OPTIMISING ACCURACY IN CREDIT RISK PREDICTION

Evaluating Supervised and Unsupervised Learning Techniques for Credit Classification and Credit Scoring

Group 9:
Zhong Zhu Chen,
He James,
Lee Melisa,
Liu Mingcheng,
Syarwina Ridwan,
Nur Diyanah Binte Hasan Malik

01

INTRODUCTION

You can enter a subtitle
here if you need it

Optimising Accuracy in Credit Risk Prediction

Evaluating Supervised and Unsupervised Learning Techniques for Credit Classification and Credit Scoring

Dataset Description

- Credit risk crucial in loan approvals, interest rates
- Dataset from Kaggle with real-world application data, delinquency records
- Multiple features (education, income, etc.)

Motivation

- 2008 Financial Crisis linked to poor credit risk assessment
- Early identification of high-risk applicants crucial
- Reduce defaults, improve financial decisions

02

EXPLORATORY DATA ANALYSIS

You can enter a subtitle here if you need it

Credit Data – Features

Original Features:

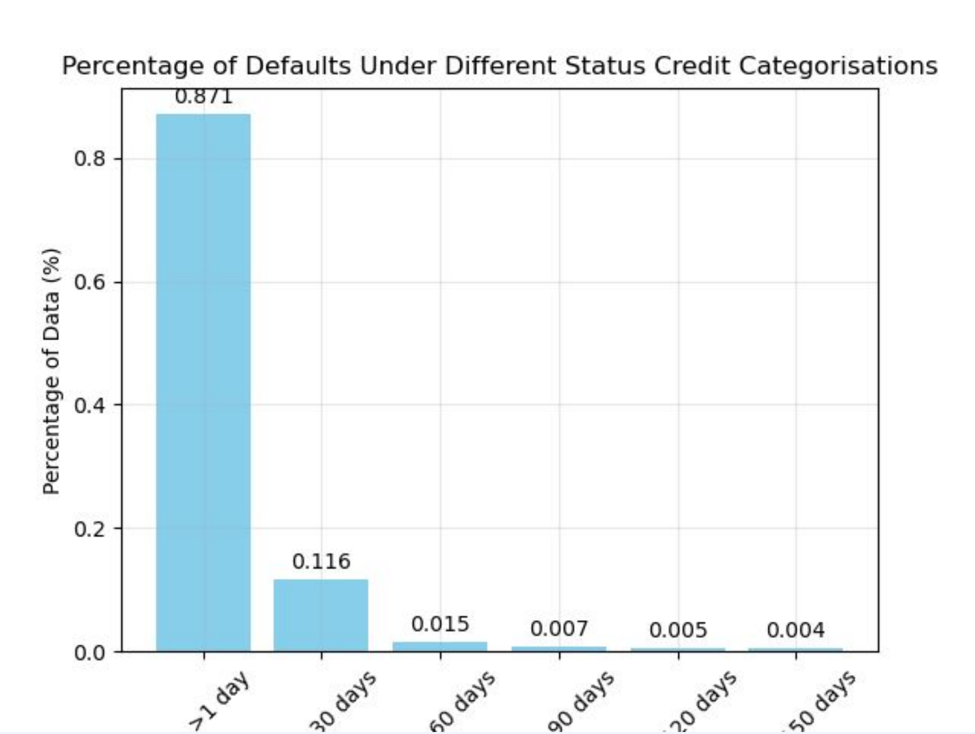
- ID: Identification for each Customer
- Months_Balance: Month data was recorded on
- Status: Degree of credit delinquency

Feature Engineering:

- open_month: The month when credit account was opened
- end_month: The latest month or month when credit account was closed
- window: The total number of months the credit account was active for

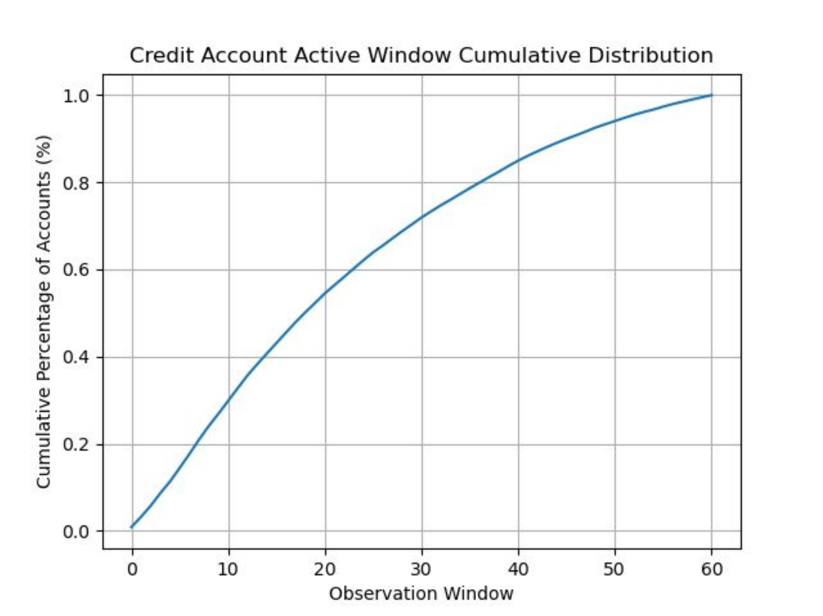
Credit Data – Credit Definition

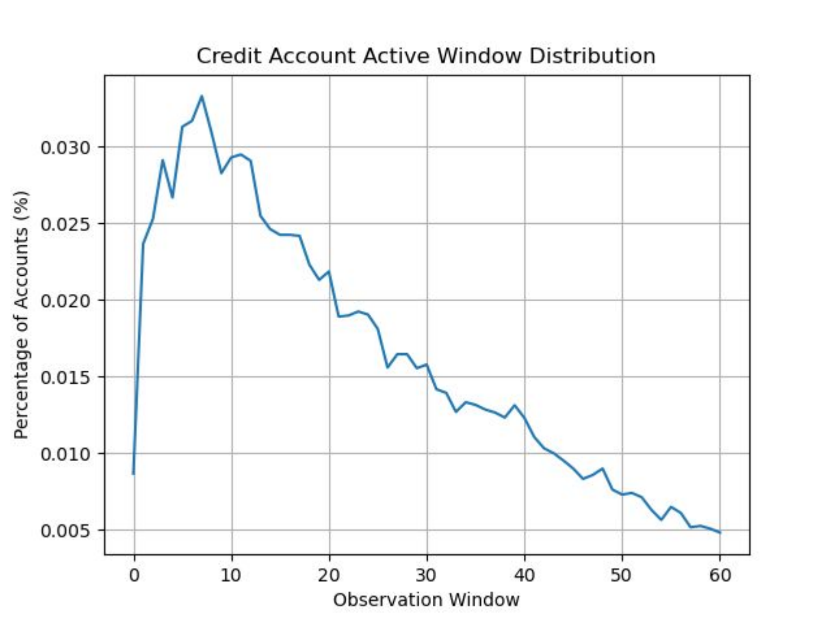
- Original Credit Status (8 different categorisations):
  - **`0`**: 1-29 days past due
  - **`1`**: 30-59 days past due
  - **`2`**: 60-89 days overdue
  - **`3`**: 90-119 days overdue
  - **`4`**: 120-149 days overdue
  - **`5`**: >150 days overdue/bad debts/write-offs
  - **`C`**: Paid off that month
  - **`X`**: No loan for the month
- Simplify credit categorisation into ‘Good’ or ’Bad’ and find ideal definition of ‘Bad Credit’
- Final ‘Bad Credit’ definition
  - **Credit Status:** [2, 3, 4, 5]
  - **Distribution:** 1.5% of Dataset (vs 2.54% of FRED last 10Y average)



Credit Data – Observation Window Testing

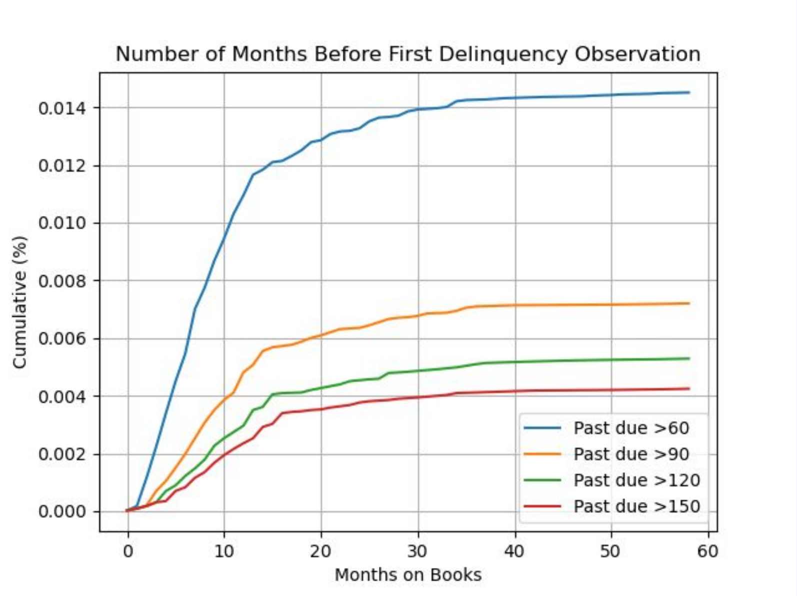
- Definition: Number of months the credit card account was active for
- Motivation: If the active window is too short, the model may lack sufficient reliability in in determining credit risk

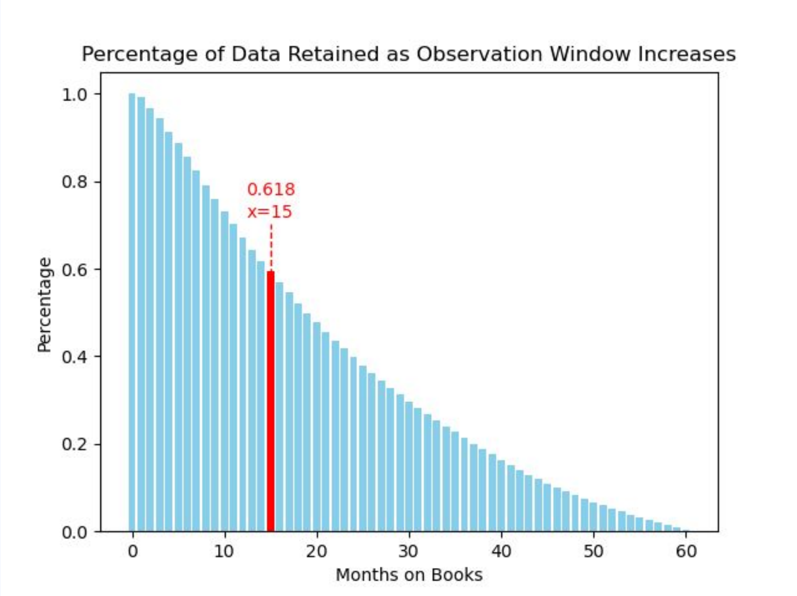




Credit Data – Observation Window Testing

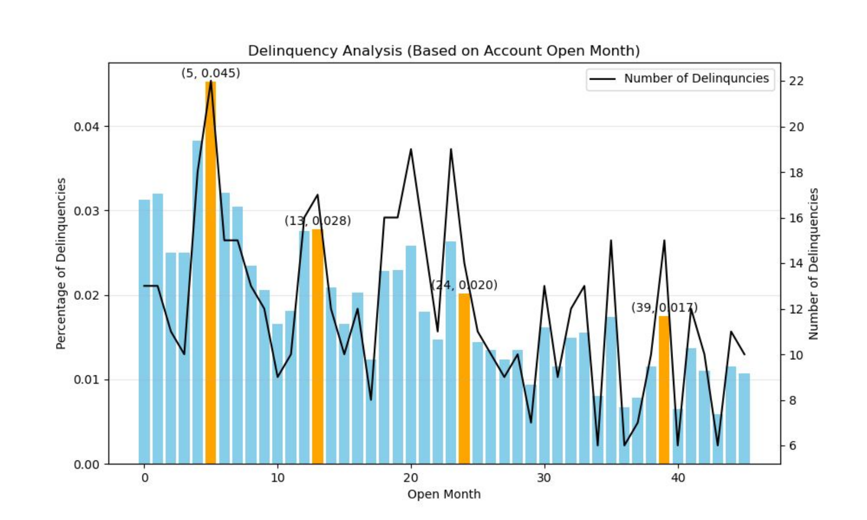
- Observe the number of months it took to observe the customer’s first delinquency.
- Identify window period where most customers would have revealed that they are ‘Bad Credit’ (Recall: ‘Bad Credit’ defined as customers with delinquencies >60 days’)

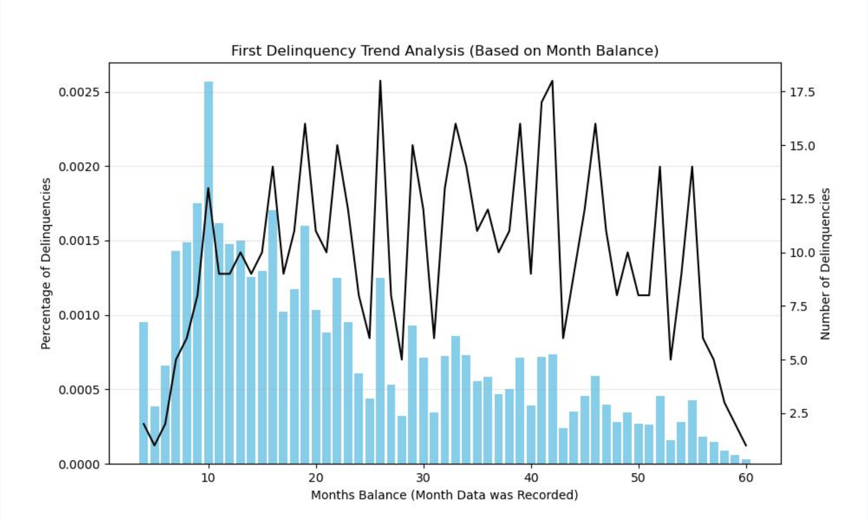




Credit Data – Vintage Analysis

- Analyse temporal/seasonal nature of cohort-based credit defaults based on <u>account opening month (LHS)</u> or based on <u>month data was recorded (LHS)</u>
- RHS Plot: Regular peaks observed every few months – indicates the potential usefulness of open_month feature
- LHS Plot: Decreasing trend in percentage or delinquency but relatively constant absolute delinquency numbers – indicates possible redundancy of months_balance feature





Applications Data – Feature Engineering

- Derived age, employment duration, income per family member
- Created employment-age ratio to reflect stability

$$\text{AGE} = \left\lfloor \frac{-\text{DAYS\_BIRTH}}{365} \right\rfloor$$

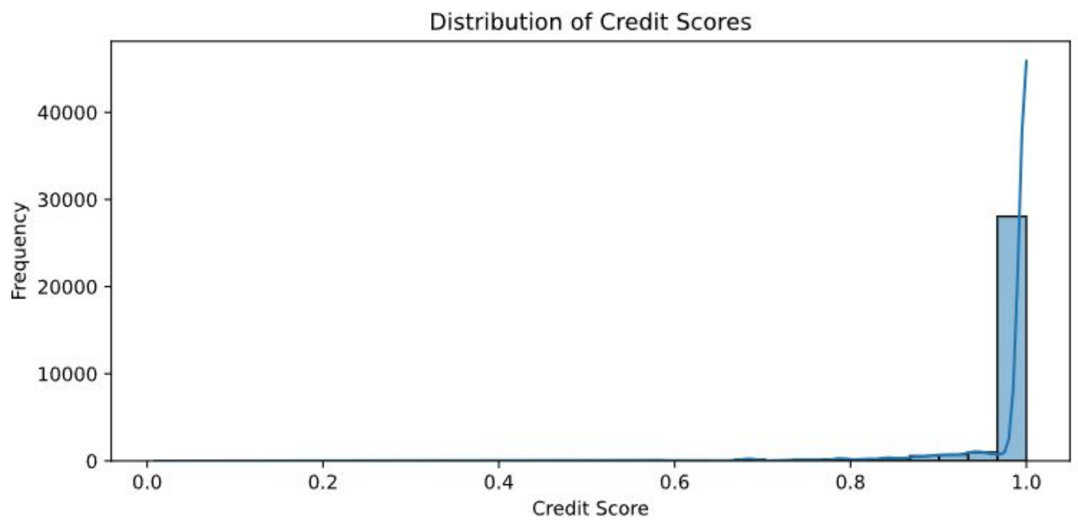
$$\text{employment\_age\_ratio} = \begin{cases} 0 & \text{if AGE} = 0 \\ \frac{\text{YEARS\_EMPLOYED}}{\text{AGE}} & \text{otherwise} \end{cases}$$

$$\text{YEARS\_EMPLOYED} = \begin{cases} 0 & \text{if DAYS\_EMPLOYED} \geq 365243 \\ \left\lfloor \frac{-\text{DAYS\_EMPLOYED}}{365} \right\rfloor & \text{otherwise} \end{cases}$$

$$\text{married} = \begin{cases} 1 & \text{if CNT\_FAM\_MEMBERS} - \text{CNT\_CHILDREN} - 1 = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{income\_per\_family\_member} = \frac{\text{AMT\_INCOME\_TOTAL}}{\text{CNT\_FAM\_MEMBERS}}$$

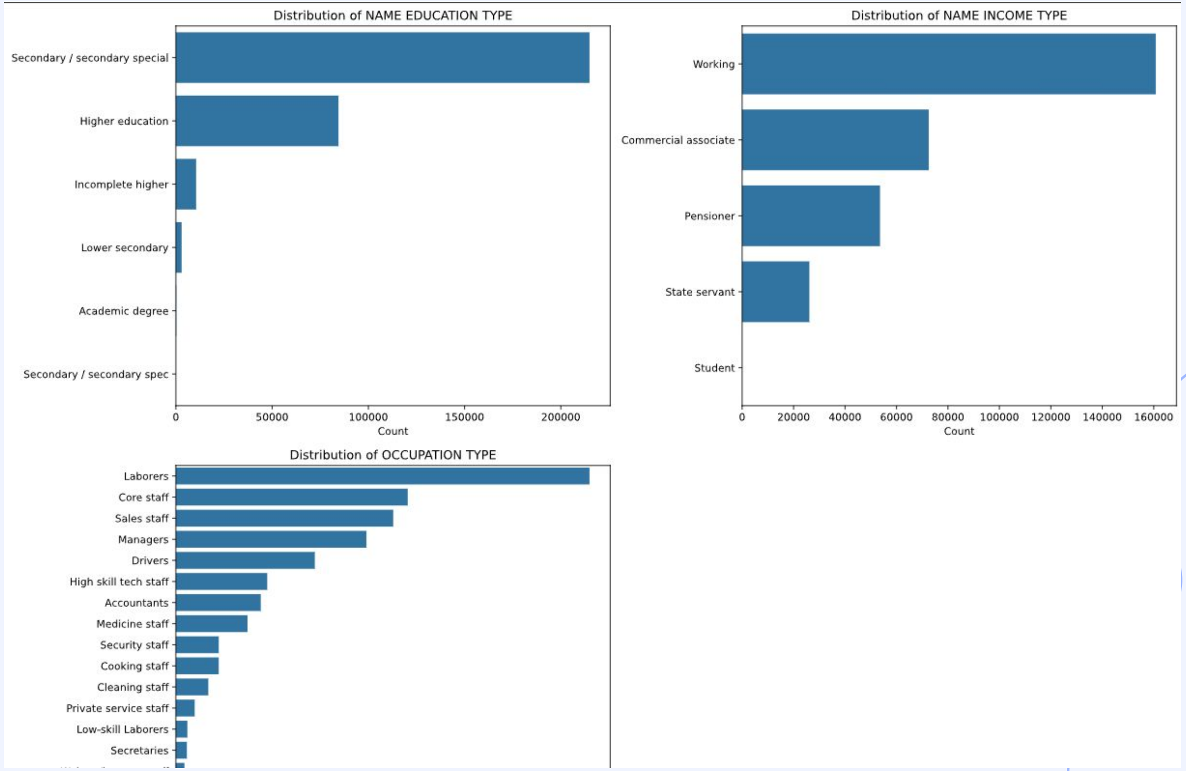
Applications Data – Credit Score Distribution



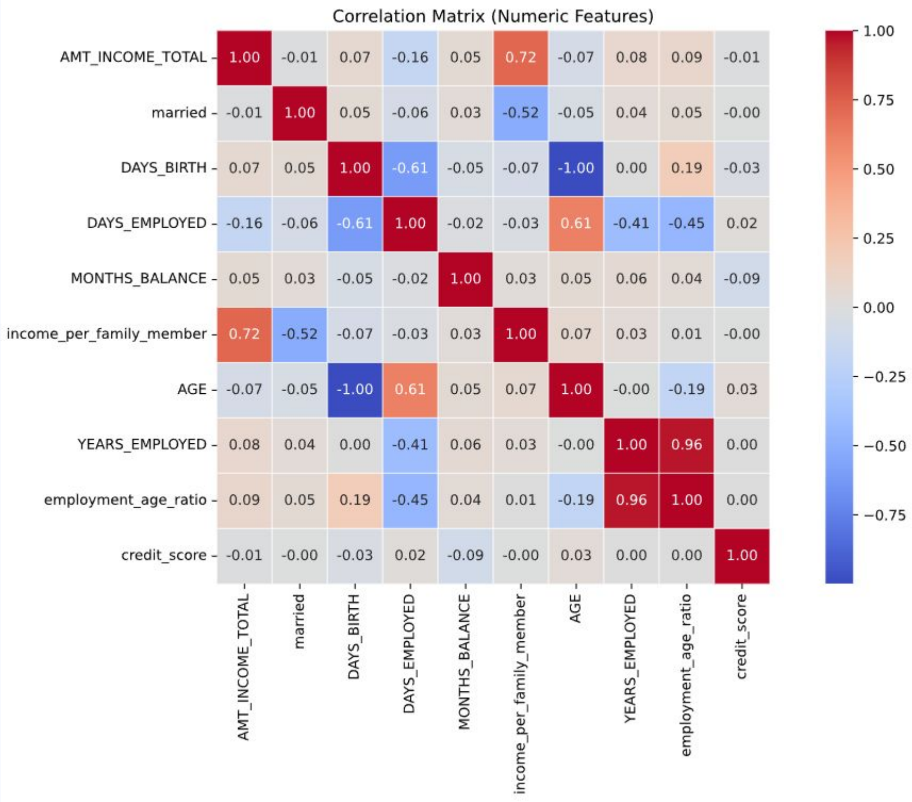
- Right-skewed: Most users scored 0.6–1.0 (few delinquencies)
- < 0.4 scores indicate high-risk applicants

Applications Data – Categorical Insights

- Majority: Secondary education, working individuals
- Top occupations: Sales, labor, core staff
- 31% with unknown occupation → imputed



Applications Data – Correlation Findings



- Positive: Age, years employed, employment-age ratio
- Weak: Income → not a strong standalone predictor

03

SUPERVISED LEARNING

You can enter a subtitle here if you need it

Data Preprocessing – Credit Scoring with Exponential Decay

- Credit score decreases exponentially with increasing severity of delinquency (STATUS).
- Higher scores are assigned to less severe delinquencies.
- Mean score per user is computed to summarize overall creditworthiness
- months is duration of financial history, serving as a proxy for credit exposure over time

$$\text{Credit Score} = e^{-\lambda \cdot \text{STATUS}}$$

$$\text{Mean Score}_{\text{user}} = \frac{1}{n} \sum_{i=1}^{n} e^{-\lambda \cdot \text{STATUS}_i}$$

$$\text{months} = \text{Number of months of credit history available}$$

Data Preprocessing – Dataset Merging + Cleaning & Encoding

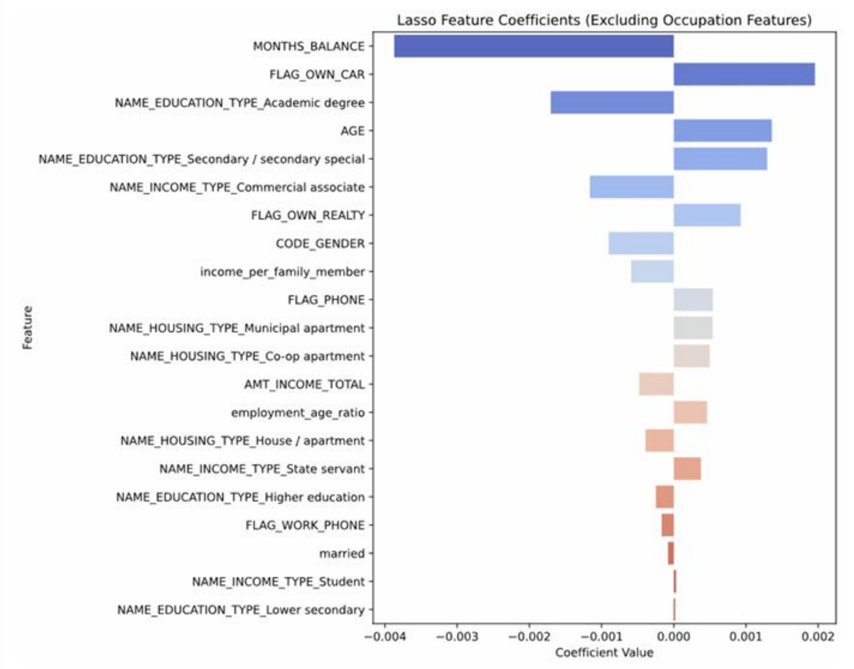
Dataset Merging:

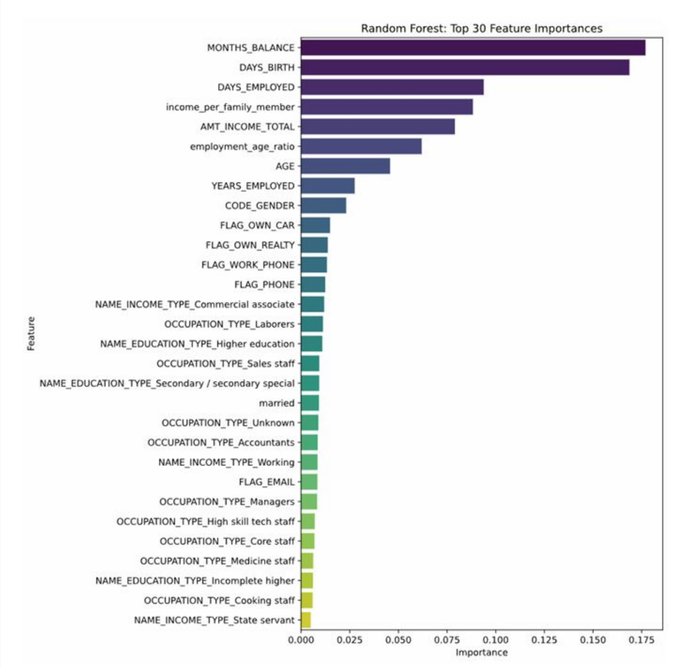
- Merged credit + application data → ~32K valid records
- Credit history window added (duration of records)

Cleaning & Encoding:

- Missing occupations imputed as "Unknown"
- Label encoding (binary), one-hot for multi-category features

Data Preprocessing – Feature Selection





Lasso regression selected 40+ important features

Random forest confirmed key predictors: age, income, credit history

Data Preprocessing – Dealing with Status ‘C’/’X’

When transforming STATUS to a credit score:
credit_score = exp(-STATUS),
STATUS ‘C’ and ‘X’ are viewed as the same as STATUS 0.

C/X status	smogn	RMSE	MAE	R2
0	0	0.0050	0.0300	0.1338
0	1	0.0166	0.0822	0.5425
1	0	0.0020	0.0176	0.1353
1	1	0.0091	0.0544	0.5436

Data Preprocessing – SMOGN

To handle the imbalance issue, we applied smogn to the dataset. (Synthetic Minority Over-Sampling Technique for Regression with Gaussian Noise)
Model performance with smogn data sees a distinct improvement in all RMSE, MAE, and R^2 scores.

C/X status	smogn	RMSE	MAE	R2
0	0	0.0050	0.0300	0.1338
0	1	0.0166	0.0822	0.5425
1	0	0.0020	0.0176	0.1353
1	1	0.0091	0.0544	0.5436

Evaluation metrics

1. $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$,--how well the model fits overall
2. $\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, less sensitive to outliers than RMSE
3. $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$ --how much of the variability is explained

Without smogn, all scores are low. ->Focus on R^2

C/X status	smogn	RMSE	MAE	R2
0	0	0.0050	0.0300	0.1338
0	1	0.0166	0.0822	0.5425
1	0	0.0020	0.0176	0.1353
1	1	0.0091	0.0544	0.5436

Logistic Regression & Lasso Regression – Motivation & Definitions

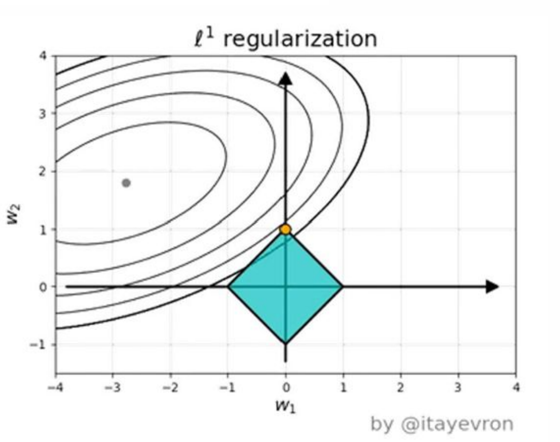
Logistic Regression

- ❖ Assumes a linear relationship between the independent variables (features) and the dependent variable (target)
- ❖ Struggles with complex, nonlinear relationships

Lasso Regression

- ❖ Linear regression with the addition of an L1 regularization term
- ❖ Penalizes the absolute values of the regression coefficients
- ❖ Reduces complexity and avoids overfitting

➢ To establish a baseline predictive performance



Logistic Regression & Lasso Regression – Model Training

Data

- ❖ Standardized with standard scalar
- ❖ 30-70 test train split

Logistic Regression

- ❖ LinearRegression() from sklearn

Lasso Regression

- ❖ LassoCV() with:
    - ➢ 5-fold internal cross-validation
    - ➢ max_iter=10000
    - ➢ random_state=42

Both cross validated with 5-fold K-Fold Cross-Validation

Logistic Regression & Lasso Regression – Residual Analysis

Homoscedasticity Check plots:

- ❖ Residuals scattered across the predicted values with no clear pattern.
  - ➢ The assumption of homoscedasticity (constant variance) holds.

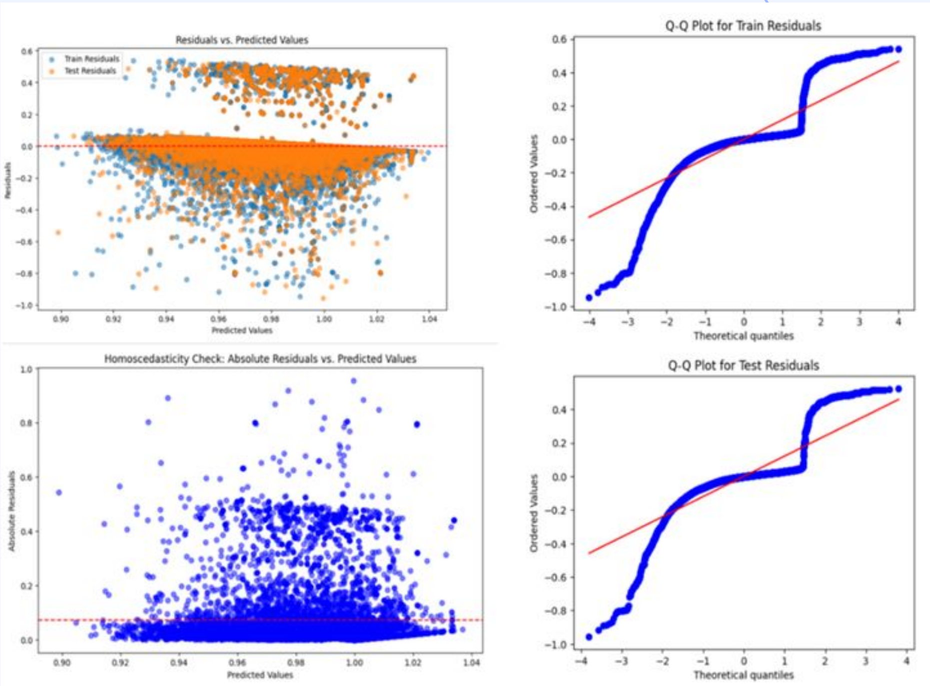


Figure 1 Residual analysis for Linear Regression model

Logistic Regression & Lasso Regression – Residual Analysis

Q-Q plots:

- ❖ Residuals greatly deviate from the red reference line.
  - ➢ Not normally distributed.
- ❖ Curve on the right side of the plot
  - ➢ Unaccounted complexities in the data, missing explanatory variables or non-linear patterns

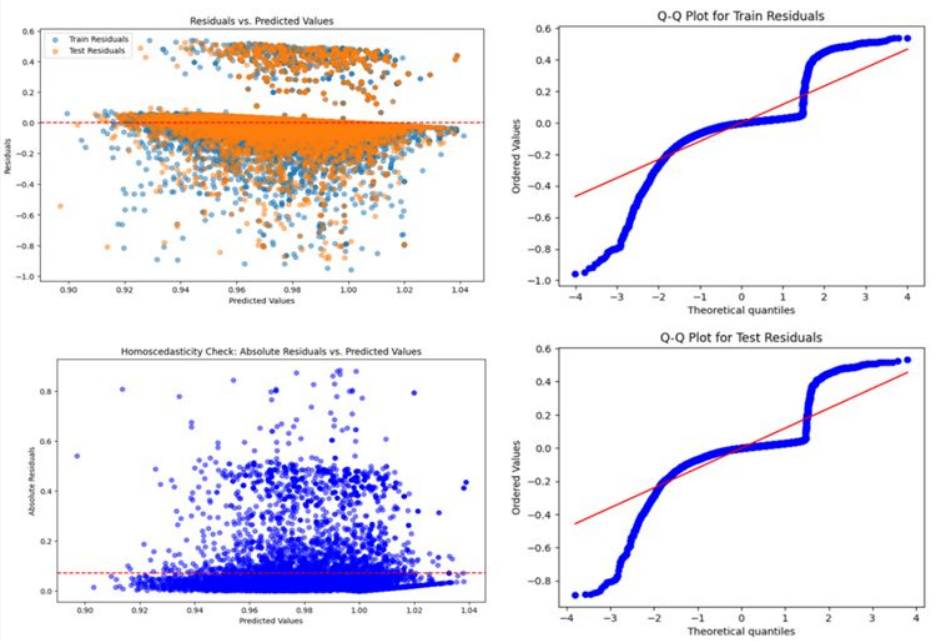


Figure 2 Residual analysis for Lasso model

Logistic Regression & Lasso Regression - Model Evaluation

- ❖ Consistent training and testing scores: no overfitting.
- ❖ Low R^2 : aren't capturing the datas' complexity.
  - ➢ Lack the ability to capture non-linear relationships in the data.

- ❖ An even lower R^2 for Lasso CV : might be penalizing important features too aggressively

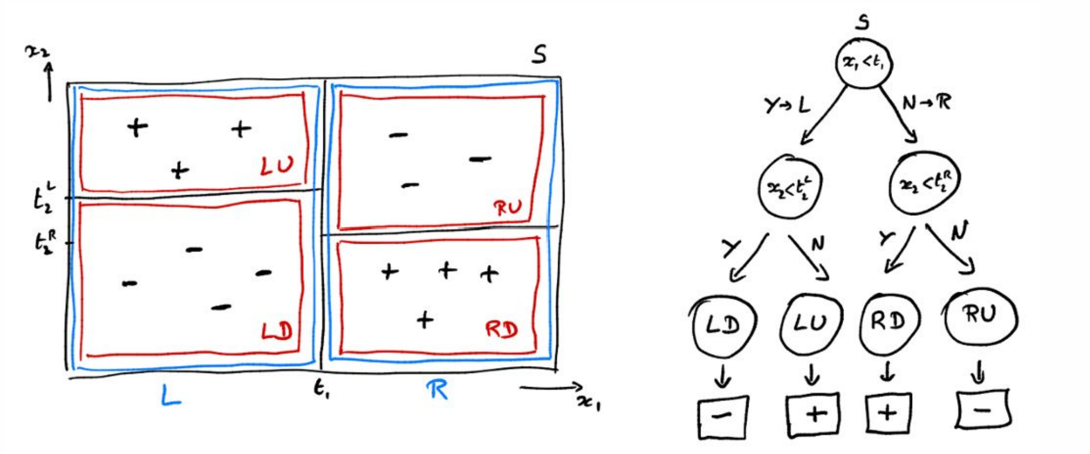
	Train			Test			CV			
Model\metrics	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	Note
Linear regression	0.1403	0.0711	0.0190	0.1391	0.0696	0.0189	0.1404	0.0712	0.0166	Underfit
Lasso	0.1403	0.0710	0.0190	0.1391	0.0696	0.0188	0.1416	0.0724	0.0002	Underfit

Decision Tree – Motivation & Definitions

Decision Tree

- ❖ Split the data into subsets based on feature values
- ❖ Each node represents a decision rule, and leaves represent predictions
- ❖ Intuitive and flexible

- ❖ Can overfit if not properly pruned or regulated



Decision Tree – Model Training

- ❖ Trained with DecisionTreeRegressor()

- ❖ Random State 42
- ❖ RandomizedSearchCV 5-fold cross validation over
  - ➢ max_depth: [5, 10, 15, 20, None],
  - ➢ min_samples_split: [2, 5, 10],
  - ➢ min_samples_leaf: [1, 2, 4],
  - ➢ max_features: ['sqrt', 'log2', None]

- ❖ Optimized separately for: RMSE, MAE, and R^2

- ❖ Selected best parameters for evaluation

Decision Tree – Tuning with Cross Validation

- High R^2 value
- Lower R^2 value (0.3452)
- Disparity between training and testing performance

- Results from Randomized Search CV indicate limiting tree depth and increasing samples per leaf can't provide much improvement in generalization.

Decision Tree – Randomized Search CV with a different data split				
Results\Optimizing Goal	Test	CV-RMSE	CV-MAE	CV- R^2
RMSE	0.1162	0.1139	0.116	0.1147
MAE	0.0450	0.0428	0.0514	0.0525
R^2	0.3452	0.3713	0.3473	0.3619
min_samples_split	2	2	5	5
min_samples_leaf	1	1	2	1
max_features	None	log2	log2	log2
max_depth	None	None	20	20
Note		Best overall		Good fit

Decision Tree – Model Evaluation

Without Hyperparameter Tuning:

- ❖ Perform exceptionally well on the training set
  - ➢ High R^2 value: captures most of the variance in the training set.
- ❖ A significant drop in performance on testing:
  - ➢ Lower R^2 value: struggles to generalize on unseen data.
    - ■ Overfitting

	Decision Tree				
Metrics	train	test	CV-RMSE	CV-MAE	CV- R^2
RMSE	0.0309	0.118	0.1074	0.1127	0.1074
MAE	0.0047	0.0455	0.0401	0.0537	0.0401
R^2	0.9524	0.2948	0.4149	0.3563	0.4149
Note	Overfit		Good fit		Good fit

Decision Tree – Model Evaluation

With Hyperparameter Tuning:

- ❖ Strategies like limiting tree depth and increasing the minimum samples per leaf can help reduce overfitting.
- ❖ Slight improvement: Other measures were needed for generalization.

	Decision Tree				
Metrics	train	test	CV-RMSE	CV-MAE	CV- R^2
RMSE	0.0309	0.118	0.1074	0.1127	0.1074
MAE	0.0047	0.0455	0.0401	0.0537	0.0401
R^2	0.9524	0.2948	0.4149	0.3563	0.4149
Note	Overfit		Good fit		Good fit

Decision Tree – Residual Analysis

- ❖ A wider spread for the test residuals (orange)
- ❖ Q-Q plot for the train set fits much better than the test one
  - ➢ Overfit
- ❖ Q-Q plots have Deviations at the tail:
  - ➢ more extreme values than expected under normality.

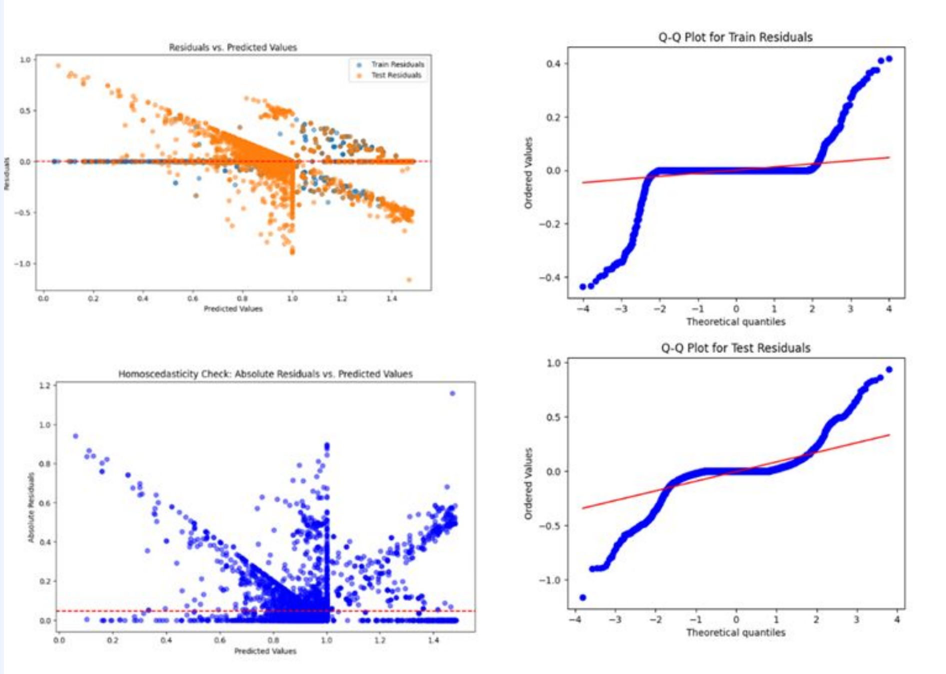


Figure 3 Residual analysis for Decision Tree↵

Decision Tree – Residual Analysis

Homoscedasticity Check plots:

- ❖ A noticeable pattern where the residual variance changes as the predicted values increase
  - ➢ No uniform variance (homoscedasticity)

Decision Tree and Random Forest models do not rely on the assumption of homoscedasticity.

In fact, it can indicate that the models are capturing non-linear relationships or complex patterns within the dataset.

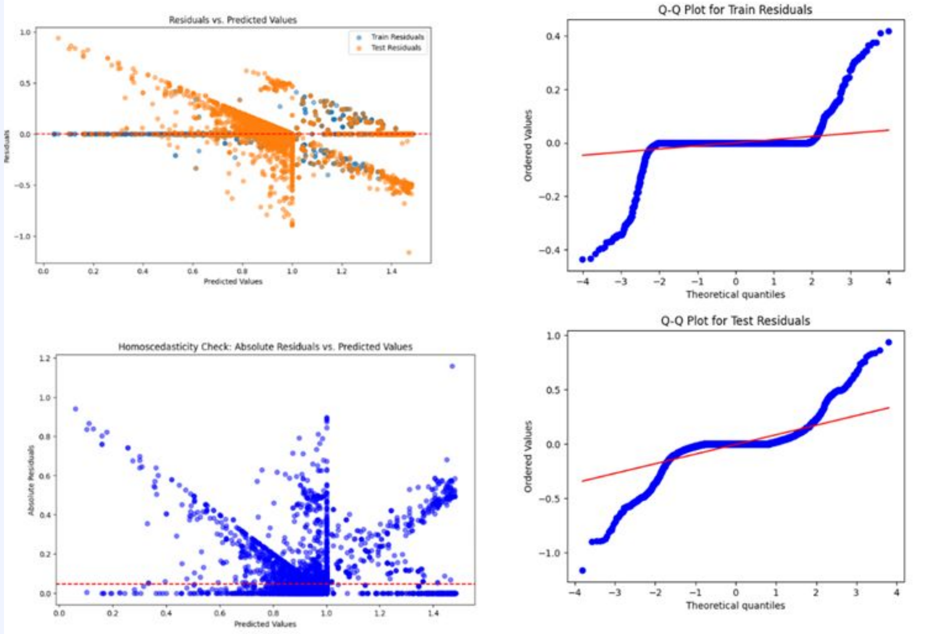
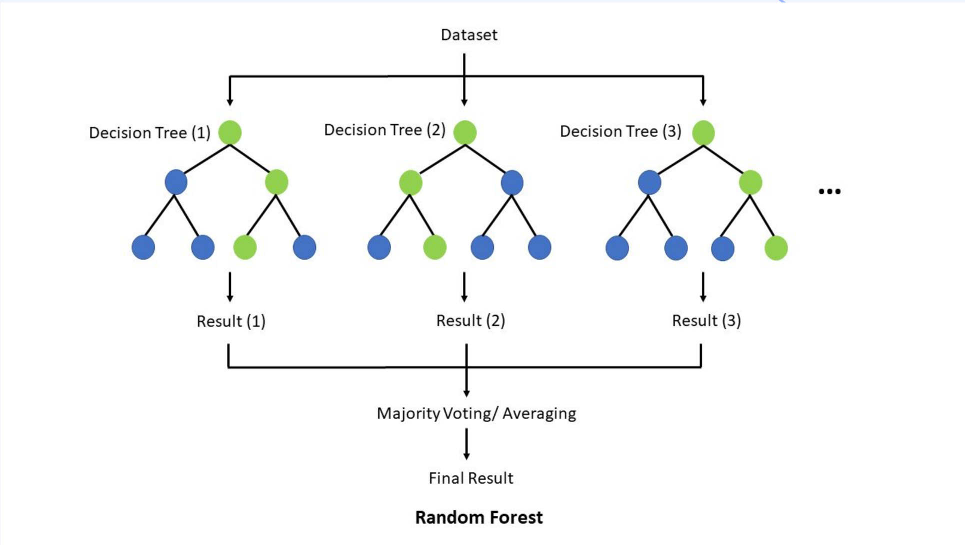


Figure 3 Residual analysis for Decision Tree

Random Forest – Motivation & Definitions

Random Forest

- ❖ An ensemble model that builds multiple decision trees and combines their predictions.
- ❖ Reduce overfitting by aggregating results.
- ❖ Perform well with complex, high-dimensional data.



Random Forest – Model Training

- Trained with RandomForestRegressor()

- Random State 42
- RandomizedSearchCV 5-fold cross validation over
  - max_depth: [5, 10, 15, 20, None],
  - min_samples_split: [2, 5, 10],
  - min_samples_leaf: [1, 2, 4],
  - max_features: ['sqrt', 'log2', None]

- Optimized separately for: RMSE, MAE, and R^2

- Selected best parameters for evaluation

Random Forest – Model Evaluation

- ❖ Higher R^2 value in testing:
  - ➢ better generalization to unseen data compared with Decision Tree.
- ❖ Difference in performance between training and testing sets:
  - ➢ Still overfitting

	Random Forest				
Metrics	train	test	CV-RMSE	CV-MAE	CV- R^2
RMSE	0.0423	0.0856	0.0917	0.0914	0.0956
MAE	0.0179	0.0409	0.0474	0.0473	0.0497
R^2	0.911	0.6286	0.5738	0.5762	0.5372
Note	Best overall			Good fit	

Random Forest – Model Evaluation

- Minor improvements with tuning:
  - Reasonably robust
  - A potential plateau in performance gains, might require further regularization or feature selection

	Random Forest				
Metrics	train	test	CV-RMSE	CV-MAE	CV- R^2
RMSE	0.0423	0.0856	0.0917	0.0914	0.0956
MAE	0.0179	0.0409	0.0474	0.0473	0.0497
R^2	0.911	0.6286	0.5738	0.5762	0.5372
Note	Best overall			Good fit	

Random Forest – Tuning with Cross Validation

- Training results are slightly worse than the Decision Tree
- R^2 value in testing demonstrate generalization is stronger
- Disparity between training and testing sets
- Potential Plateau for performance gains

Random Forest - Randomized Search CV with a different data split				
Results\Optimizing Goal	Test	CV-RMSE	CV-MAE	CV- R^2
RMSE	0.089	0.0941	0.0943	0.0924
MAE	0.0416	0.0478	0.0481	0.0451
R^2	0.616	0.5707	0.5691	0.5861
min_samples_split	2	10	5	5
min_samples_leaf	1	1	2	2
max_features	None	None	sqrt	None
max_depth	None	None	None	None
Note	Best overall			Good fit

Random Forest – Residual Analysis

- ❖ Smaller residuals than Decision Tree:
  - ➢ better performance

Q-Q plot:

- ❖ Deviations at the tail:
  - ➢ more extreme values than expected under normality.
- ❖ Heavy-tailed residuals:
  - ➢ outliers or data complexity.

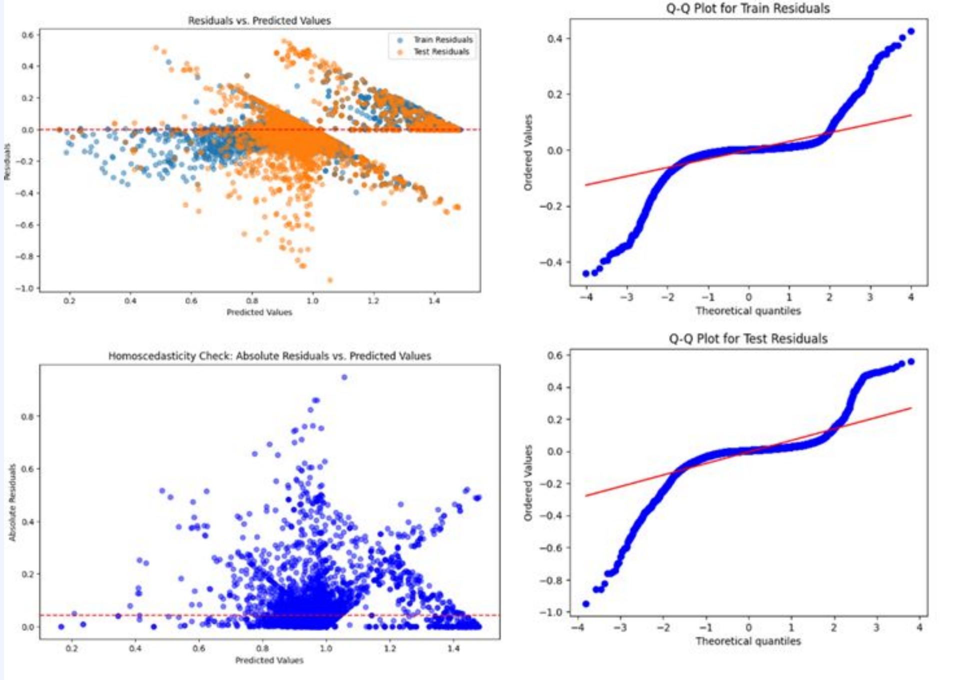
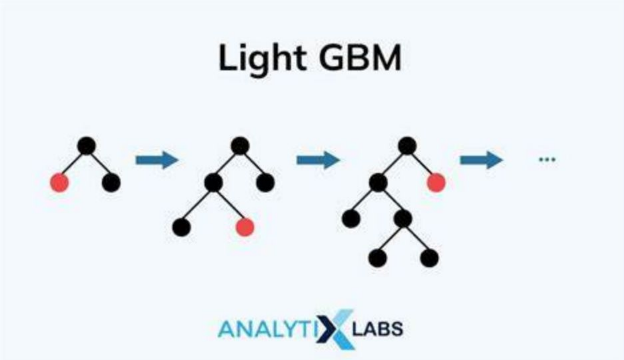


Figure 4 Residual analysis for Random Forest

LightGBM – Motivation & Definitions

- ❖ A gradient boosting model
- ❖ Create decision tree-based ensembles
- ❖ Features leaf-wise growth and flexibility with hyperparameters

- ❖ Designed for speed and scalability
- ❖ Capable of handling large datasets with many features while providing accurate predictions



LightGBM – Training and Tuning

- The hyperparameter tuning was done through GridSearchCV on the grids shown in the table.
- The grid was adjusted according to the best parameters found in the first grid.

params	adjusted grid	previous grid
learning rate	[0.35, 0.4, 0.45]	[0.05, 0.1, 0.2, 0.3, 0.4, 0.5]
num_of_leaves	[550,575,600,625,650,675]	[255,300,400,511,600,700]
data in leaf	[5, 10, 20, 50]	[5, 10, 20, 50]
feature fraction	[0.8, 1.0]	[0.6, 0.8, 1.0]
bagging fraction	[0.8, 1.0]	[0.6, 0.8, 1.0]
bagging freq	[5,10]	[5,10]
max_depth	[-1, 5]	[-1, 5, 10]
boosting type	['dart']	['gbdt', 'rf', 'dart']

LightGBM – Training and Tuning

- Overall, the best parameters are as indicated in the 4th column.

- The parameters in the 5th column differs from the 4th in lr only
  - Minor difference in performance

LightGBM-model evaluation				
params	learning rate	0.4	0.4	0.35
	num_of_leaves	600	600	600
	data in leaf	20	5	5
	feature fraction	1	0.8	0.8
	bagging fraction	1	1	1
	bagging freq	5	5	5
	max_depth	-1	-1	-1
	boosting type:	dart	dart	dart
CV	R^2	0.5886	0.6060	0.6221
Train	RMSE	0.0009	0.0009	0.0009
	MAE	0.0059	0.0054	0.0056
	R^2	0.9536	0.9538	0.9538
Test	RMSE	0.0072	0.0068	0.0068
	MAE	0.0412	0.0375	0.0375
	R^2	0.6375	0.6595	0.6585
Note			Best overall	Good fit

LightGBM – Residual Analysis

- ❖ Close-to-zero residual means on both sets:
  - ➢ making predictions with minimal bias.
- ❖ The relatively low variance on the training set:
  - ➢ consistently accurate on the training data.
- ❖ Notably higher variance on the testing set:
  - ➢ predictions on unseen data are less consistent
  - ➢ slight overfitting.

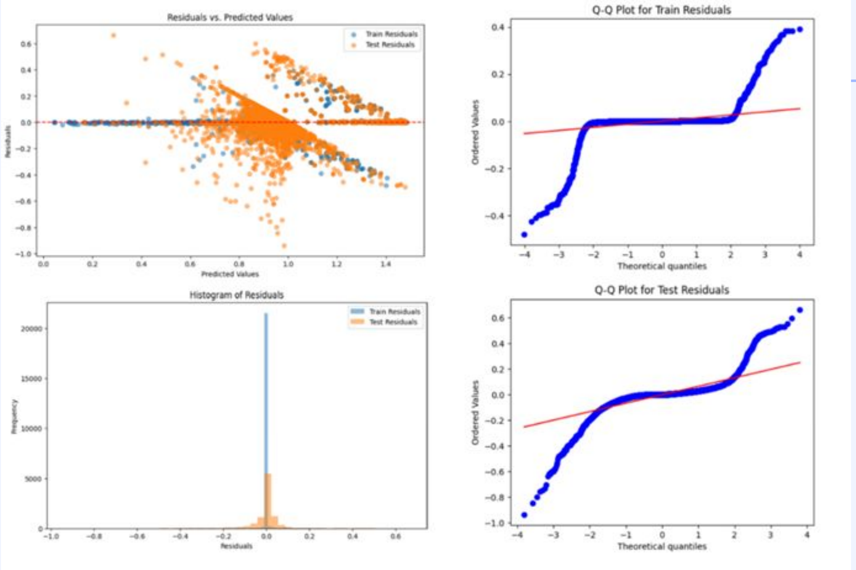


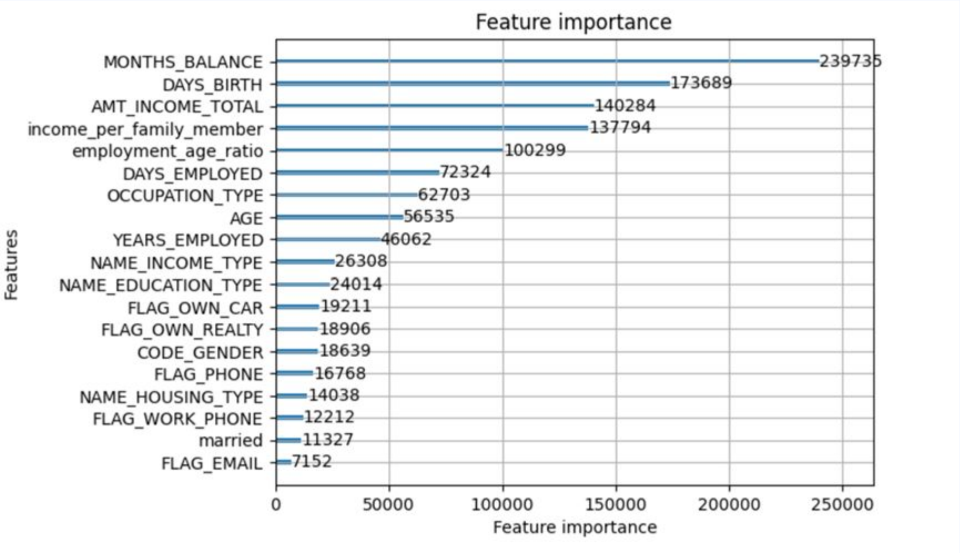
Figure 5 Residual analysis for LightGBM

Train Mean	0.042%	Train Variance	0.092%
Test Mean	-0.094%	Test Variance	0.675%

LightGBM – Feature Importance Analysis

- Made better use of features compared with Lasso

- Top features: MONTHS_BALANCE, DAYS_BIRTH, DAYS_EMPLOYED and income features,
  - in accordance with that of the Random Forest model.



LightGBM – Feature Importance Analysis

- Train the model on a dataset with selected feature: The performance is slightly worse.
  - Features all contain more or less useful information
  - Contribution may be negligible when considering the cost

	metrics	scaled	scaled +top18	scaled +top9
Train	RMSE	0.0009	0.0009	0.0009
	MAE	0.0054	0.0055	0.0057
	R^2	0.9538	0.9538	0.9527
Test	RMSE	0.0068	0.0068	0.0073
	MAE	0.0375	0.0377	0.0398
	R^2	0.6573	0.6568	0.6333

LightGBM – Model Evaluation

- Better than the Random Forest model:
  - Much Better RMSE and MAE on the train set.
  - Much Better RSME on the test set and slightly improved MAE.
  - Higher R^2 on the train & test set

LightGBM-model evaluation				
params	learning rate	0.4	0.4	0.35
	num_of_leaves	600	600	600
	data in leaf	20	5	5
	feature fraction	1	0.8	0.8
	bagging fraction	1	1	1
	bagging freq	5	5	5
	max_depth	-1	-1	-1
	boosting type:	dart	dart	dart
CV	R^2	0.5886	0.6060	0.6221
Train	RMSE	0.0009	0.0009	0.0009
	MAE	0.0059	0.0054	0.0056
	R^2	0.9536	0.9538	0.9538
Test	RMSE	0.0072	0.0068	0.0068
	MAE	0.0412	0.0375	0.0375
	R^2	0.6375	0.6595	0.6585
Note			Best overall	Good fit

LightGBM – Model Evaluation

'DART'-Dropouts meet Multiple Additive Regression Trees.

- ❖ Address Over-specialization:
  - ➢ trees added later in boosting focus too narrowly on specific instances
- ❖ Randomly drop trees during training
  - ➢ Encourages the model to rely on a broader set of trees,
  - ➢ improve generalization and reduce overfitting.

LightGBM-model evaluation				
params	learning rate	0.4	0.4	0.35
	num_of_leaves	600	600	600
	data in leaf	20	5	5
	feature fraction	1	0.8	0.8
	bagging fraction	1	1	1
	bagging freq	5	5	5
	max_depth	-1	-1	-1
	boosting type:	dart	dart	dart
CV	R^2	0.5886	0.6060	0.6221
Train	RMSE	0.0009	0.0009	0.0009
	MAE	0.0059	0.0054	0.0056
	R^2	0.9536	0.9538	0.9538
Test	RMSE	0.0072	0.0068	0.0068
	MAE	0.0412	0.0375	0.0375
	R^2	0.6375	0.6595	0.6585
Note			Best overall	Good fit

LightGBM – Model Evaluation

- Clear difference between training and testing sets:
  - the overfitting issue still exists.
- Improvement for hyperparameter tuning is negligible
  - a potential plateau in performance gains.

LightGBM-model evaluation				
params	learning rate	0.4	0.4	0.35
	num_of_leaves	600	600	600
	data in leaf	20	5	5
	feature fraction	1	0.8	0.8
	bagging fraction	1	1	1
	bagging freq	5	5	5
	max_depth	-1	-1	-1
	boosting type:	dart	dart	dart
CV	R^2	0.5886	0.6060	0.6221
Train	RMSE	0.0009	0.0009	0.0009
	MAE	0.0059	0.0054	0.0056
	R^2	0.9536	0.9538	0.9538
Test	RMSE	0.0072	0.0068	0.0068
	MAE	0.0412	0.0375	0.0375
	R^2	0.6375	0.6595	0.6585
Note			Best overall	Good fit

Supervised Model Performance Summary

Linear Models (Linear, Lasso):

- Simple, interpretable baselines
- Consistent RMSE but poor R^2
- Weak at modeling non-linearities

Tree-Based Models (DT, RF):

- Captured feature interactions better
- Random Forest improved generalization over Decision Tree

LightGBM:

- Best R^2, efficient training with DART boosting
- Slight overfitting observed → potential performance plateau

Future Directions:

- Explore richer features
- Try hybrid/ensemble models (e.g., XGBoost)

04

UNSUPERVISED LEARNING

You can enter a subtitle here if you need it

Unsupervised Preprocessing – Overview

1. Apply Predefined definition of bad credit
    - Outer merged credit and applications dataset together
    - Label encoded categorical values
    - Binarised credit_status with bad credit = 1
2. Check for missing data
    - Dropped 25.8% of rows with missing data
3. Remove duplicate IDs to reduce class imbalance
    - Filtered out duplicate IDs
    - Verified default proportion remained unchanged after transformation (1.689% ‘Bad Credit’)
4. Scaling data

K-Means – Motivation & Definitions

K-Means

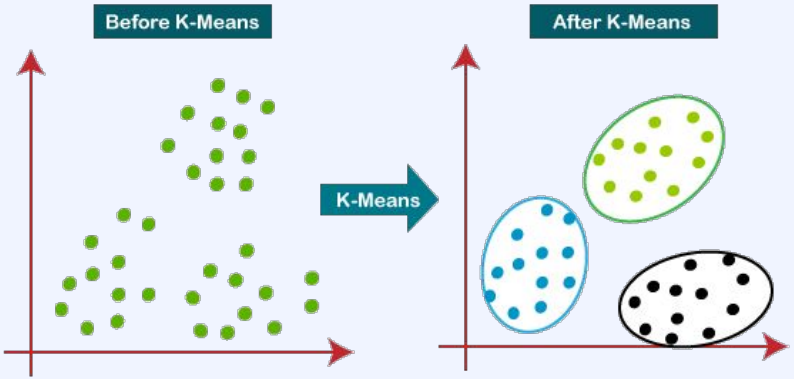
- ❖ Groups similar data points together based on feature similarity
- ❖ Partition the dataset into k distinct, non-overlapping clusters

Motivations:

- ❖ Centroid-based clustering
- ❖ Scalability and computational efficiency
- ❖ Effective on scaled, continuous data
- ❖ Ease of evaluation and hyperparameter tuning

Steps:

- ❖ Select k cluster centroids (hyperparameter tuning)
- ❖ Assign each data point to the nearest centroid
- ❖ Update centroids based on the mean of assigned points
- ❖ Repeat the process until convergence

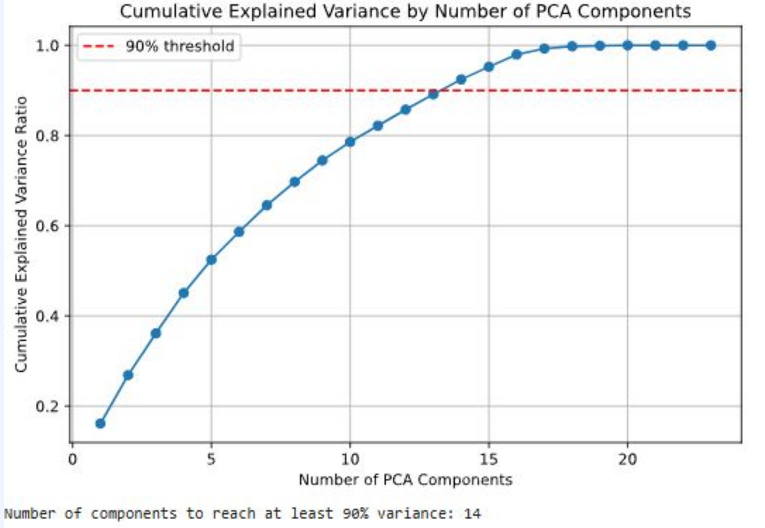


K-Means – Preprocessing (PCA)

- ❖ **Dimensionality reduction**: very high dimension gives curse of dimensionality
- ❖ **Noise reduction**: filter noise by focusing on principal components that capture most variance

Tune n_components by Cumulative Explained Variance:

- ❖ Fit PCA incrementally from 1 component up to min(n_samples, n_features)
- ❖ Track the cumulative explained variance ratio at each step
- ❖ Decide a threshold of 90%



n_components = 14

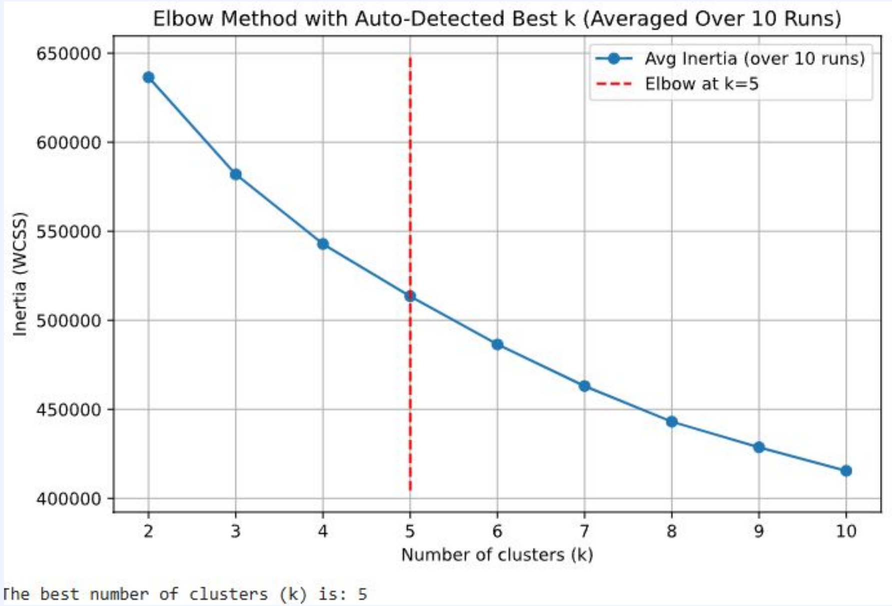
K-Means (Traditional Elbow Method) – Train & Tuning

```
kmeans = KMeans(n_clusters=k, n_init=n_init, init=init_method, random_state=run)
```

Hyperparameters:

- k: number of clusters (to be tuned)
- n_init=10 : number of times the algorithm will run with different starting centroid seeds
- init_method="k-means++" : method for initializing the centroids. Typical method is "k-means++"
- Random_state(run)=42 : sets the seed for random number generator used during centroid initialization

k = 5



Use KneeLocator to detect the "elbow"

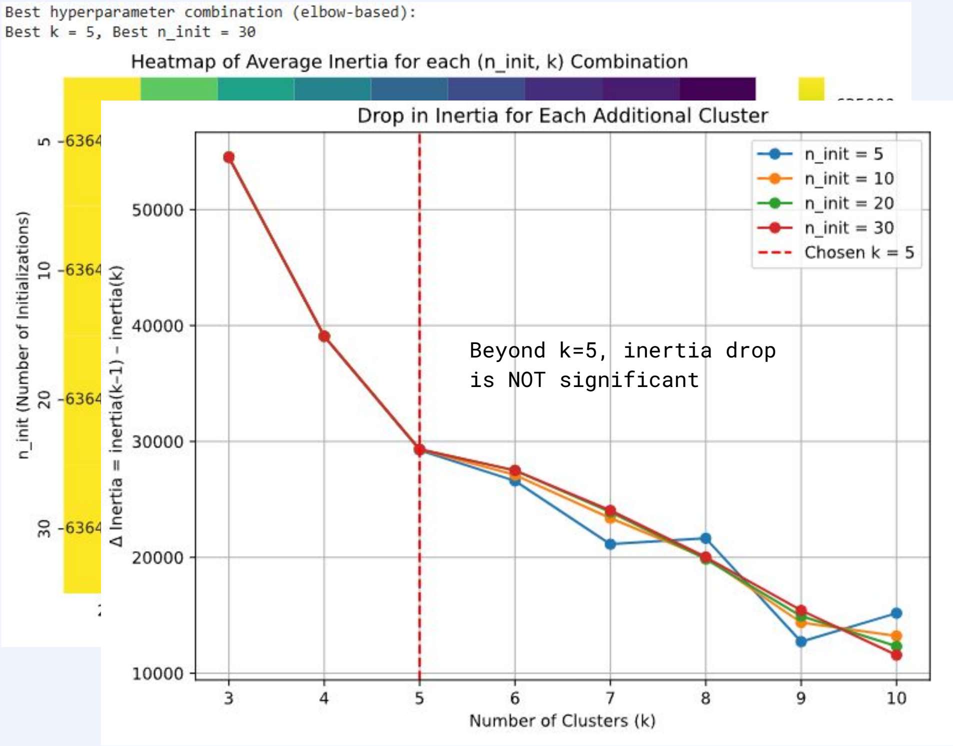
K-Means (Enhanced Elbow Method) – Train & Tuning

```
kmeans = KMeans(n_clusters=k, n_init=n_init, init=init_method, random_state=run)
```

Hyperparameters:

- k: number of clusters (to be tuned)
- n_init : number of times the algorithm will run with different starting centroid seeds (to be tuned)
- init_method="k-means++" : method for initializing the centroids. Typical method is "k-means++"
- Random_state(run)=42 : sets the seed for random number generator used during centroid initialization

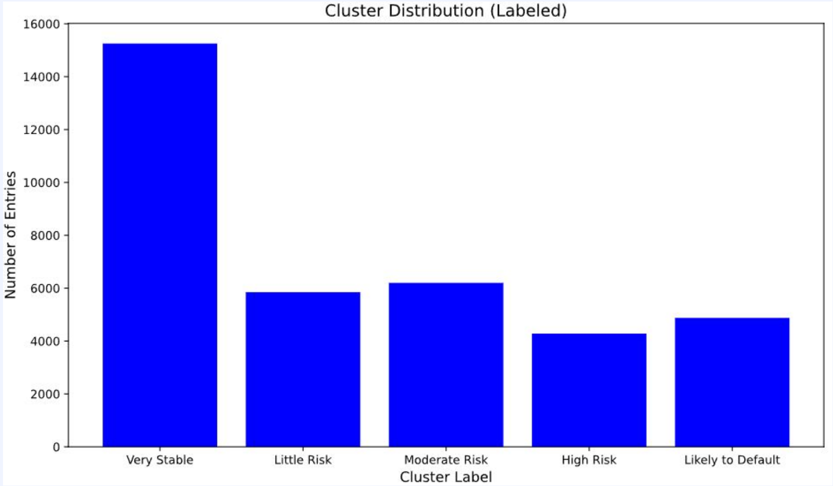
k = 5
n_init = 30



K-Means (Elbow Method) – Clustering

K-Means algorithm is expected to group customers with similar risk profiles into the same cluster.

Each cluster groups together customers with similar “risk” characteristics (e.g. delinquency days, etc.). Once we have run the K-Means, we can compute the centroid of each cluster in the original feature space and look at its average risk metrics. Then, sort these centroids by their mean risk score (by the some first principal components that most correlates with risk).




```
label_map = {
    0: "Very Stable",
    1: "Little Risk",
    2: "Moderate Risk",
    3: "High Risk",
    4: "Likely to Default"
}
```

```
Final K-Means trained with k=5, n_init=30
RiskProfile
Very Stable          15071
Moderate Risk         6127
Likely to Default     5818
High Risk             4840
Little Risk           4249
Name: count, dtype: int64
```

K-Means (Traditional Silhouette Method)

A silhouette score indicates how similar a data point is to its own cluster as compared to other clusters.

Poor Performance

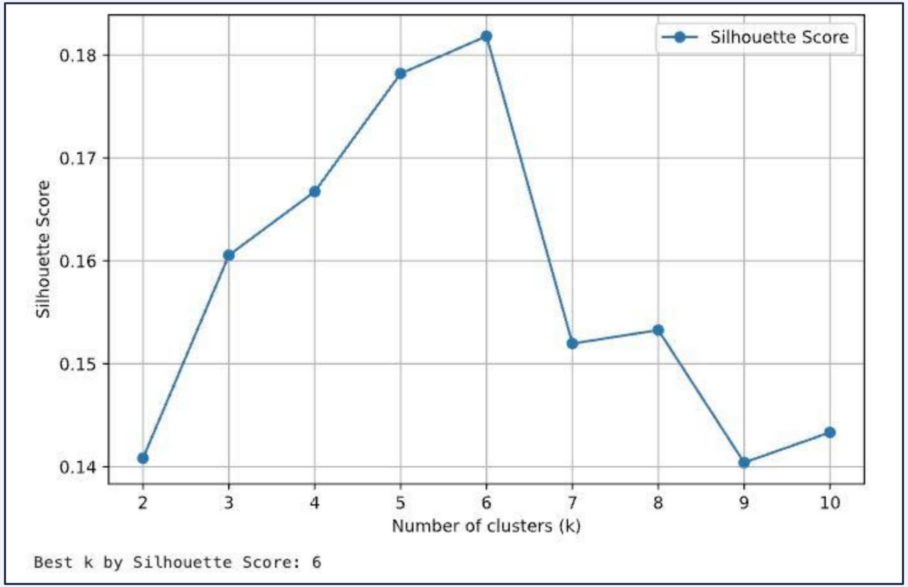
Good Performance

-1 0 1

Hyperparameters:

- k: no. of clusters
- n_init=10
- init_method=”k-means++”
- Random_state(run)=42

Best k = 6



K-Means (Enhanced Silhouette Method)

The enhanced silhouette has an additional parameter, n_init , [5,10,20,30], which determines how many times the algorithm will run with different starting centroid seeds.

k and n_init combination which results in the greatest silhouette score

Poor Performance

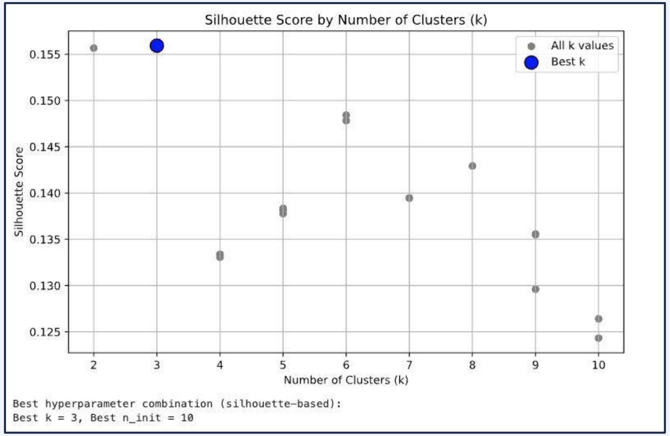
Good Performance

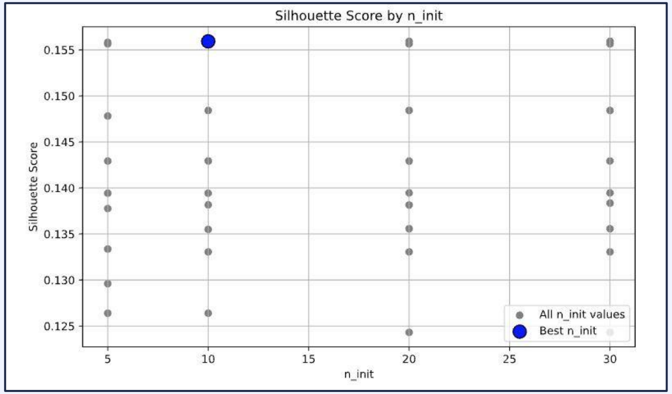
-1 0 1

Hyperparameters:

- k: no. of clusters
- n_init = [5,10,20,30]: no. of times algorithm runs with different starting centroid seeds
- init_method="k-means++"
- Random_state(run)=42

Best k = 3
Best n_init = 10



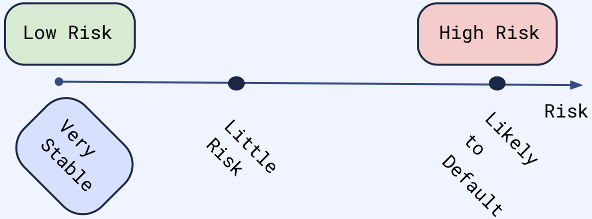


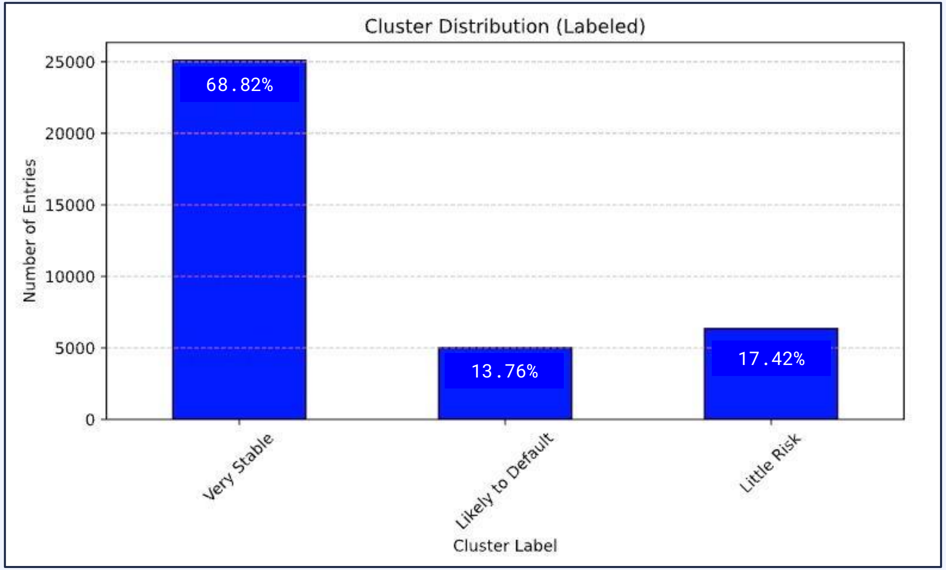
K-Means (Silhouette Method) – Clustering

K-Means algorithm groups the customers into three different clusters based on their risk profiles.

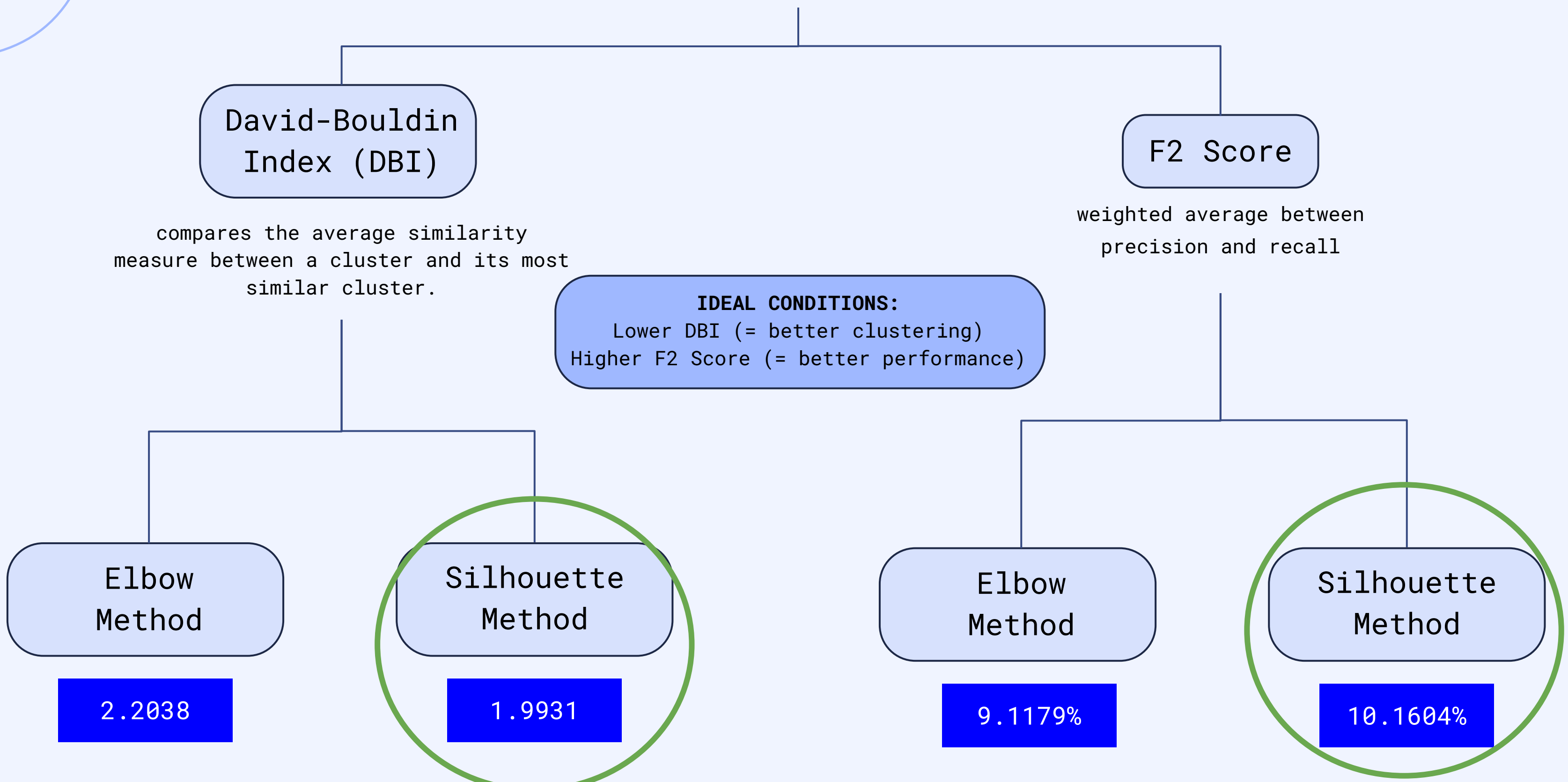
The risk profiles are:

- Likely to Default
- Little Risk
- Very Stable





K-Means Evaluation



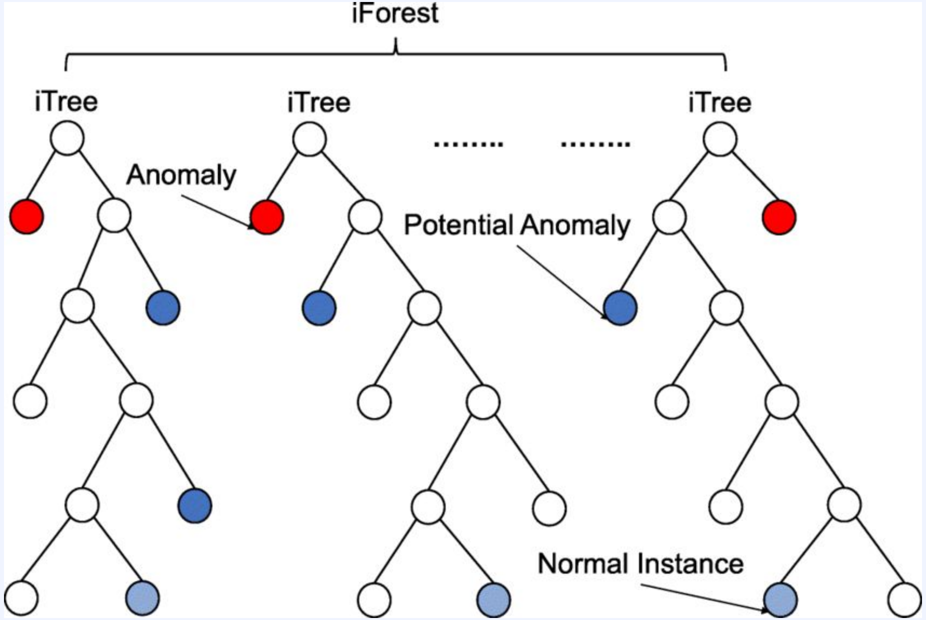
iForest – Motivation & Definitions

Motivation: Highly imbalanced nature of the dataset (1.7% ‘Bad credit’)’ makes it naturally suited for an anomaly detection approach.

iForest Definition: Recursive construction of binary trees (isolation trees) choosing random features and splitting them using the feature minimum and maximum.

Core Intuition: Anomalies requires fewer splits to isolate because they differ significantly from rest of data.

How it works: Uses average path length *h(x)* from root node to terminating node to evaluate anomaly score *s(x, n)*



$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \quad c(m) = \begin{cases} 2H(m-1) - \frac{2(m-1)}{n} & \text{for } m > 2 \\ 1 & \text{for } m = 2 \\ 0 & \text{otherwise} \end{cases}$$

iForest – Training and Tuning

Hyperparameters:

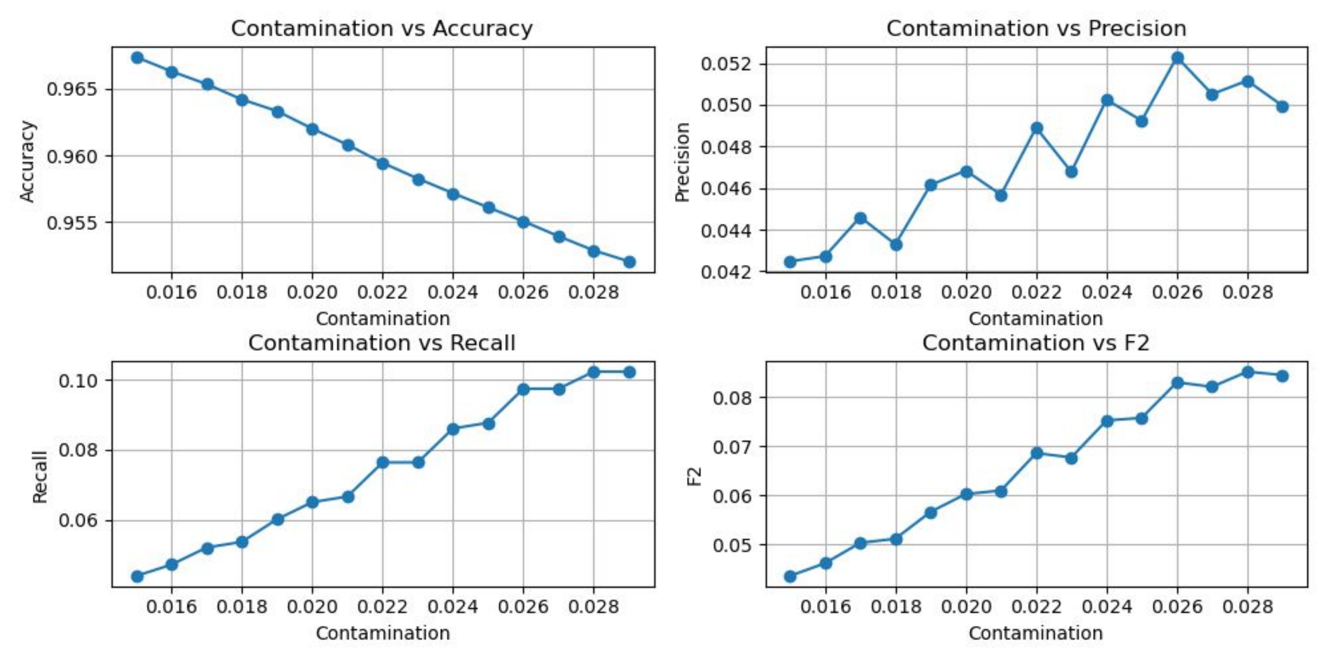
- n_estimators=200: The number of base estimators in the ensemble
- max_samples=1.0: Number of samples to use to train each tree
- contamination=c: The proportion of anomalies in the dataset (to be tuned)
- max_features=1.0: Proportion of features used when training each tree

Evaluation Metrics:

- Precision: TP/(TP+FP)
- Recall: TP/(TP+FN)
- Accuracy: (TP+TN)/Total Data
- F2 Score: (1+β^2) * (Precision + Recall)/(β^2 * Precision + Recall)

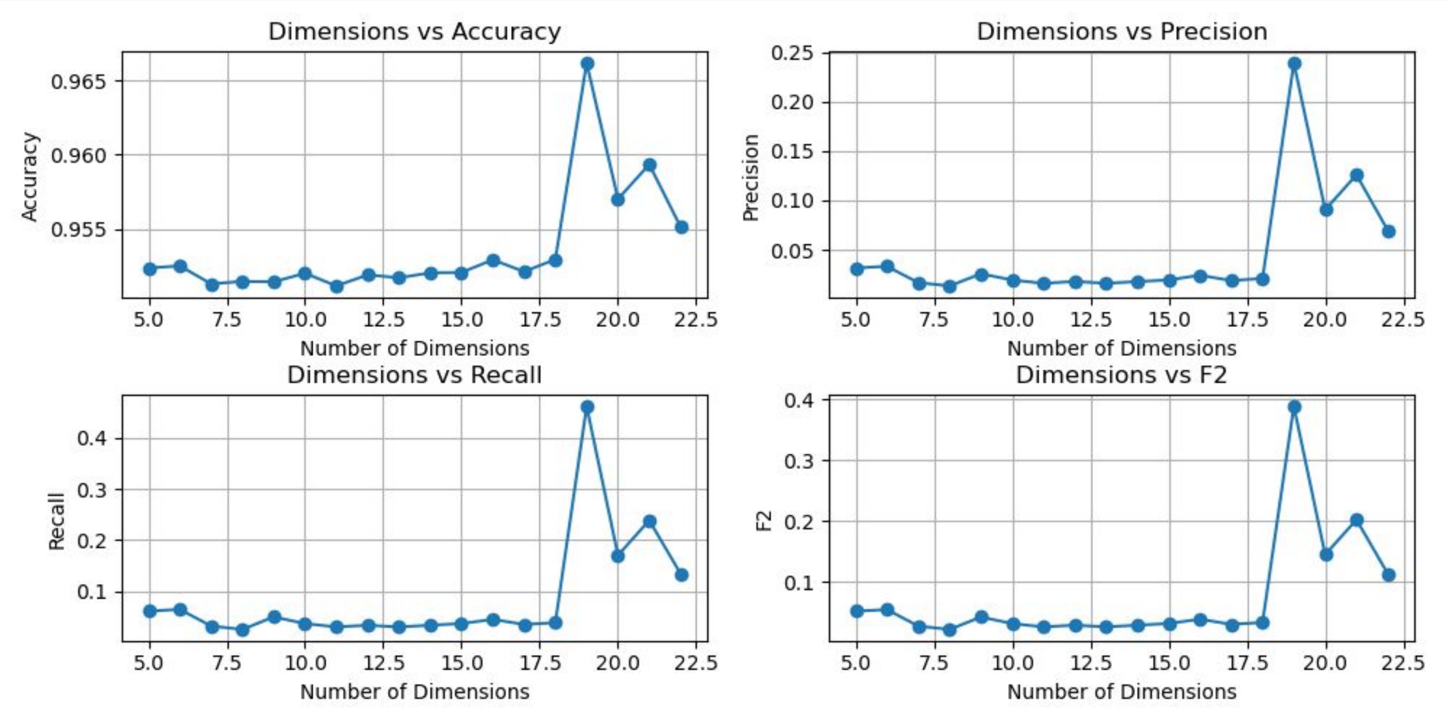
iForest – Base Model 1 (Contamination Tuning)

Hypothesis: Anomalies detected corresponds to ‘Bad Credit’ customers, as they represent a minority class and are more likely to exhibit behavioural patterns that deviate from the majority ‘Good Credit’ group
Contamination range: [1.5% - 3.0%]



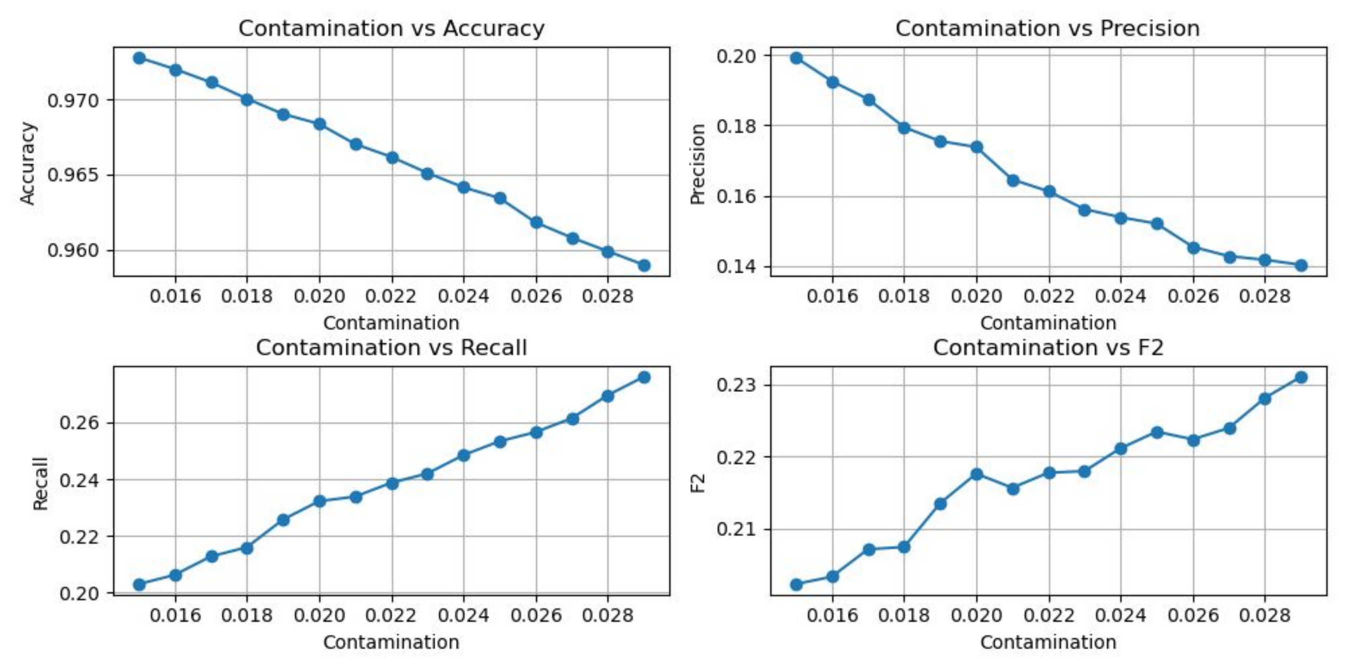
iForest – Model 2 (Autoencoders)

Hypothesis: Autoencoders can reduce data noise and improve model performance
Number of Dimensions: [5 - 22]



iForest – Model 3 (Applying Feature Selection)

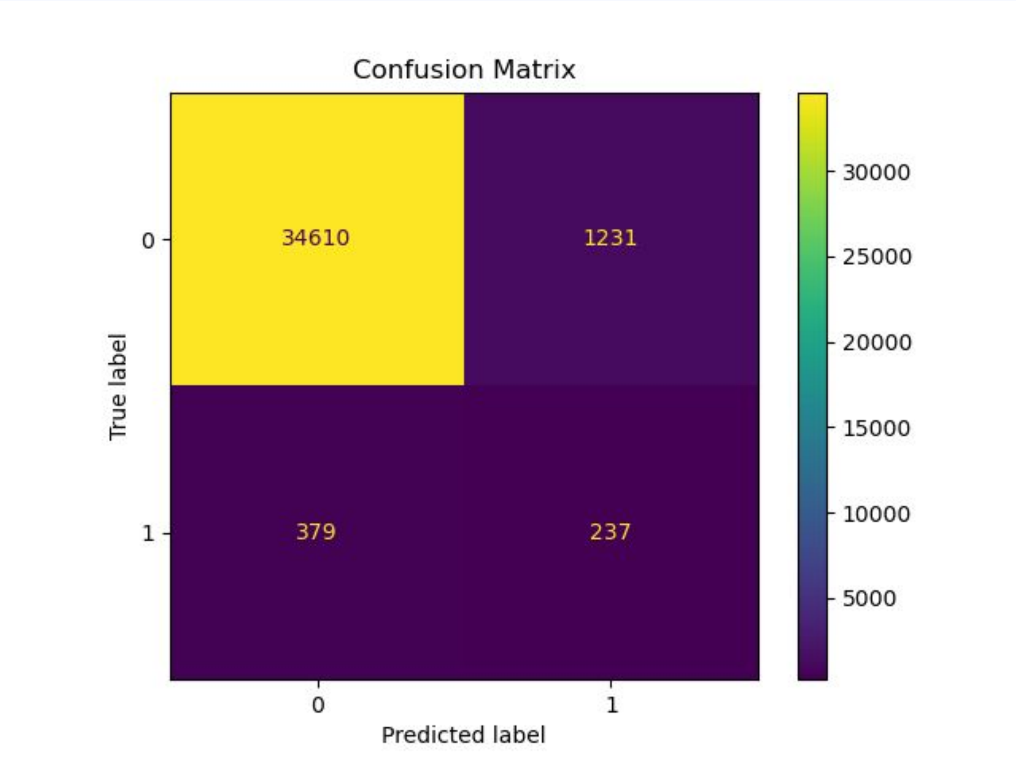
Features used: ['FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'DAYS_BIRTH', 'MONTHS_BALANCE', 'NAME_EDUCATION_TYPE', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE']



iForest – Model 4 (Using All Good Credit)

Hypothesis:
Training model on only good credit can improve extreme class imbalance and improve performance

Accuracy: 95.6%
Precision: 16.2%
Recall: 38.5%
F2 Score: 30.2%



Unsupervised Overall Evaluation

Criteria	Isolation Forest	K-means
F2 Score	30.2%	10.2%
Interpretability	Moderate – based on anomaly scores, harder to trace specific decisions	High – centroids are intuitive, but may not capture complex patterns
Efficiency	Training: O(t*m*logm) Inference: O(t*logm)	Training: O(n*k*i*d) Inference: O(k*d)

05

PROJECT EVALUATION

You can enter a subtitle here if you need it

Overall Evaluation

Model Evaluation Summary Table (✓ = Yes, ✗ = No, ~ = Partial)

Criteria / Model	Linear Reg.	Lasso Reg.	Decision Tree	Random Forest	LightGBM	K-Means	Isolation Forest
Handles Non-linearity	✗	✗	✓	✓	✓	✓	~
Robust to Outliers	✗	✗	~	✓	✓	✗	✓
Good Interpretability	✓	✓	✓	~	~	✗	✗
Feature Importance Available	✗	✓	✓	✓	✓	✗	✗
Handles Target Imbalance	✗	✗	✗	~	✓	✓	✓
Scalable to Large Datasets	✓	✓	✗	✓	✓	✗	✓
Hyperparameter Tuning Required	✗	✓	✓	✓	✓	✓	✓
Effective with High Dimensionality	✓	✓	✗	✓	✓	✗	✓
Produces Usable Credit Score Output	✓	✓	✓	✓	✓	✗	✗
Visual Interpretability (Plots etc.)	✓	✓	✓	~	✓	✓	✓

THANK YOU!