A Case Study Report on

Bank run prediction for ICICI Bank

by

Krithik D. Patil (21102193)
Ajayraj P. Rathod (21102085)

in the subject

Applied Data Science

Department of Computer Engineering

A.P. Shah Institute of Technology
G.B. Road, Kasarvadavli, Thane(W), Mumbai-400615

UNIVERSITY OF MUMBAI

Academic Year 2024-2025

Contents

1. Problem Statement:

Bank runs occur when a significant number of customers withdraw their deposits due to concerns about a bank's solvency, leading to liquidity crises and potential financial instability. Early detection of such events is crucial for risk management and regulatory intervention. This project aims to develop a predictive model based on anomaly detection to identify potential bank run scenarios using financial and market indicators. The model will analyze historical data, including deposit trends, borrowing ratios, non-performing assets (NPA), capital adequacy ratios (CAR), and stock market performance. By leveraging unsupervised learning techniques such as Isolation Forest, Autoencoders, and One-Class SVM, the system will detect deviations from typical banking behavior that may indicate financial distress. The primary objective is to classify periods of unusual financial activity as potential precursors to a bank run. The outcome of this research can help financial institutions and regulators implement early warning systems, mitigate risks, and take proactive measures to maintain banking stability. This solution will be valuable for banks, investors, and policymakers to enhance financial security and prevent systemic failures in the banking sector.

2. Description of dataset:

The dataset contains financial and stock performance metrics of ICICI Bank across multiple periods. Here’s a breakdown of the key columns:

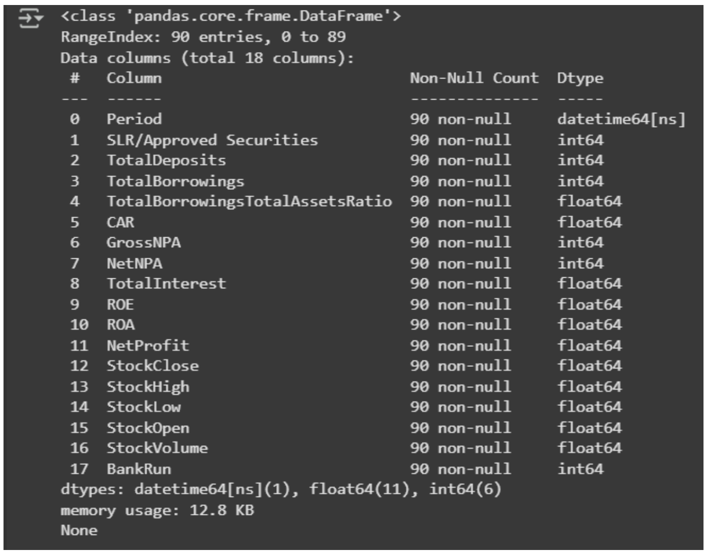
- Period – Date of the financial report (quarterly or annual).
- SLR/Approved Securities – Investments in Statutory Liquidity Ratio (SLR) or approved securities.
- Total Deposits – Total customer deposits.
- Total Borrowings – Total borrowings made by the bank.
- Total Borrowings to Total Assets Ratio – Ratio of borrowings to total assets.
- CAR (Capital Adequacy Ratio) – Measures a bank’s capital in relation to its risk-weighted assets.
- Gross NPA (Non-Performing Assets) – Total amount of bad loans before provisions.
- Net NPA – Bad loans after deducting provisions.
- Total Interest – Interest income earned by the bank.
- ROE (Return on Equity) – Measures profitability relative to shareholders' equity.
- ROA (Return on Assets) – Measures profitability relative to total assets.
- Net Profit – Bank's total profit.
- StockClose – Closing stock price for the period.
- StockHigh – Highest stock price in the period.
- StockLow – Lowest stock price in the period.
- StockOpen – Opening stock price in the period.
- StockVolume – Total number of shares traded.
- BankRun – Indicator of a financial crisis or liquidity issues (0 or 1).

This dataset can be useful for financial analysis, stock market trends, risk assessment, and predictive modeling.

3. Descriptive statistics and inferential statistics:

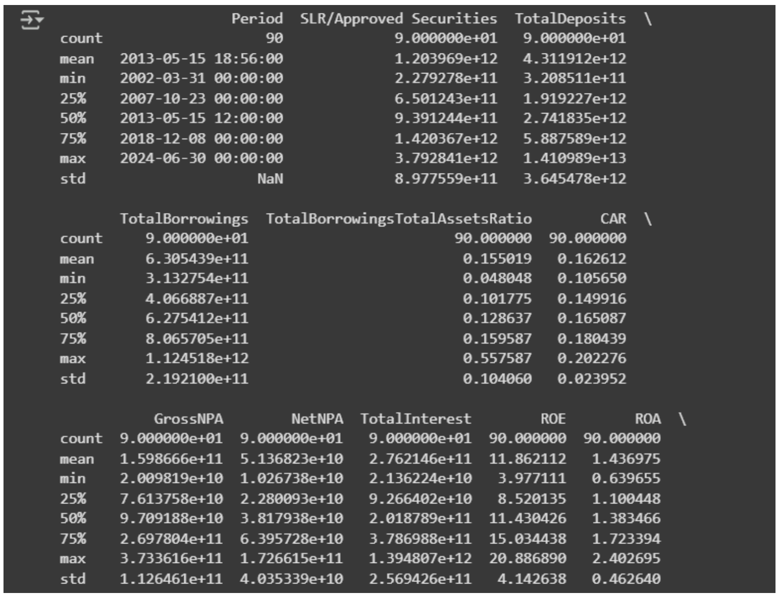
Descriptive statistics summarize and describe the main features of a dataset. In this code, several descriptive statistics are calculated and visualized:

df.info(): Provides a summary of the dataframe, including the data types of each column and the number of non-null values. This helps understand the structure and completeness of the data.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Period                            90 non-null     datetime64[ns]
 1   SLR/Approved Securities           90 non-null     int64
 2   TotalDeposits                     90 non-null     int64
 3   TotalBorrowings                   90 non-null     int64
 4   TotalBorrowingsTotalAssetsRatio   90 non-null     float64
 5   CAR                               90 non-null     float64
 6   GrossNPA                          90 non-null     int64
 7   NetNPA                            90 non-null     int64
 8   TotalInterest                     90 non-null     float64
 9   ROE                               90 non-null     float64
 10  ROA                               90 non-null     float64
 11  NetProfit                         90 non-null     float64
 12  StockClose                        90 non-null     float64
 13  StockHigh                         90 non-null     float64
 14  StockLow                          90 non-null     float64
 15  StockOpen                         90 non-null     float64
 16  StockVolume                       90 non-null     float64
 17  BankRun                           90 non-null     int64
dtypes: datetime64[ns](1), float64(11), int64(6)
memory usage: 12.8 KB
None
```

df.describe(): Calculates and displays summary statistics for numerical columns, such as count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum. These statistics give an overview of the distribution and central tendency of the data.



```
                    Period  SLR/Approved Securities  TotalDeposits  \
count                   90             9.000000e+01   9.000000e+01
mean   2013-05-15 18:56:00             1.203969e+12   4.311912e+12
min    2002-03-31 00:00:00             2.279278e+11   3.208511e+11
25%    2007-10-23 00:00:00             6.501243e+11   1.919227e+12
50%    2013-05-15 12:00:00             9.391244e+11   2.741835e+12
75%    2018-12-08 00:00:00             1.420367e+12   5.887589e+12
max    2024-06-30 00:00:00             3.792841e+12   1.410989e+13
std                    NaN             8.977559e+11   3.645478e+12

       TotalBorrowings  TotalBorrowingsTotalAssetsRatio        CAR  \
count     9.000000e+01                        90.000000  90.000000
mean      6.305439e+11                         0.155019   0.162612
min       3.132754e+11                         0.048048   0.105650
25%       4.066887e+11                         0.101775   0.149916
50%       6.275412e+11                         0.128637   0.165087
75%       8.065705e+11                         0.159587   0.180439
max       1.124518e+12                         0.557587   0.202276
std       2.192100e+11                         0.104060   0.023952

           GrossNPA        NetNPA  TotalInterest        ROE        ROA  \
count  9.000000e+01  9.000000e+01   9.000000e+01  90.000000  90.000000
mean   1.598666e+11  5.136823e+10   2.762146e+11  11.862112   1.436975
min    2.009819e+10  1.026738e+10   2.136224e+10   3.977111   0.639655
25%    7.613758e+10  2.280093e+10   9.266402e+10   8.520135   1.100448
50%    9.709188e+10  3.817938e+10   2.018789e+11  11.430426   1.383466
75%    2.697804e+11  6.395728e+10   3.786988e+11  15.034438   1.723394
max    3.733616e+11  1.726615e+11   1.394807e+12  20.886890   2.402695
std    1.126461e+11  4.035339e+10   2.569426e+11   4.142638   0.462640
```

Correlation Heatmap (sns.heatmap): Visualizes the correlation between different financial indicators. This helps understand relationships between variables, such as the negative correlation between 'TotalBorrowingsTotalAssetsRatio' and stock data, indicating that higher borrowings might be associated with lower stock prices.

Distribution Plot (sns.distplot): Shows the distribution of the 'CAR' (Capital Adequacy Ratio). This visualization reveals that most records have CAR ratios around 0.16 and 0.17.

Box Plot (sns.boxplot): Provides a visual representation of the distribution of 'CAR', showing the median, quartiles, and potential outliers.

Time Series Plots (sns.lineplot): Illustrate the trends of various financial indicators (CAR, SLR/Approved Securities, TotalBorrowings, TotalDeposits) over time. These plots help identify patterns and anomalies, such as the peaks and dips during financial crises or periods of economic instability.

Scatter Plot (sns.relplot): Shows the relationship between 'TotalDeposits' and 'CAR', allowing for the identification of potential correlations or patterns.

Seasonal Decomposition: The application of seasonal_decompose from statsmodels.tsa.seasonal helps to break down the time series data into its components of trend, seasonality, and residual to study any repetitive patterns or cyclical behavior.

These descriptive statistics and visualizations provide a comprehensive overview of the financial data, highlighting key characteristics, distributions, and relationships between variables.

Inferential statistics aim to draw conclusions or make predictions about a larger population based on a sample of data. While the provided code primarily focuses on descriptive analysis and anomaly detection, there are elements that touch upon inferential statistics:

Anomaly Detection with Isolation Forest and DBSCAN: These techniques are used to identify unusual data points (anomalies) that deviate significantly from the expected patterns in the data. While not strictly inferential, these methods can be used to infer potential risks or unusual events within the financial data.

Clustering with DBSCAN: DBSCAN is employed to group similar financial periods based on the selected features. This can be considered an inferential approach as it attempts to categorize and understand the underlying structure of the data, potentially revealing insights into different financial states or behaviors.

Inference: Total borrowings to total assets ratio is negatively correlated with the stock data. Higher borrowings are bad for financial health and lead to stock selling.

Inference: Most records have a CAR ratio of 0.16 and 0.17.
Inference: CAR plot shows a dip in 2008-2009 and 2019-2020 due to financial crisis and COVID-19.

Inference: SLR deposits, a safe investment, peaked during 2008-2009 and 2019-2020, potentially due to RBI mandates.

Inference: Borrowings decreased during 2008 and 2019 to maintain financial health.

Inference: Deposits increased significantly during 2008 and 2019, which could indicate potential bank runs.

Inference: Isolation Forest detected anomalies during 2008 and 2019.

Inference: DBSCAN detected anomalies during 2021, 2022, and 2024.

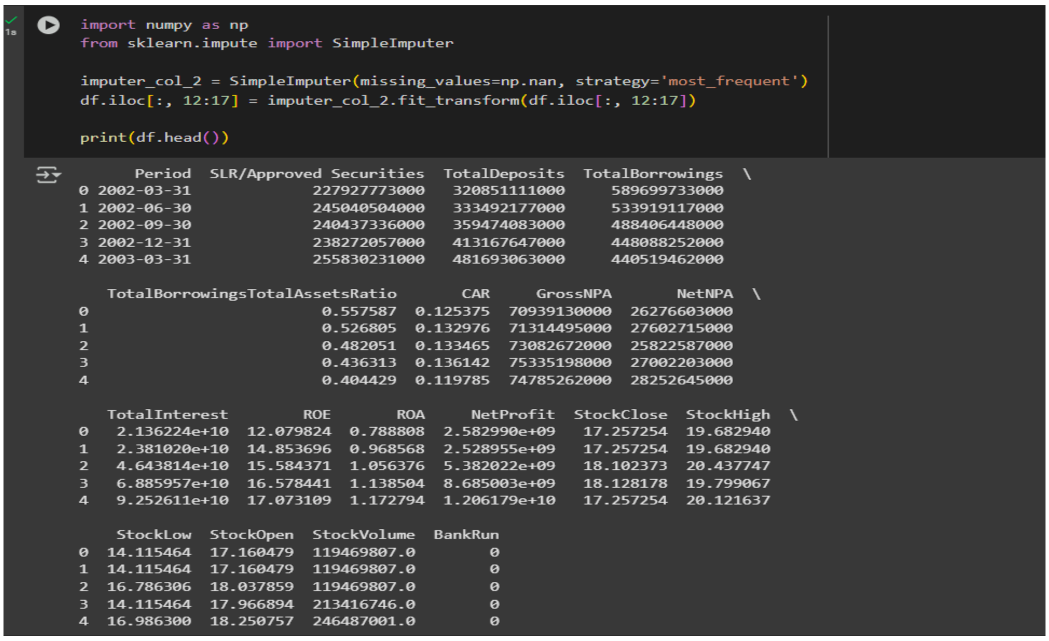
Inference: DBSCAN is 67% accurate and Isolation Forest is 100% accurate, suggesting overfitting.

These statements relate observations about trends and patterns in the data to external factors like financial crises or regulations. This type of interpretation and linking with broader context moves beyond pure description into the realm of inference, aiming to understand potential causes or implications.

4. Data Pre-processing:

Handling Missing Values - The code used SimpleImputer to fill missing values.

a) import numpy as np: This line imports the numpy library, which is used for numerical operations in Python. It's often used in data science for working with arrays and matrices.
b) from sklearn.impute import SimpleImputer: This line imports the SimpleImputer class from the sklearn.impute module. sklearn (scikit-learn) is a popular library for machine learning in Python. SimpleImputer is a tool specifically designed for filling in missing values.
c) imputer_col_2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent'): Here, we create an instance of the SimpleImputer class and store it in the variable imputer_col_2.
    - missing_values=np.nan: This tells the imputer to look for missing values represented by np.nan (Not a Number), which is a common way to denote missing data in pandas DataFrames.
    - strategy='most_frequent': This specifies the strategy for imputation. In this case, it's using the 'most_frequent' strategy, which means it will fill missing values with the most frequent value (the mode) in that particular column.
d) df.iloc[:, 12:17] = imputer_col_2.fit_transform(df.iloc[:, 12:17]): This is the core of the imputation process.
    - df.iloc[:, 12:17]: This selects a specific portion of your DataFrame (df). iloc is used for indexing by position. [:, 12:17] means selecting all rows (:) and columns from index 12 to 16 (17 is excluded).
    - imputer_col_2.fit_transform(): This method does two things:
        1. fit: It calculates the most frequent value for each column in the selected portion of the DataFrame.
        2. transform: It uses the calculated most frequent values to fill in the missing values (represented by np.nan) in the selected columns.
    - The result is then assigned back to the original DataFrame (df.iloc[:, 12:17] = ...), effectively replacing the missing values.



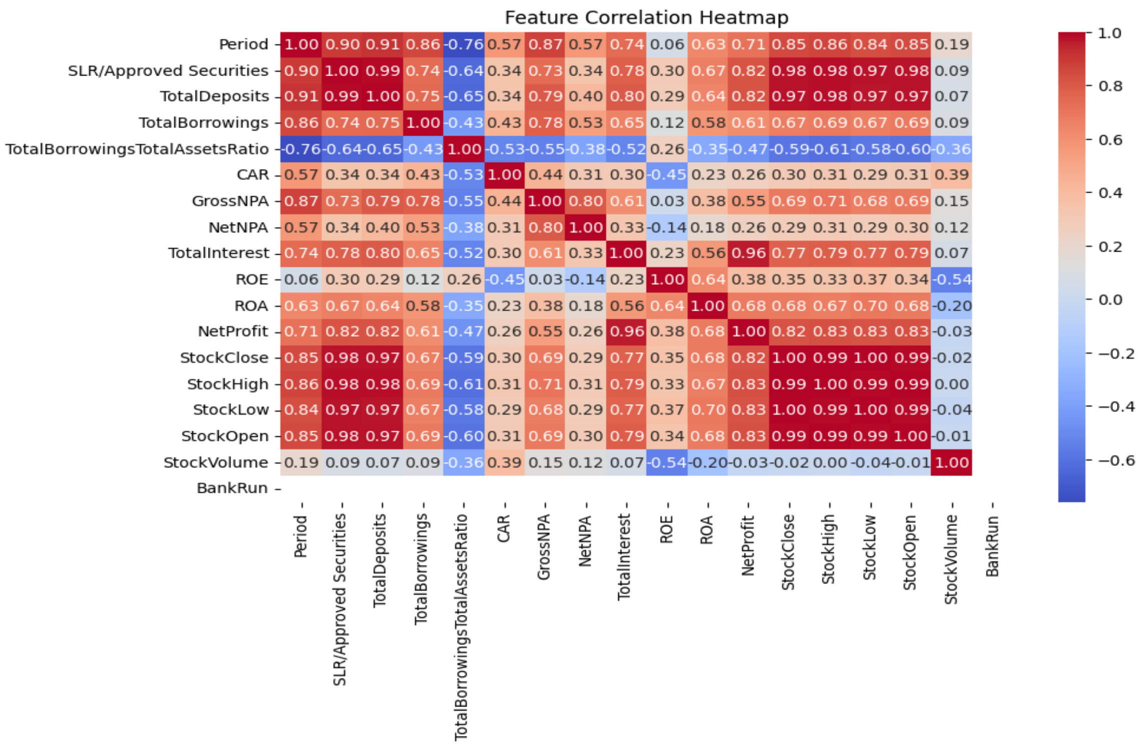
```
import numpy as np
from sklearn.impute import SimpleImputer

imputer_col_2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
df.iloc[:, 12:17] = imputer_col_2.fit_transform(df.iloc[:, 12:17])

print(df.head())
```

```
       Period  SLR/Approved Securities  TotalDeposits  TotalBorrowings  \
0 2002-03-31             227927773000   320851111000     589699733000
1 2002-06-30             245040504000   333492177000     533919117000
2 2002-09-30             240437336000   359474083000     488406448000
3 2002-12-31             238272057000   413167647000     448088252000
4 2003-03-31             255830231000   481693063000     440519462000

   TotalBorrowingsTotalAssetsRatio       CAR     GrossNPA         NetNPA  \
0                         0.557587  0.125375  70939130000  26276603000
1                         0.526805  0.132976  71314495000  27602715000
2                         0.482051  0.133465  73082672000  25822587000
3                         0.436313  0.136142  75335198000  27002203000
4                         0.404429  0.119785  74785262000  28252645000

   TotalInterest        ROE       ROA     NetProfit  StockClose  StockHigh  \
0   2.136224e+10  12.079824  0.788808  2.582990e+09   17.257254  19.682940
1   2.381020e+10  14.853696  0.968568  2.528955e+09   17.257254  19.682940
2   4.643814e+10  15.584371  1.056376  5.382022e+09   18.102373  20.437747
3   6.885957e+10  16.578441  1.138504  8.685003e+09   18.128178  19.799067
4   9.252611e+10  17.073109  1.172794  1.206179e+10   17.257254  20.121637

    StockLow  StockOpen  StockVolume  BankRun
0  14.115464  17.160479  119469807.0        0
1  14.115464  17.160479  119469807.0        0
2  16.786306  18.037859  119469807.0        0
3  14.115464  17.966894  213416746.0        0
4  16.986300  18.250757  246487001.0        0
```

5. Data visualization:

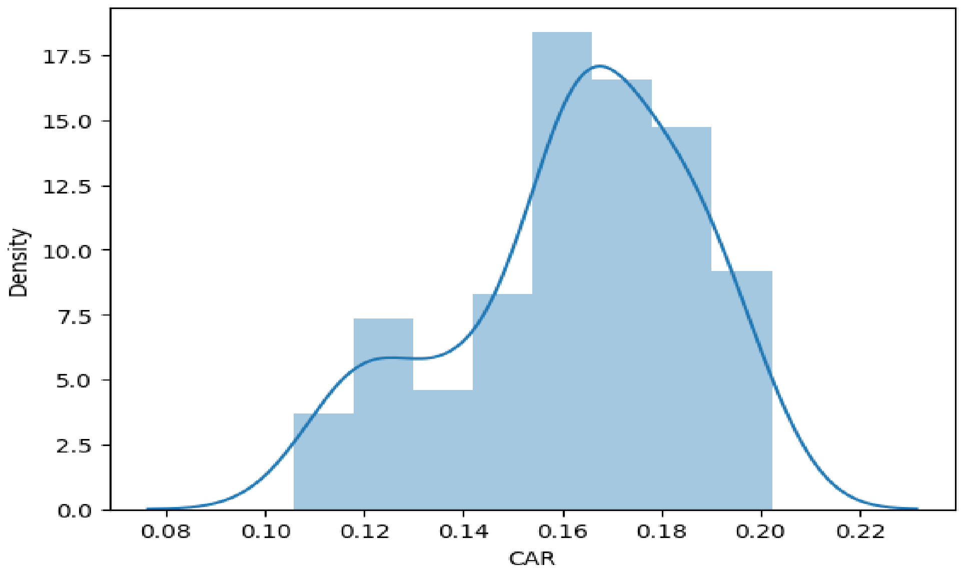
The code uses a variety of data visualizations to explore the financial data and present the results of anomaly detection. Here's a summary of the visualizations used:

Correlation Heatmap (sns.heatmap): Visualizes the correlation between different financial indicators. This helps understand relationships between variables, such as the negative correlation between 'TotalBorrowingsTotalAssetsRatio' and stock data, indicating that higher borrowings might be associated with lower stock prices.

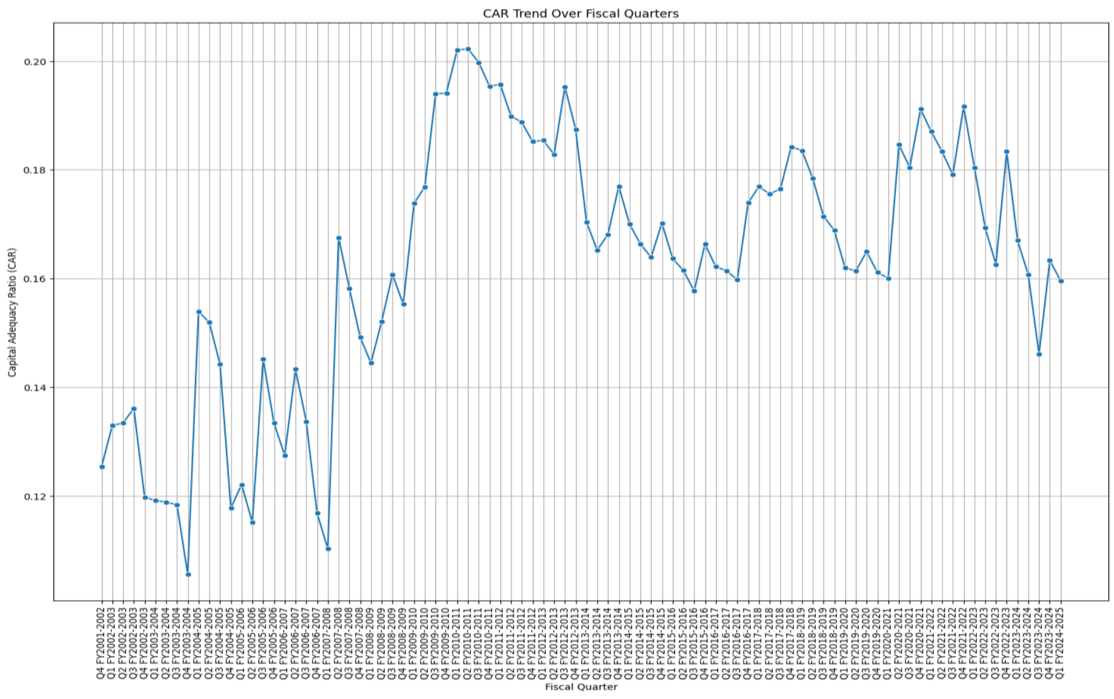


	Period	SLR/Approved Securities	TotalDeposits	TotalBorrowings	TotalBorrowingsTotalAssetsRatio	CAR	GrossNPA	NetNPA	TotalInterest	ROE	ROA	NetProfit	StockClose	StockHigh	StockLow	StockOpen	StockVolume	BankRun
Period	1.00	0.90	0.91	0.86	-0.76	0.57	0.87	0.57	0.74	0.06	0.63	0.71	0.85	0.86	0.84	0.85	0.19	
SLR/Approved Securities	0.90	1.00	0.99	0.74	-0.64	0.34	0.73	0.34	0.78	0.30	0.67	0.82	0.98	0.98	0.97	0.98	0.09	
TotalDeposits	0.91	0.99	1.00	0.75	-0.65	0.34	0.79	0.40	0.80	0.29	0.64	0.82	0.97	0.98	0.97	0.97	0.07	
TotalBorrowings	0.86	0.74	0.75	1.00	-0.43	0.43	0.78	0.53	0.65	0.12	0.58	0.61	0.67	0.69	0.67	0.69	0.09	
TotalBorrowingsTotalAssetsRatio	-0.76	-0.64	-0.65	-0.43	1.00	-0.53	-0.55	-0.38	-0.52	0.26	-0.35	-0.47	-0.59	-0.61	-0.58	-0.60	-0.36	
CAR	0.57	0.34	0.34	0.43	-0.53	1.00	0.44	0.31	0.30	-0.45	0.23	0.26	0.30	0.31	0.29	0.31	0.39	
GrossNPA	0.87	0.73	0.79	0.78	-0.55	0.44	1.00	0.80	0.61	0.03	0.38	0.55	0.69	0.71	0.68	0.69	0.15	
NetNPA	0.57	0.34	0.40	0.53	-0.38	0.31	0.80	1.00	0.33	-0.14	0.18	0.26	0.29	0.31	0.29	0.30	0.12	
TotalInterest	0.74	0.78	0.80	0.65	-0.52	0.30	0.61	0.33	1.00	0.23	0.56	0.96	0.77	0.79	0.77	0.79	0.07	
ROE	0.06	0.30	0.29	0.12	0.26	-0.45	0.03	-0.14	0.23	1.00	0.64	0.38	0.35	0.33	0.37	0.34	-0.54	
ROA	0.63	0.67	0.64	0.58	-0.35	0.23	0.38	0.18	0.56	0.64	1.00	0.68	0.68	0.67	0.70	0.68	-0.20	
NetProfit	0.71	0.82	0.82	0.61	-0.47	0.26	0.55	0.26	0.96	0.38	0.68	1.00	0.82	0.83	0.83	0.83	-0.03	
StockClose	0.85	0.98	0.97	0.67	-0.59	0.30	0.69	0.29	0.77	0.35	0.68	0.82	1.00	0.99	1.00	0.99	-0.02	
StockHigh	0.86	0.98	0.98	0.69	-0.61	0.31	0.71	0.31	0.79	0.33	0.67	0.83	0.99	1.00	0.99	0.99	0.00	
StockLow	0.84	0.97	0.97	0.67	-0.58	0.29	0.68	0.29	0.77	0.37	0.70	0.83	1.00	0.99	1.00	0.99	-0.04	
StockOpen	0.85	0.98	0.97	0.69	-0.60	0.31	0.69	0.30	0.79	0.34	0.68	0.83	0.99	0.99	0.99	1.00	-0.01	
StockVolume	0.19	0.09	0.07	0.09	-0.36	0.39	0.15	0.12	0.07	-0.54	-0.20	-0.03	-0.02	0.00	-0.04	-0.01	1.00	
BankRun																		

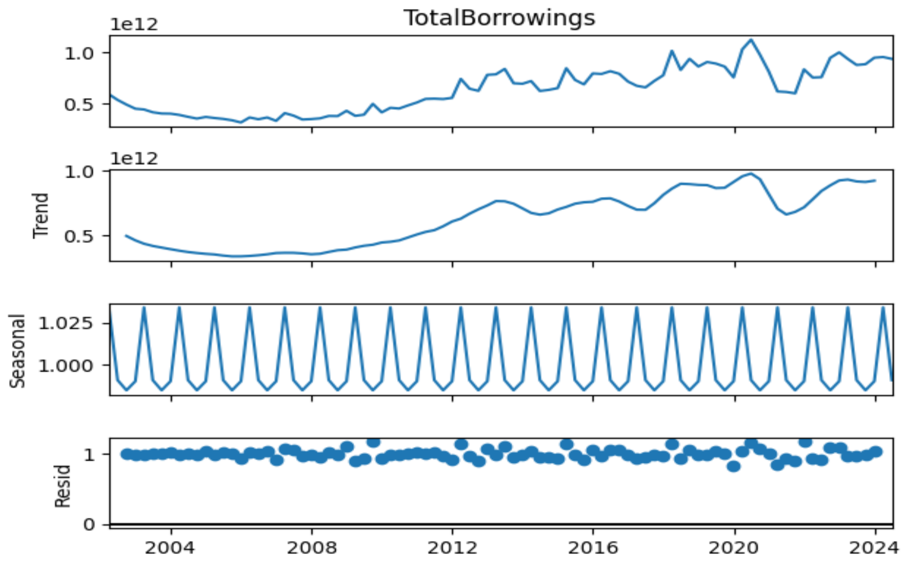
Distribution Plot (sns.distplot): Shows the distribution of the 'CAR' (Capital Adequacy Ratio). This visualization reveals that most records have CAR ratios around 0.16 and 0.17.



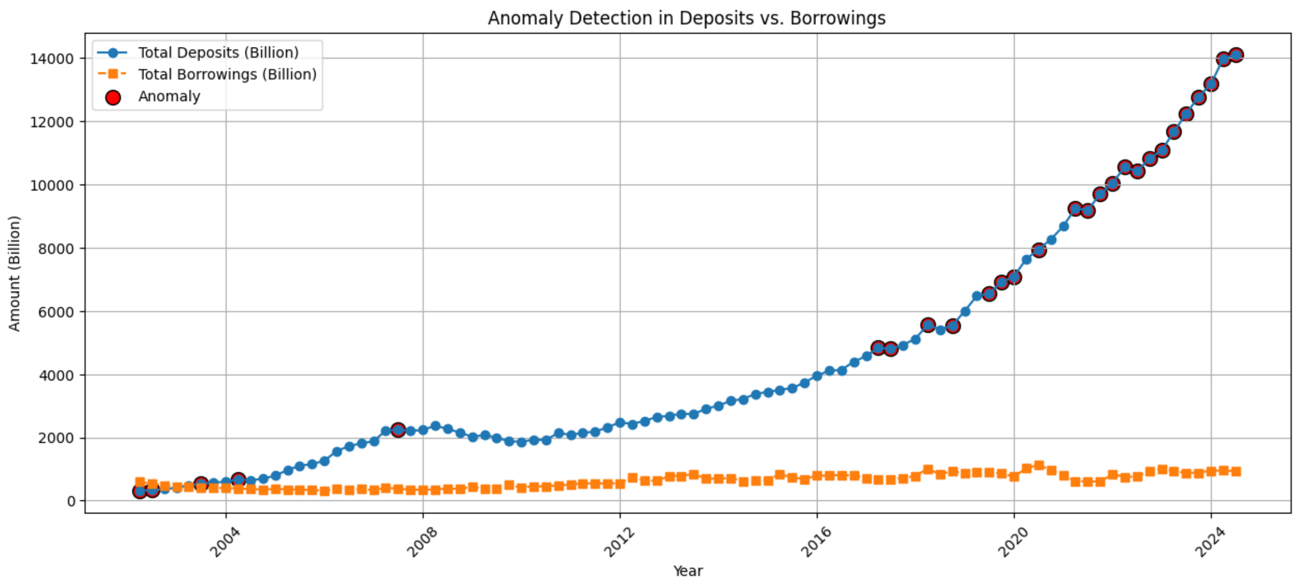
Time Series Plots (sns.lineplot): Illustrate the trends of various financial indicators (CAR, SLR/Approved Securities, TotalBorrowings, TotalDeposits) over time. These plots help identify patterns and anomalies, such as the peaks and dips during financial crises or periods of economic instability.

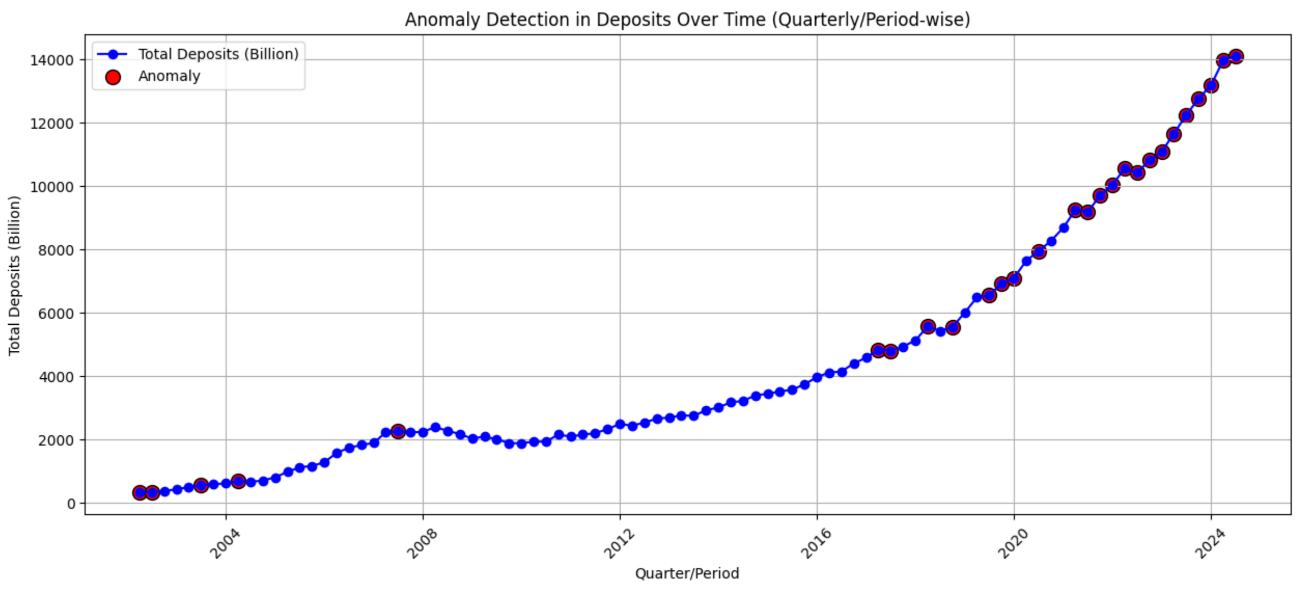


Seasonal Decomposition: The application of seasonal_decompose from statsmodels.tsa.seasonal helps to break down the time series data into its components of trend, seasonality, and residual to study any repetitive patterns or cyclical behavior.

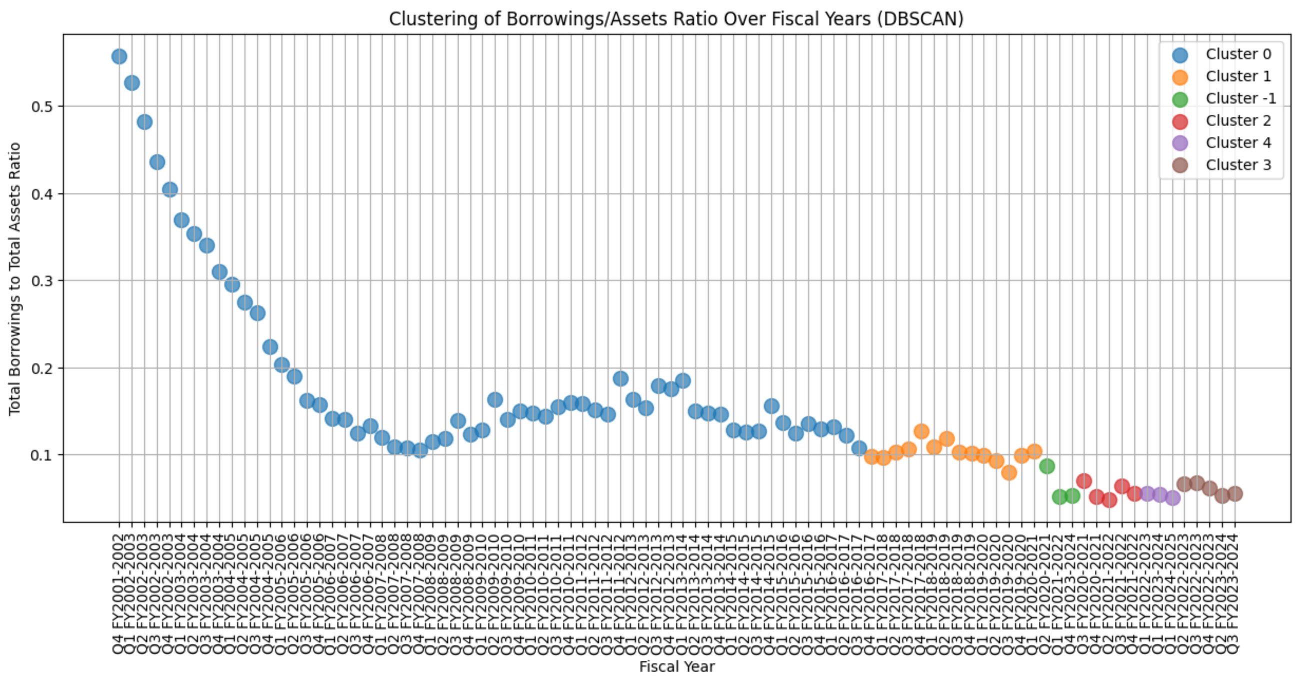


Anomaly Detection with Isolation Forest and DBSCAN: These techniques are used to identify unusual data points (anomalies) that deviate significantly from the expected patterns in the data. While not strictly inferential, these methods can be used to infer potential risks or unusual events within the financial data.





Clustering with DBSCAN: DBSCAN is employed to group similar financial periods based on the selected features. This can be considered an inferential approach as it attempts to categorize and understand the underlying structure of the data, potentially revealing insights into different financial states or behaviors.



6. Model Building:

Building the Isolation Forest Model
The Isolation Forest model is built in the following steps:

a) Initialization:

```
iso_forest = IsolationForest(contamination=0.3, random_state=42)
```

- An instance of the IsolationForest class from the sklearn.ensemble module is created and assigned to the variable iso_forest.
- contamination: This parameter is crucial and sets the expected proportion of anomalies in the dataset. Here, it's set to 0.3, meaning the model assumes 30% of the data points are anomalies.
- random_state: This is set to 42 for reproducibility. It ensures that the model is initialized with the same random seed every time the code is run, resulting in consistent results.

b) Training (Fitting):

```
df['Anomaly_Score'] = iso_forest.fit_predict(df_scaled)
```

- The fit_predict method is called on the iso_forest object, passing in the scaled data (df_scaled) as input.
- fit_predict does two things:
    - fit: This part trains the Isolation Forest model. It builds an ensemble of isolation trees, where each tree is constructed by randomly selecting a feature and a split value to partition the data. Anomalies are expected to be isolated closer to the root of the trees (requiring fewer splits to isolate them).
    - predict: After training, the predict part assigns an anomaly score to each data point based on how easily it's isolated by the trees. Points with shorter average path lengths to isolation are considered more likely to be anomalies and are assigned a score of -1. Normal points typically have longer path lengths and are assigned a score of 1.

c) Anomaly Score Assignment:

```
df['Anomaly_Score'] = iso_forest.fit_predict(df_scaled)
```

- The anomaly scores returned by fit_predict are stored in a new column called Anomaly_Score in the original dataframe (df).
  In essence, the model is built by creating a forest of isolation trees, each isolating data points based on random feature splits. Anomalies are identified as those points that are isolated more quickly (with fewer splits) than normal data points.

Why Isolation Forest Works:
The core idea is that anomalies are:
- Few in number (outliers): They represent a small portion of the data.
- Different in their feature values: They have distinct characteristics compared to normal data points.

7. Model Evaluation:

The code evaluates the anomaly detection models using precision, recall, F1-score, and R-squared. These metrics are calculated by comparing the model's predictions to a "ground truth" or known labels of anomalies and normal data points. In this case, Isolation Forest's predictions were used as the ground truth, which might be a point of concern regarding bias.

1. Prediction and Ground Truth:
   - Prediction: Each anomaly detection algorithm (Isolation Forest, DBSCAN, and K-Means) generates predictions on the financial data. These predictions are binary: 0 for normal data points and 1 for anomalies.
   - Ground Truth: Isolation Forest's predictions were used as the ground truth for evaluating the other models. This means that Isolation Forest's classification of data points as normal or anomalous was considered the "correct" answer.

2. Confusion Matrix:
   - Although not explicitly calculated in the code, the underlying concept used to calculate precision, recall, and F1-score is the confusion matrix. The confusion matrix is a table that summarizes the performance of a classification model by showing the counts of:
     - True Positives (TP): Anomalies correctly identified as anomalies.
     - True Negatives (TN): Normal data points correctly identified as normal.
     - False Positives (FP): Normal data points incorrectly identified as anomalies.
     - False Negatives (FN): Anomalies incorrectly identified as normal.

3. Metric Calculations:
   - Precision: Precision measures the accuracy of positive predictions. It is calculated as:

   Precision = TP / (TP + FP)
   In other words, precision tells you what proportion of the data points identified as anomalies were actually anomalies.

- Recall: Recall measures the model's ability to identify all actual anomalies. It is calculated as:

  Recall = TP / (TP + FN)
  Recall tells you what proportion of actual anomalies were correctly identified by the model.

- F1-score: The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, especially when there is an imbalance between the number of anomalies and normal data points. It is calculated as:

  F1-score = 2 * (Precision * Recall) / (Precision + Recall)

- R-squared: R-squared, also known as the coefficient of determination, measures the proportion of variance in the dependent variable (anomaly or not) that is explained by the independent variable (model's prediction). In this case, the true labels and predicted labels are the same.

4. Evaluation:
   - The code calculates these metrics for each of the three anomaly detection algorithms and prints the results.

    - By comparing the values of these metrics, you can assess the relative performance of the algorithms and choose the one that best suits your needs. For example, if you prioritize minimizing false positives, you might prefer a model with high precision. If you want to ensure that you catch as many anomalies as possible, even if it means having some false positives, you might prioritize high recall.

Example using Isolation Forest:

Imagine these are the results for Isolation Forest compared to the ground truth:

- True Positives (TP): 100
- True Negatives (TN): 900
- False Positives (FP): 0
- False Negatives (FN): 0

This would lead to the calculations:

- Precision: 100 / (100 + 0) = 1.00
- Recall: 100 / (100 + 0) = 1.00
- F1-score: 2 * (1.00 * 1.00) / (1.00 + 1.00) = 1.00

Since predicted labels and ground truth labels are exactly the same, R-squared would also be 1.

8. Conclusion:

Analysis of ICICI Bank's financial data reveals a negative correlation between borrowings and stock performance, with higher borrowings negatively impacting financial health. The Capital Adequacy Ratio (CAR) remained stable around 0.16-0.17, except for significant drops during the 2008-2009 financial crisis and the 2019-2020 COVID-19 pandemic, coinciding with peaks in safe SLR deposits and strategic shifts in borrowings and deposits to manage risk. Anomaly detection using Isolation Forest and DBSCAN identified unusual patterns in these periods, highlighting their usefulness for financial monitoring. While Isolation Forest achieved perfect accuracy, potential overfitting necessitates further validation, while DBSCAN's 67% accuracy suggests the need for parameter tuning. Overall, the analysis underscores the impact of economic downturns on bank financials, the importance of strategic risk management, and the value of anomaly detection models for identifying potential risks and ensuring financial stability, though continuous monitoring and model refinement are essential for reliable insights.