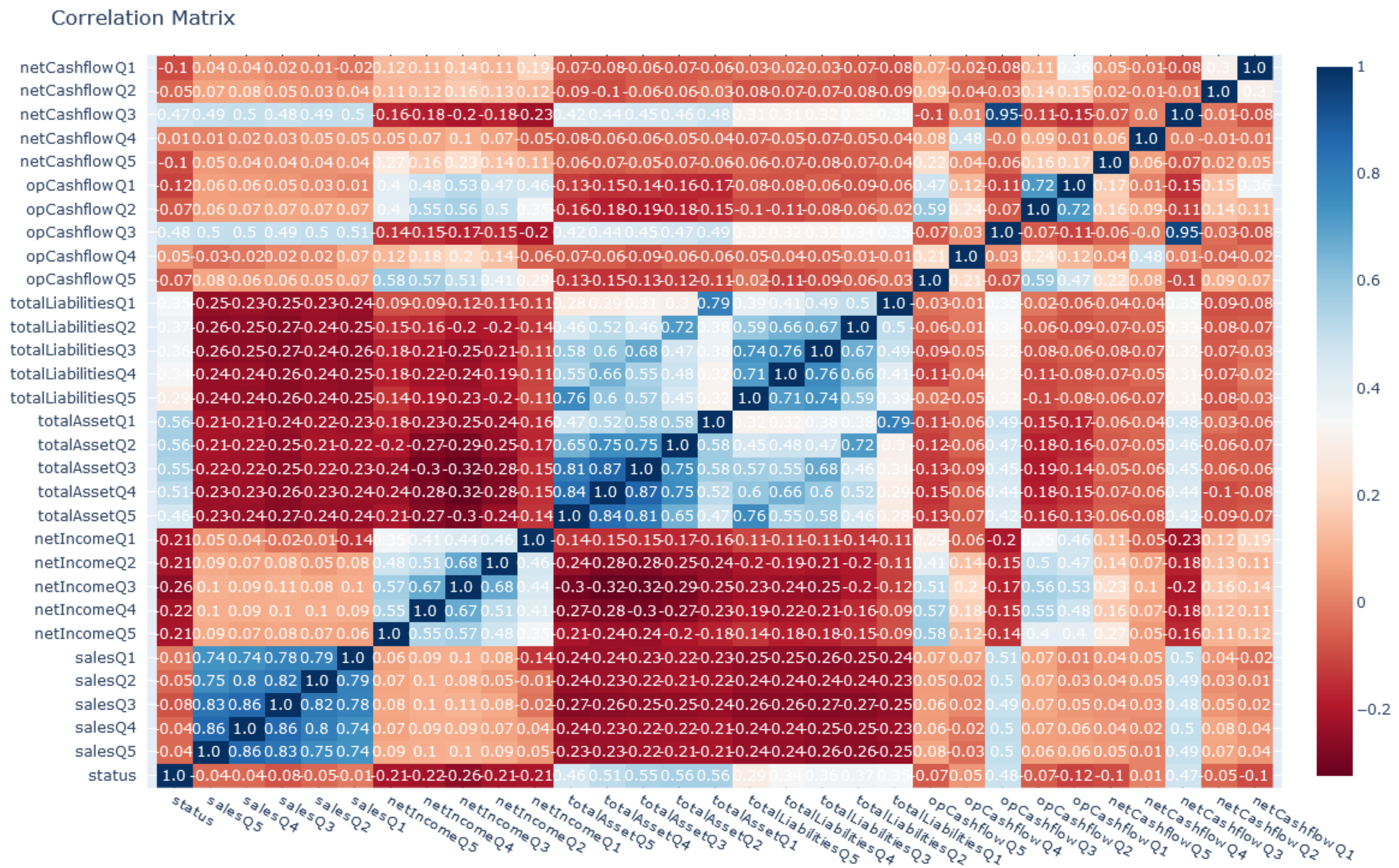
PCA Biplot (PCA1 vs PCA2)



3D PCA Biplot (PCA1 vs PCA2 vs PCA3)



5.1. Data Visualization/Dashboard



Correlation Matrix

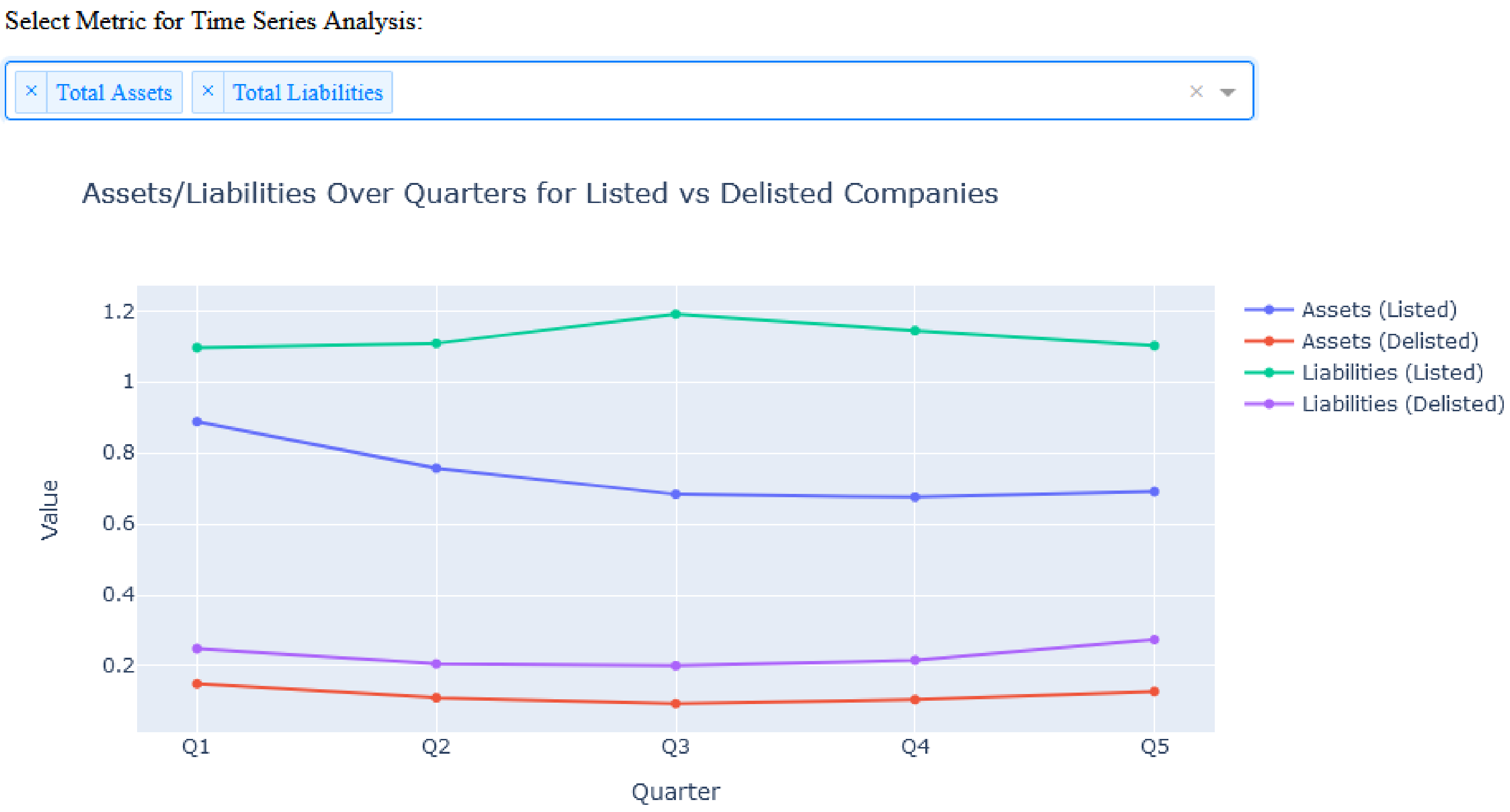
5.1. Data Visualization/Dashboard

Minmax-scaled Dataset

5.1. Data Visualization/Dashboard

Standard-scaled Dataset

5.1. Data Visualization/Dashboard



Robust-scaled Dataset