PORTUGUESE BANK MARKETING DATA SET – SUBSCRIPTION PREDICTION

IE7300 Statistical Learning in Engineering
(Spring 2023)

PROJECT GROUP 7

Dinesh Balasubramanian

Siddharthan Singaravel

Suna Lee

Yunfan Dai

Introduction

Portuguese banking institutions have been experimenting with direct marketing campaigns in recent years as a component of their marketing plan to increase sales and maintain track of clients. This campaign's objective is to induce clients to sign up for one of the bank's financial products (term deposit is taken into consideration here). To aid with the campaign evaluation for the specified issue statement, the bank's client was required to inform the institution after each call whether they intended to subscribe to the bank term deposit (marking it as a successful campaign) or not (marking it as an unsuccessful campaign).

Objective

The project's objective is to predict which customers and market groups would respond favorably (subscribing to bank term deposit is taken into consideration here) to a bank's direct marketing campaign and to identify the factors that help banks successfully entice customers to subscribe to their campaigns. Also, it would help the banks choose high-value clients who pose fewer risks.

Plan

We can predict the success of the marketing campaigns by using three supervised learning algorithms.

- Logistic Regression
- K – Nearest Neighbors
- Support Vector Machines

The best algorithm is judged based on classification cross validation techniques such as accuracy, f-score, precision, and recall.

Dataset Description

The dataset is related to direct marketing campaigns run by a certain Portuguese bank. To determine if the bank term deposit would be subscribed to or not, it was necessary to make more than one contact with the same client.

The dataset consists of ~42,000 records (and ~4000 records for testing) which represent the instances of calls to the clients and 21 features, including the target feature indicating if the campaign was a success or not. The features are categorized into 5 types for better interpretability.

Data Source: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Category	Feature
Client Data	1. age (numeric)
	2. job (categorical)
	3. marital (categorical)
	4. education (categorical)
	5. credit_in_default (categorical)
	6. housing (categorical)
	7. loan (categorical)
Previous Contact Data	8. contact (categorical)
	9. month (categorical)
	10. day_of_week (categorical)
	11. duration (numeric)
Socio and Economic Context Attributes	12. employment_variation_rate (numeric)
	13. consumer_confidence_index (numeric)
	14. number_of_employees (numeric)
	15. consumer_price_index (numeric)
	16. euribor_3_month_rate (numeric)
Other Features	17. campaign (numeric)
	18. pdays (numeric)
	19. previous (numeric)
	20. poutcome (categorical)
Output Variable	21. y (binary – yes/no)

Table 1. Dataset Description

Exploratory Data Analysis and Data Preprocessing

Target Class Distribution

The dataset is heavily imbalanced with only 11.3% of the total records marked as the class of interest (yes).

Fig 1. Target Class Distribution

Total Number of Clients by Age

The below plot charts the client distribution by their age. A majority of the clients are within the age group of 30 – 50 with a mean age of 40. 19% of the clients in the age group of 20 – 29 were marked 'yes', significantly higher than the dataset mean.

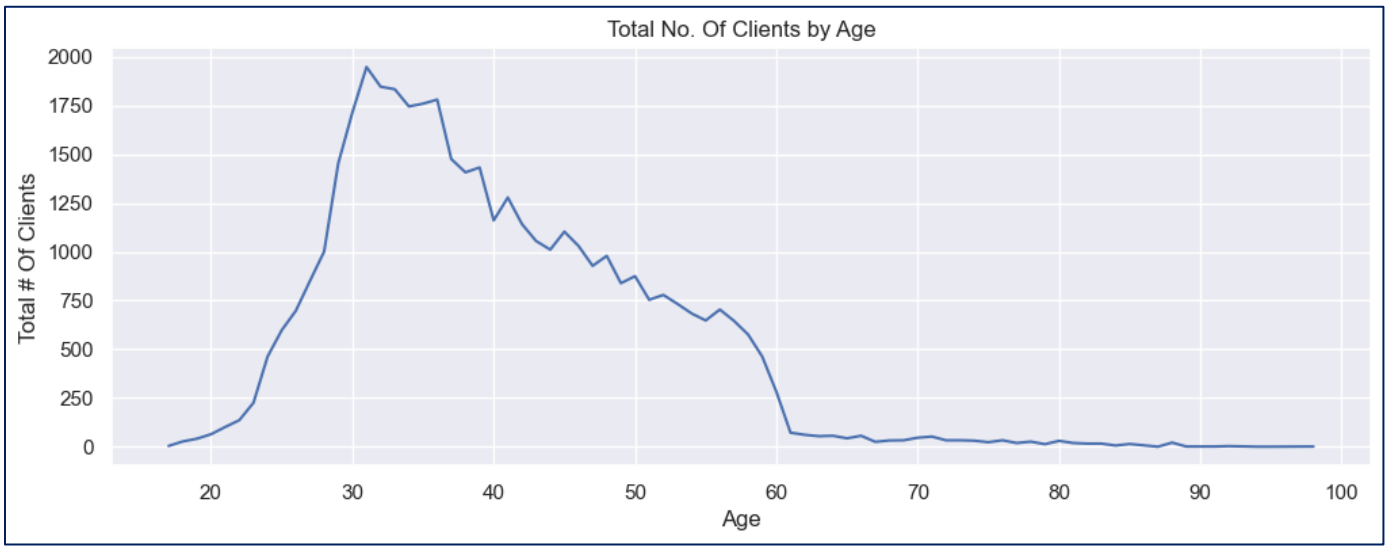

Fig 2. Total Number of Clients by Age

Conversion Rates by Age

The below graph shows how conversion rates change with the age of the client. Clients above the age of 60 are more likely to be marked a success than clients below the age of 60. A hypothesis for the case is that clients aged 60+ are more likely to spend time on the phone with the customer representative which in turn impact call outcome.

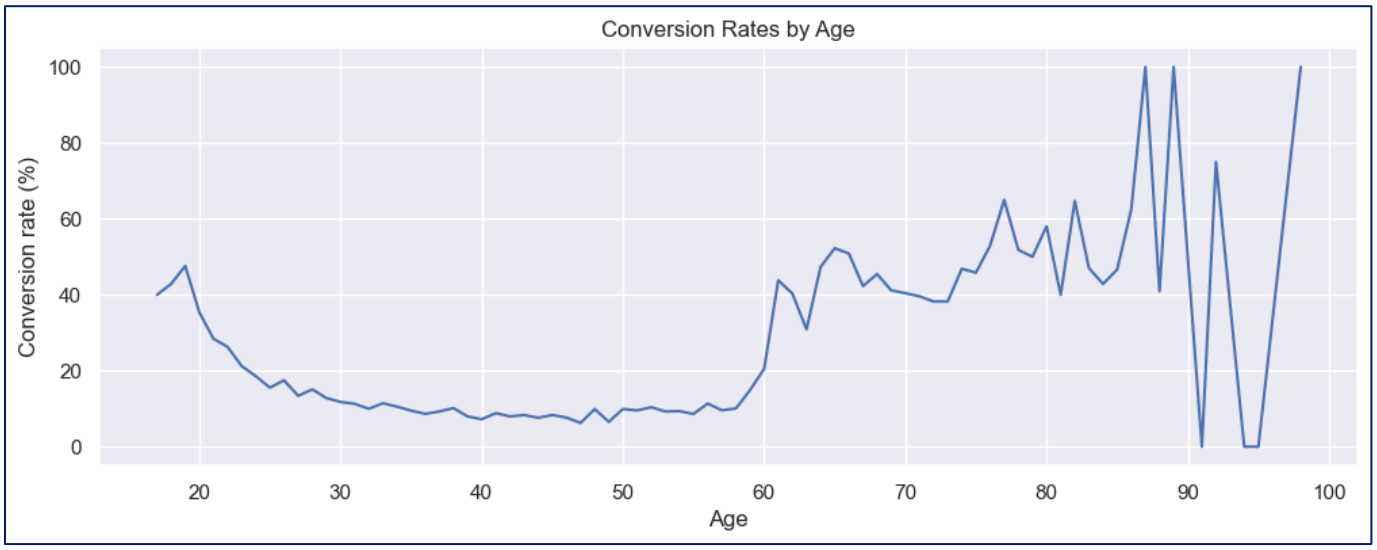

Fig 3. Conversion Rates by Age

Conversion Rates by Age and Marital Status

The below graph shows a breakdown of the clients by their marital status, showing the target population for the bank.

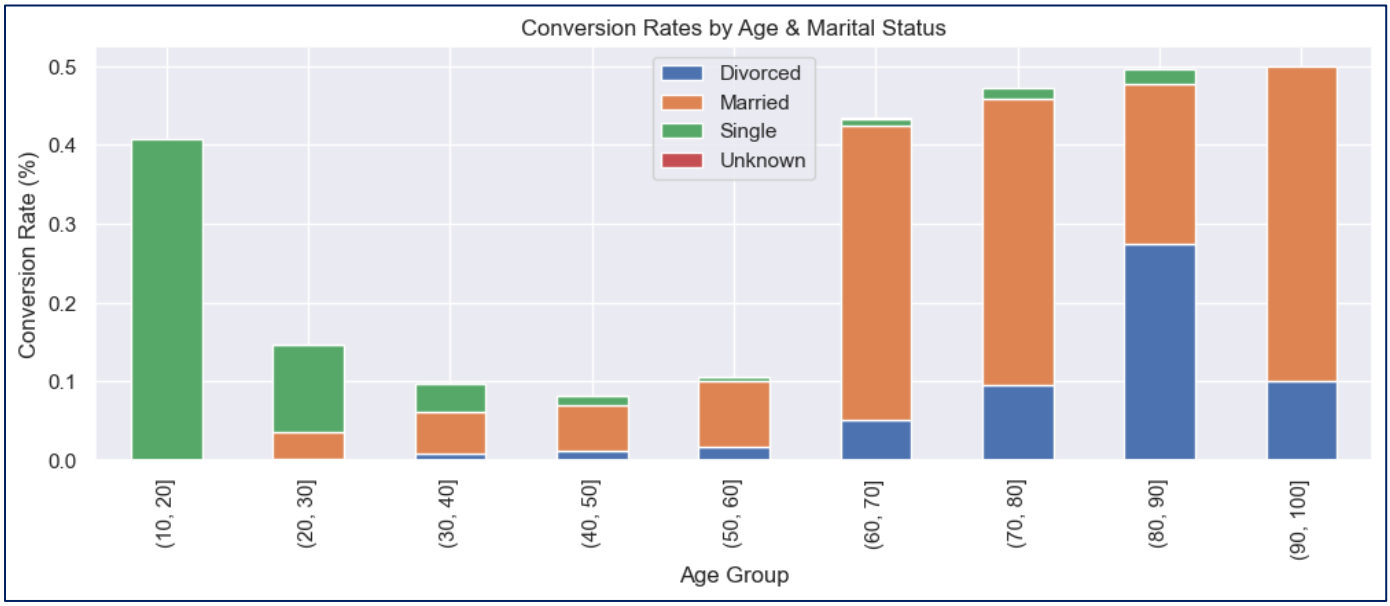

Fig 4. Conversion Rates by Age and Marital Status

Relationship between Number of Calls and Duration of Calls

Though more calls lasted fewer than 100 seconds, calls that lasted longer than 500 seconds were more likely to be marked success. Another interesting observation is how fewer calls were made to people who subscribed.

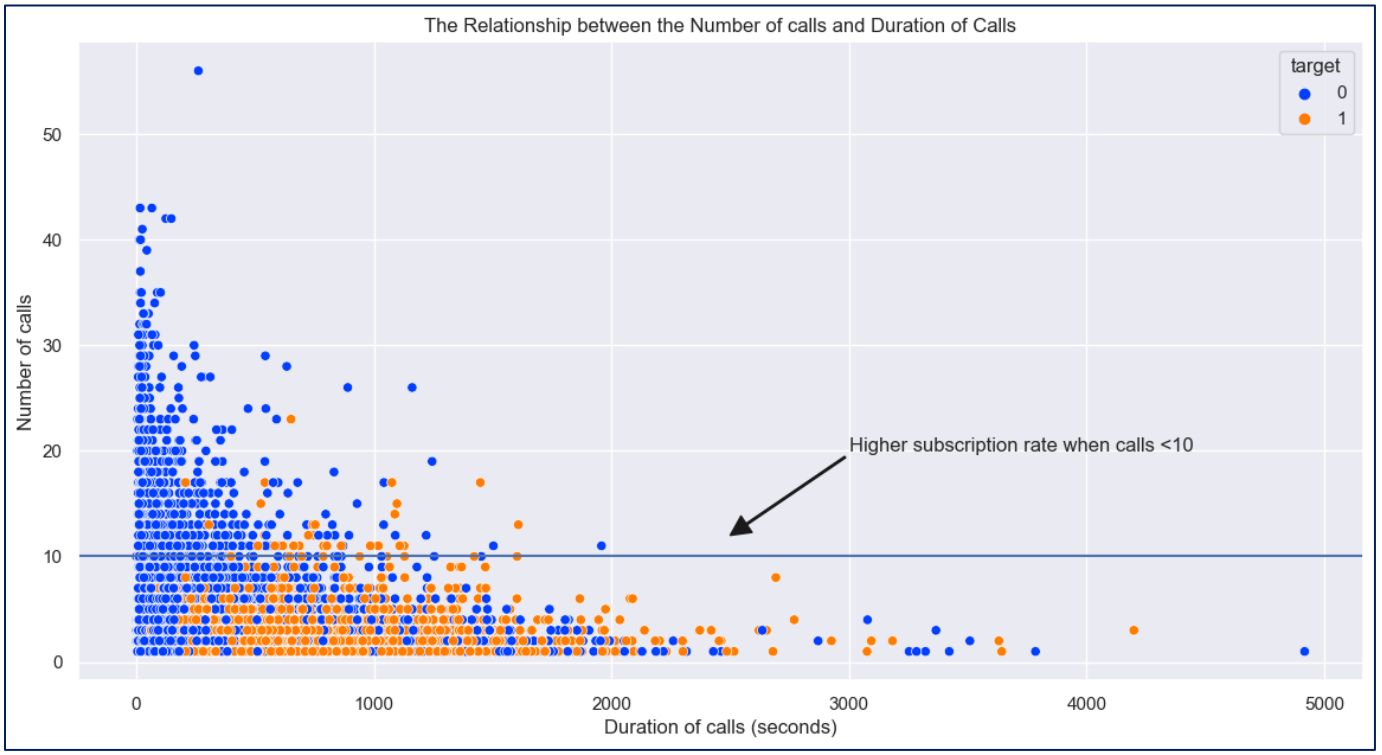

Fig 5. Number of Calls vs Duration of Calls

Correlation Analysis – Categorical Features

No meaningful patterns emerge only from the categorical features considering the success class is distributed within the categories of each feature.

Fig 6. Correlation Analysis – Categorical Features

Pearson Correlation between Macroscopic Factors and Target Variable

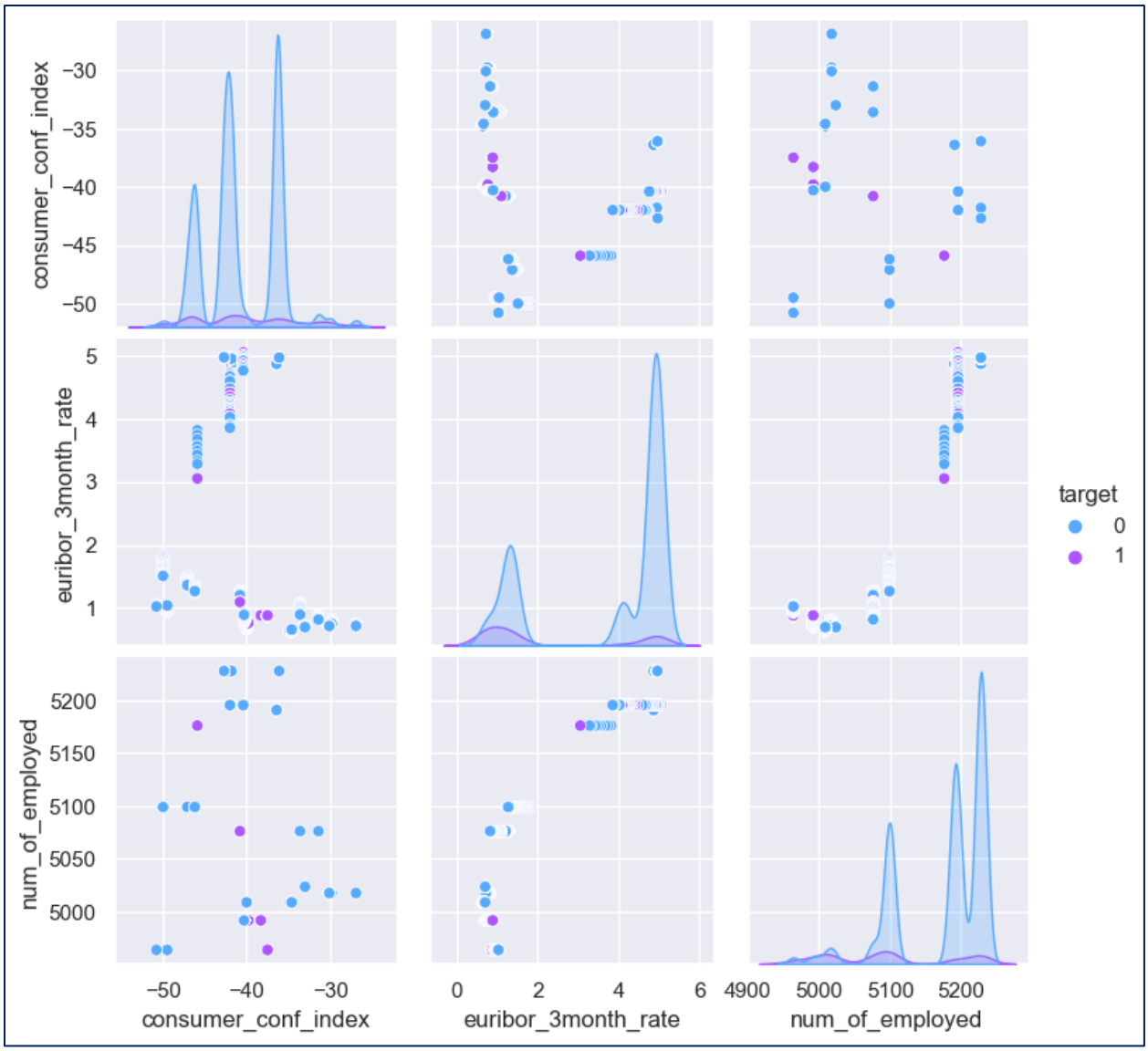

Fig 7. Correlation between Macroscopic Economic Factors and Target Variable

Visualizing Skewness in Continuous Features

The graph below shows the skewness of the data variables. As observed, there is no recognizable pattern that will help directly associate the distribution of the numerical feature with the target variable. Naïve Bayes that assumes its numerical features to be normally distributed cannot be employed considering the weak normality observed in the graphs.

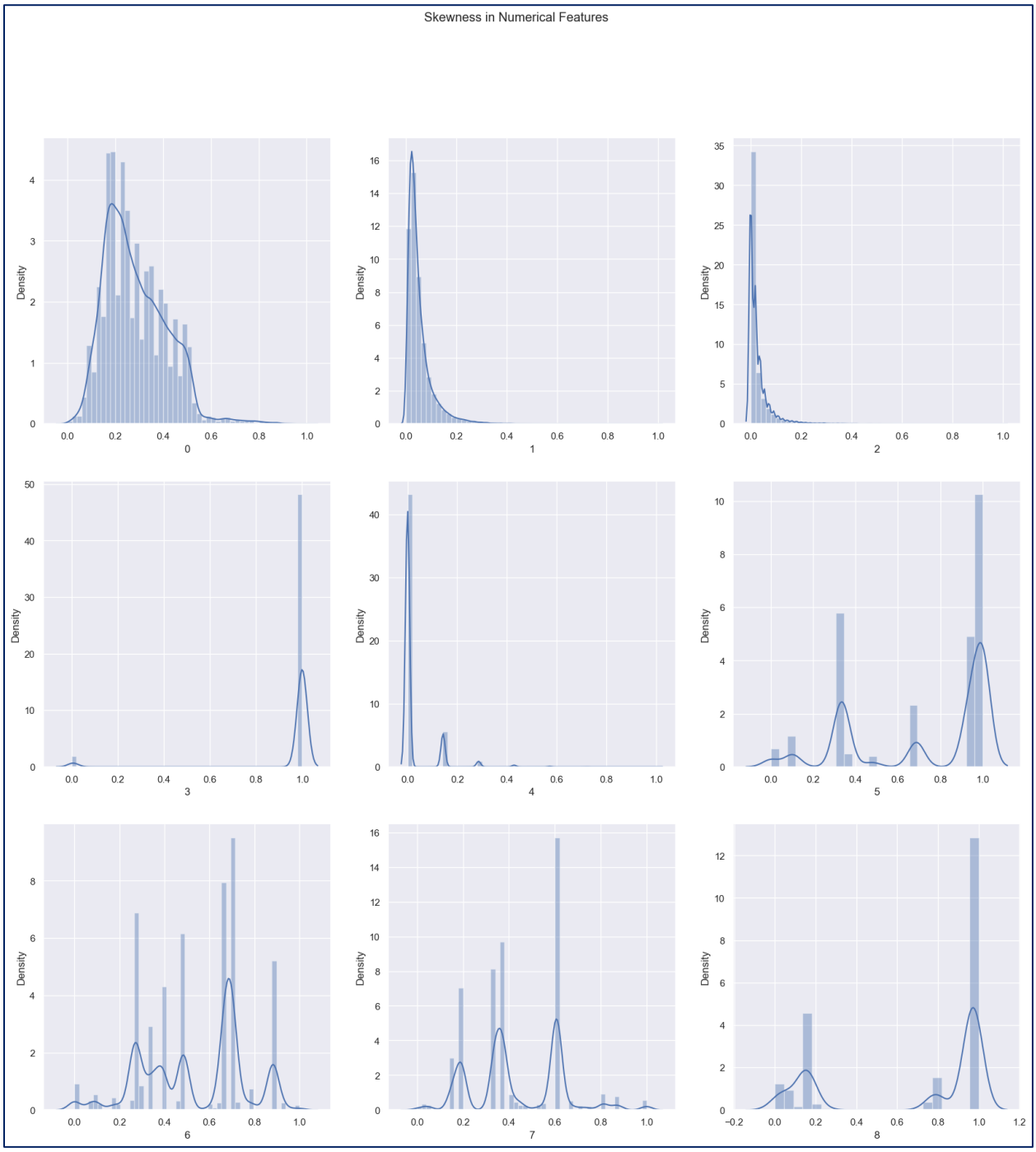

Fig 8. Skewness in Numerical Features

Analysing Outliers

The dataset is modelled after removing extremities from the data matrix.

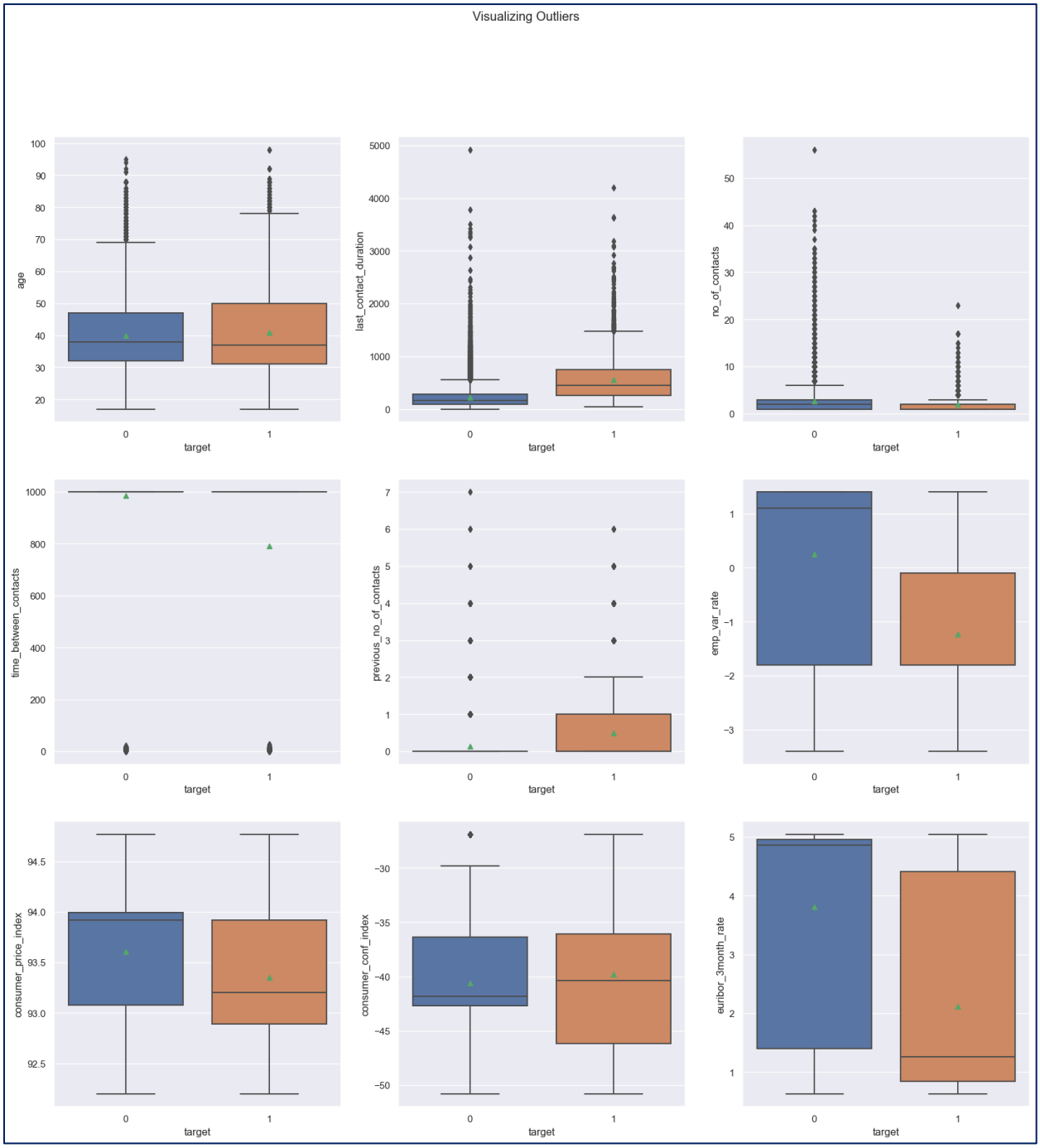

Fig 9. Visualising Outliers

High Correlation Filter - Visualizing Numerical Feature Correlations

Using a high-correlation cut-off of 0.75, *euribor_3month_rate*, *num_of_employed*, and *consumer_price_index* highly correlated with *emp_var_rate* are dropped from the matrix.

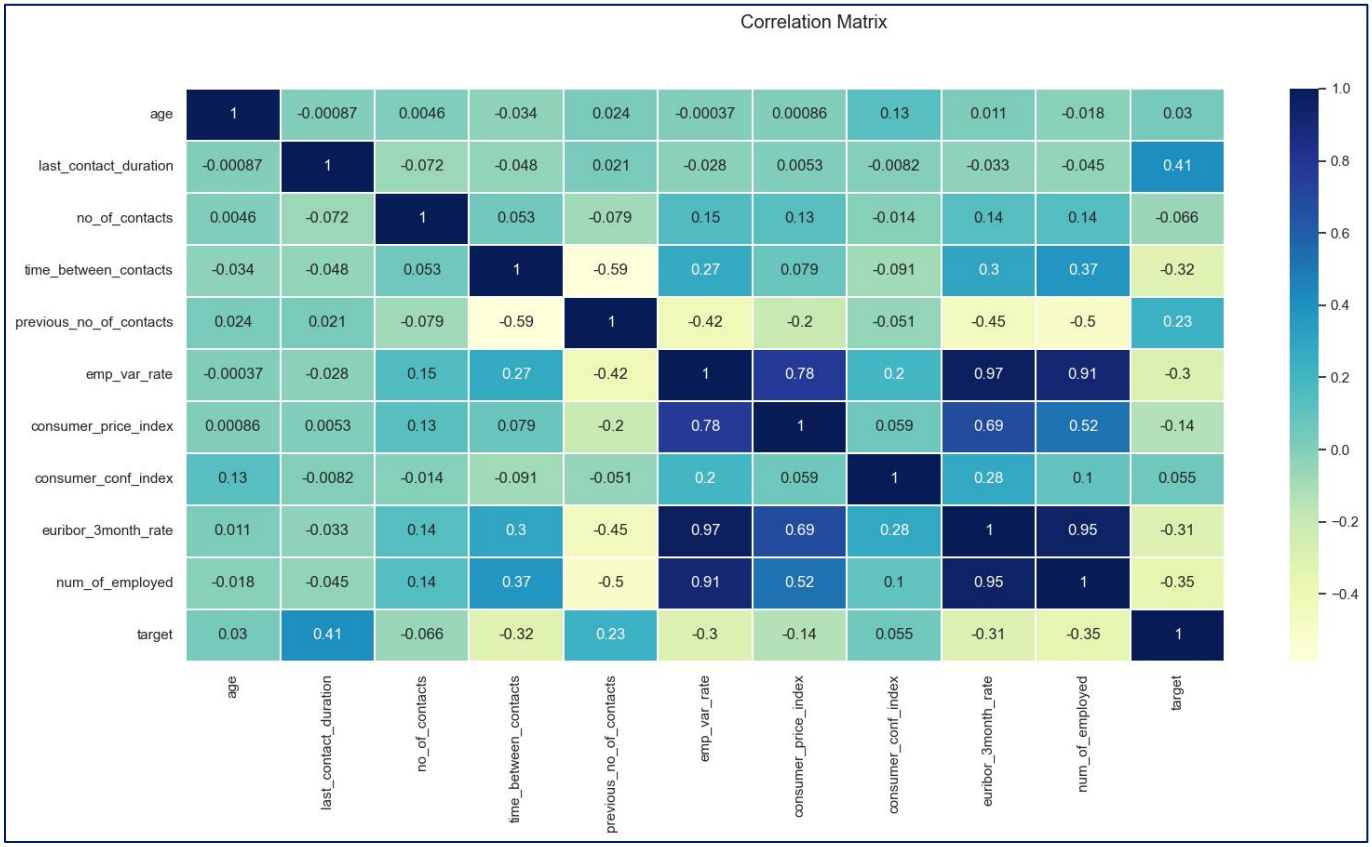

Fig 10. Correlation Analysis – Numerical Features

Imbalanced Dataset – Oversampling Methods

Synthetic Minority Oversampling TEchnique (SMOTE)

To generate synthetic instances using SMOTE, the algorithm randomly selects a minority class instance "a" and identifies its "k" nearest neighbors also belonging to the minority class. The synthetic instance is then created by selecting one of these nearest neighbors, "b," at random and connecting "a" and "b" to form a line segment in the feature space. The synthetic instance is generated by taking a convex combination of the feature values of "a" and "b."

The image below shows how minority classes are oversampled by building regions that contain nearby minority class points.

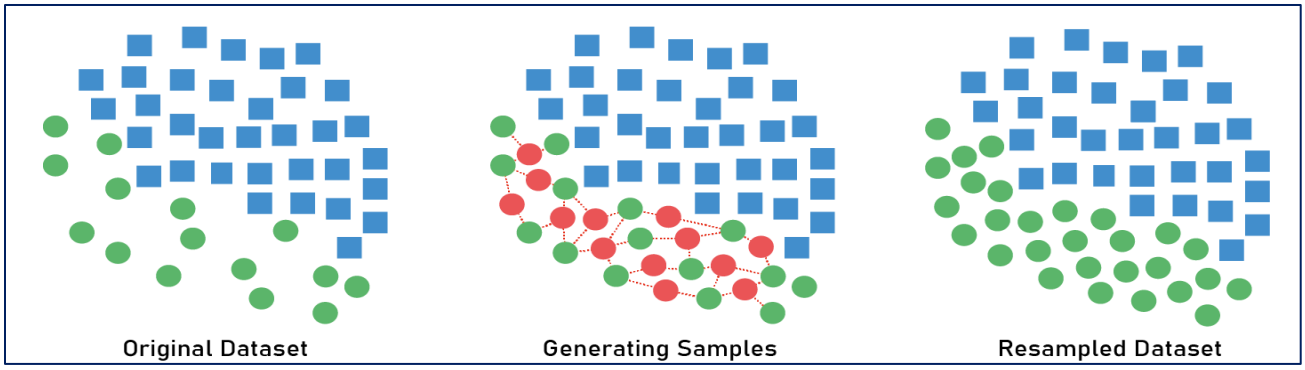

Fig 10. SMOTE Algorithm

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100) * T$ synthetic minority class samples

(∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
if $N < 100$
  then Randomize the T minority class samples
    $T = (N/100) * T$
    $N = 100$
endif
$N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
k = Number of nearest neighbors
$numattrs$ = Number of attributes
$Sample[\,][\,]$: array for original minority class samples
$newindex$: keeps a count of number of synthetic samples generated, initialized to 0
$Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
for $i \leftarrow 1$ **to** T
    Compute k nearest neighbors for i, and save the indices in the $nnarray$
    Populate(N, i, $nnarray$)
endfor

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)
while $N \neq 0$
    Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
    for $attr \leftarrow 1$ **to** $numattrs$
      Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
      Compute: gap = random number between 0 and 1
      $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
    endfor
    $newindex{+}{+}$
    $N = N - 1$
endwhile
return (∗ *End of Populate.* ∗)
End of Pseudo-Code.

Fig 11. SMOTE Pseudocode

Adaptive Synthetic Algorithm (ADASYN)

ADASYN is an oversampling technique that tackles imbalanced datasets by generating synthetic data points for the minority class. Unlike other oversampling methods, ADASYN introduces a weighted distribution to generate more synthetic data for minority class samples that are difficult to learn. By doing so, the algorithm focuses on improving the classification performance for the hard-to-learn minority class examples, which can significantly enhance the overall model accuracy. As with SMOTE, ADASYN generates synthetic observations along a straight line between a minority class observation and its k-nearest minority class neighbors.

1. Let n_l denote the number of observations of the majority class and let n_s denote the number of observations of the minority class. Calculate $G = (n_l - n_s) \times \beta$.
2. Let x_i, $i = 1, \ldots, n_s$, denote the observations belonging to the minority class and let A denote the set of all x_i, such that $A \ni x_i$. For every x_i:
3. Calculate the Euclidean distance between x_i and all other elements of A to obtain the k-nearest neighbors of x_i.
4. Let S_{ik} denote the set of the k-nearest neighbors of x_i,
5. Define Δ_i as the number of observations in the k-nearest neighbors' region of x_i that belong to the majority class. Calculate the ratio r_i defined as $r_i = \Delta_i / k, i = 1, \ldots, n_s$.
6. Normalize r_i according to $\widehat{r_i} = r_i / \sum_{i=1}^{n_s} r_i$, so that $\widehat{r_i}$ is a probability $\left(\sum_i \widehat{r_i} = 1\right)$.
7. Calculate $g_i = \widehat{r_i} \times G$, which is the number of synthetic observations that need to be generated for each x_i.
8. Randomly sample g_i synthetic observations denoted x_{ij},$(j = 1, \ldots, g_i)$ from S_{ik} with replacement.
9. Let λ denote a number in the range [0,1]. For a given x_{ij}, generate a synthetic observation according to $x_k = x_i + \lambda\ (x_i - x_{ij})$, where λ is uniformly drawn for each x_k.
10. Stop algorithm.

Fig 12. ADASYN Pseudocode

Dimensionality Reduction (Principal Component Analysis)

Principal Component Analysis (PCA)

PCA is a statistical technique that is widely used in machine learning and data analysis to reduce the complexity of high-dimensional datasets. The main objective of PCA is to extract the most important patterns from the dataset and represent them in a smaller set of variables or dimensions, which are called principal components.

To achieve this, PCA transforms the original variables of the dataset into a new set of uncorrelated variables that are a linear combination of the original ones. These principal components are arranged in decreasing order of importance, with the first principal component explaining the most significant amount of variance in the data. The number of principal components to be retained is determined by the amount of variance they can explain, and this is a crucial factor that affects the quality of the dimensionality reduction.

```
Var Ratio with 1 components: 0.13544303104732933
Var Ratio with 2 components: 0.25342800980775904
Var Ratio with 3 components: 0.3498739185257226
Var Ratio with 4 components: 0.4299553377238861
Var Ratio with 5 components: 0.49788396443916916
Var Ratio with 6 components: 0.5625283335515611
Var Ratio with 7 components: 0.6252773119667658
Var Ratio with 8 components: 0.6807083936192346
Var Ratio with 9 components: 0.7172530973831236
Var Ratio with 10 components: 0.7518156414702122
Var Ratio with 11 components: 0.7840679561446422
Var Ratio with 12 components: 0.8134824945324802
Var Ratio with 13 components: 0.8348380697878862
Var Ratio with 14 components: 0.8537716274476606
Var Ratio with 15 components: 0.8719994603528535
Var Ratio with 16 components: 0.8888031303534288
Var Ratio with 17 components: 0.9052854303867174
Var Ratio with 18 components: 0.9189097976505998
Var Ratio with 19 components: 0.9305914356544139
```

Fig 13. Principal Component Analysis (PCA) – n_components

PCA in this case is not very helpful considering 18 components in the Z-axis are required to retain 90% of the original data variance, which is only slightly fewer than the number of columns in the original dataset. Loss of information is another major drawback with the employment of this method. So, for this project, we intend to work on the normalized untransformed dataset.

Model Implementation

Logistic Regression

Logistic model, also known as logit model, is a statistical approach used to model the likelihood of an event occurring achieved by expressing the log-odds for the event as a linear combination of one or more independent variables. In regression analysis, the focus is on estimating the parameters of the logistic model, which corresponds to the coefficients in the linear combination.

Logistic Regression is considered Naïve Bayes' discriminative counterpart. $P(y|\mathbf{x}_i)$ is modelled as the equation below based on the sigmoid function.

$$P(y|\mathrm{x}_i) = \frac{1}{1 + e^{-y(\mathrm{w}^T\mathrm{x}_i + b)}}$$

In the Maximum a Posteriori (MAP) estimate, we find the parameters that maximize the posterior.

$$P(w \mid D) = P(w \mid X, y) \propto P(y \mid X, w)P(w)$$

$$\hat{w}_{MAP} = \underset{w}{argmax}\ log(P(y \mid X, w)P(w)) = \underset{w}{argmin} \sum_{i=1}^{n} log\ (1 + e^{-y_i w^T x_i}) + \lambda w^\top w,'$$

We can use negative log posterior since Logistic Regression has no closed form solution. So, we can use Gradient Descent optimization on the equation $\ell(\mathbf{w}) = \sum_{i=1}^{n} log\ (1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) + \lambda \mathbf{w}^\top \mathbf{w}$

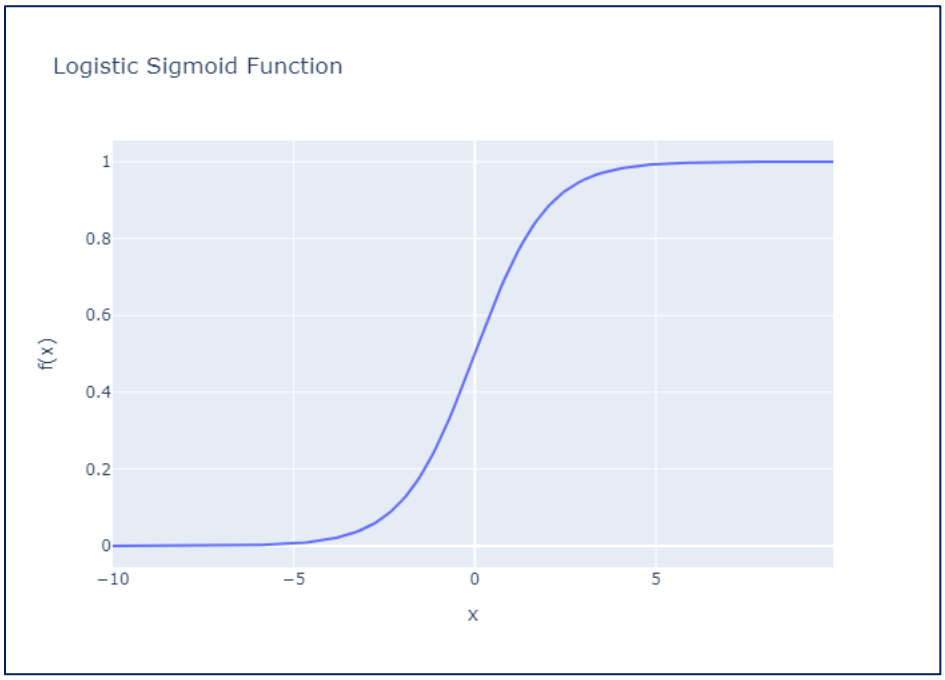

Fig 14. Logistic Sigmoid Function

SMOTE Model Cost Function on Custom Logit Function

By varying model hyperparameters (error threshold, learning rate), we have obtained the best performing model on the SMOTE sampled data. Below are the arguments for the logit class object.

- LearningRate = 1e-4
- Epsilon = 0.0001
- maxIteration = 10,000

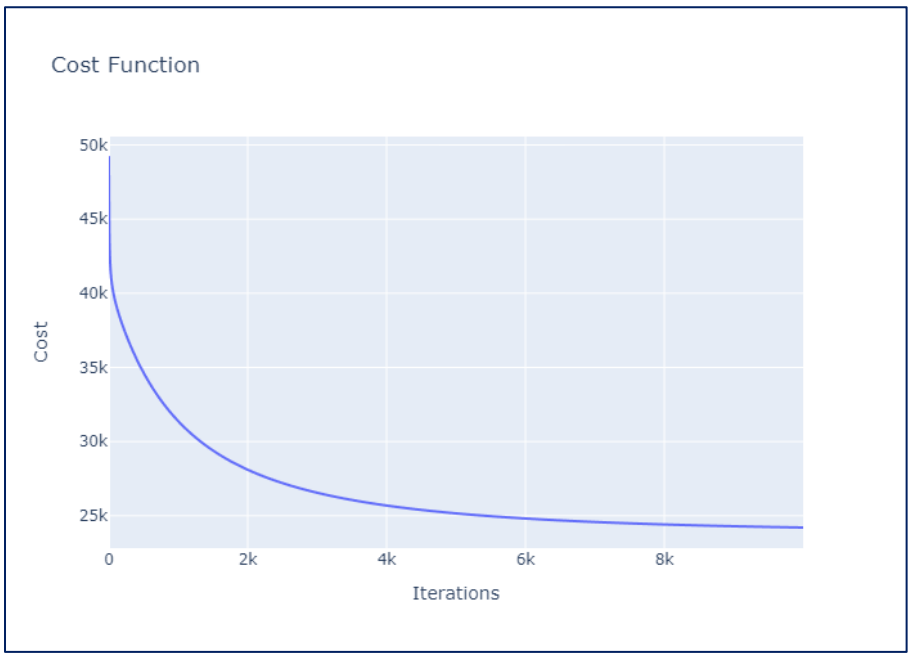

Fig 15. Logistic Regression Cost Function

Logistic Regression: Model Performance Metrics

Model	Accuracy	Recall
SMOTE Model (Custom)	0.802	0.938
ADASYN Model (Custom)	0.804	0.938
sklearn (Original)	0.908	0.246

Table 2. Logistic Regression Summary

k-Nearest Neighbors Classifier (k-NN)

The k-nearest neighbors algorithm, or k-NN, is a non-parametric supervised learning method. The input to the algorithm consists of the k closest training examples to be considered. For k-NN, the output is the class membership of the object being evaluated. This is determined by a plurality vote of the k nearest neighbors, with the object being assigned to the class that is most common among its neighbors. If k is equal to 1, the object is simply assigned to the class of its single nearest neighbor.

One notable feature of k-NN is that it approximates the function locally, and all computation is deferred until the function evaluation stage. Since the algorithm relies on distance (Minkoski, in our case) for classification, it is important to normalize the training data if the features represent different physical units or come in vastly different scales. This can dramatically improve the accuracy of the algorithm.

Minkowski Distance

Minkowski distance is a distance metric that is used to measure the distance between two points in a n-dimensional space, a generalization of Euclidean distance and Manhattan distance.

$$(\sum_{i=1}^{n} |X_i - Y_i|^p)^{1/p}$$

- Euclidean, when p = 2
- Manhattan, when p = 1

Performance Tuning by Varying k

```
k-NN Model k = 1 | Precision = 0.778, Recall = 0.942, Accuracy = 0.966, F-score = 0.852
k-NN Model k = 2 | Precision = 0.778, Recall = 0.942, Accuracy = 0.966, F-score = 0.852
k-NN Model k = 3 | Precision = 0.533, Recall = 0.923, Accuracy = 0.908, F-score = 0.676
k-NN Model k = 4 | Precision = 0.545, Recall = 0.923, Accuracy = 0.912, F-score = 0.686
k-NN Model k = 5 | Precision = 0.462, Recall = 0.923, Accuracy = 0.88, F-score = 0.615
k-NN Model k = 6 | Precision = 0.495, Recall = 0.904, Accuracy = 0.894, F-score = 0.639
k-NN Model k = 7 | Precision = 0.403, Recall = 0.923, Accuracy = 0.85, F-score = 0.561
k-NN Model k = 8 | Precision = 0.431, Recall = 0.904, Accuracy = 0.866, F-score = 0.584
k-NN Model k = 9 | Precision = 0.362, Recall = 0.904, Accuracy = 0.824, F-score = 0.516
```

Fig 16. Performance Tuning by Varying k

The below graph shows how varying k impacts model performance (Accuracy).

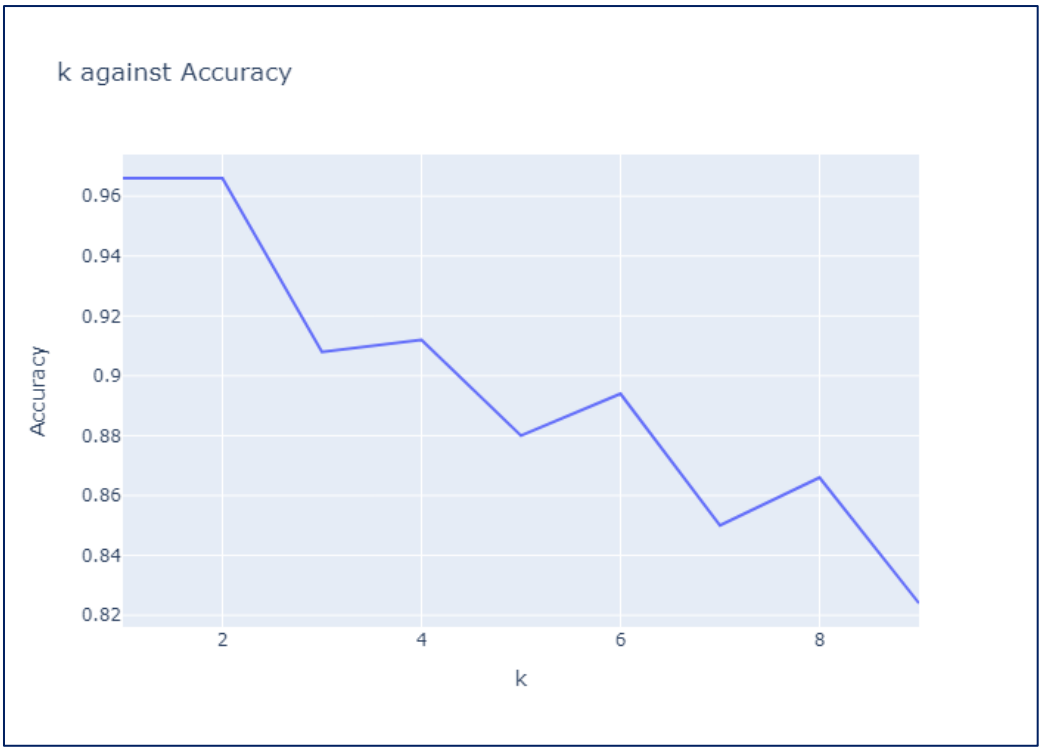

Fig 17. k Against Accuracy (k-Nearest Neighbors)

k-NN: Model Performance Metrics (for k = 2)

Model	Accuracy	Recall
SMOTE Model (Custom)	0.97	0.942
ADASYN Model (Custom)	0.97	0.942
sklearn (Original)	0.933	0.403

Table 3. k-NN Summary

Support Vector Machines

An extension of the standard model, soft margin SVM allows for some misclassification of points. The standard SVM algorithm aims to identify a hyperplane that can perfectly separate the two classes of data points, which may not always be possible due to non-linear separability or the presence of outliers. In such cases, the soft margin SVM algorithm introduces slack variables that allow some data points to be misclassified or placed within the margin boundary. By minimizing the sum of these slack variables, the soft margin SVM algorithm seeks to identify the hyperplane that can best separate the two classes while allowing for some degree of misclassification.

$$\begin{aligned} \underset{\mathbf{w},\ b,\ \zeta}{\text{minimize}} \quad & \|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\zeta_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0 \quad \forall i \in \{1,\dots,n\} \end{aligned}$$

The slack variable permits the points to be even on the other side of their classification but penalizes the function for any slack. C, the regularization constant, helps decide if the problem is a hard margin classifier or a soft margin classifier.

If C is large, then the objective becomes strict.

If C is small, then the objective becomes loose for a simpler solution.

Dual Form

The objective becomes simpler with the solving of the Lagrangian dual.

$$\begin{aligned} &\text{maximize } f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (\mathbf{x}_i^\mathsf{T}\mathbf{x}_j) y_j c_j, \\ &\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i. \end{aligned}$$

Kernel Tricks

Kernel methods handle high-dimensional data by mapping it to a higher-dimensional feature space using a kernel function. The kernel function calculates the similarity between two data points in the original space, and by working in a higher-dimensional space, it becomes easier to identify linear boundaries between data points that are not separable in the original space. This technique is known as the "kernel trick" and allows for efficient computation of non-linear decision boundaries.

Linear, polynomial, Gaussian, and sigmoid are a few examples of kernel tricks.

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}}.$$

Support Vector Machines: Model Performance Metrics

Model	Accuracy	Recall
Custom (Original)	0.902	0.242
SMOTE Model (Custom)	0.785	0.953
ADASYN Model (Custom)	0.789	0.953
sklearn (Original)	0.044	0.246

Table 4. SVM Summary

Varying Regularization Constant (C)

It is observed as the C increases, the accuracy increases till a breaking point and falls.

```
100%|██████████| 10000/10000 [02:01<00:00, 82.08it/s]
100%|██████████| 10000/10000 [01:42<00:00, 97.39it/s]
Soft-Margin SVM C = 2 | Precision = 0.386, Recall = 0.92, Accuracy = 0.831, F-score = 0.544
100%|██████████| 10000/10000 [01:19<00:00, 125.37it/s]
Soft-Margin SVM C = 3 | Precision = 0.427, Recall = 0.871, Accuracy = 0.858, F-score = 0.573
100%|██████████| 10000/10000 [01:17<00:00, 129.25it/s]
Soft-Margin SVM C = 4 | Precision = 0.446, Recall = 0.843, Accuracy = 0.868, F-score = 0.583
100%|██████████| 10000/10000 [01:13<00:00, 136.69it/s]
Soft-Margin SVM C = 5 | Precision = 0.457, Recall = 0.834, Accuracy = 0.874, F-score = 0.591
100%|██████████| 10000/10000 [01:09<00:00, 143.99it/s]
Soft-Margin SVM C = 6 | Precision = 0.239, Recall = 0.989, Accuracy = 0.654, F-score = 0.385
100%|██████████| 10000/10000 [01:15<00:00, 132.26it/s]
Soft-Margin SVM C = 7 | Precision = 0.236, Recall = 0.991, Accuracy = 0.648, F-score = 0.382
100%|██████████| 10000/10000 [01:14<00:00, 134.64it/s]
Soft-Margin SVM C = 8 | Precision = 0.233, Recall = 0.991, Accuracy = 0.641, F-score = 0.377
100%|██████████| 10000/10000 [01:11<00:00, 140.05it/s]
Soft-Margin SVM C = 9 | Precision = 0.23, Recall = 0.993, Accuracy = 0.635, F-score = 0.373
```

Fig 18. Varying C in SVM Soft-Margin

Results

k-NN (ADASYN and SMOTE oversampled datasets) provide the best accuracy and recall for the objective. Second in consideration will be the Logistic Regression models developed on the oversampled datasets considering their relatively fast capabilities. SVM is equally good in terms of classification. The results for the testing validation are presented below each model.

K-NN was developed on a k (nearest neighbors parameter) of 2 which showed the best accuracy for both ADASYN and SMOTE. sklearn model underperforms when compared to the custom-built classes.

Logistic Regression was developed on the oversampled datasets as well. The model takes three hyperparameters – learning rate (alpha), and epsilon (error threshold), and max iterations. The hyperparameters after fine-tuning performed best at LearningRate = 1e-4, Epsilon = 0.0001, and maxIteration = 10,000. sklearn version of Logistic Regression outperformed the custom models in terms of accuracy but was poor in recall.

Soft Margin Support Vector Machine (SVM) was modelled with the following parameters – regularization constant (C) of 1, after considering a trade-off between recall and accuracy. The optimizer function of the Lagrangian dual works with a learning rate of 0.00001, linear kernel function, and max iterations of 10,000.

Discussion

- The dataset cannot be directly used to train the model considering the imbalance in the target class of interest. So, oversampling or undersampling techniques need to be applied as part of pre-processing for a fair validation. Here, we have employed SMOTE and ADASYN.

- It is observed that SMOTE produces better results for the models than the baseline training dataset. It is recommended that the predictions are scored based on the SMOTE algorithm.

- Dimensionality reduction techniques such as Principal Component Analysis (PCA) were found to be less effective considering the number of components required to retain the information. Another downside is the lack of interpretability in the new z dimension.

- Macroscopic economic factors are a factor in the client decisions. Incorporating that information into the dataset will likely improve model performance.

- As an extension to this project, other models such as Decision Tree classifier can be employed that will navigate high-dimensional data and automatically perform feature selection without losing interpretability. Also, no assumptions are made on the dataset.

- Recall is the more important performance measure in this modelling since a false negative output results in the loss of business with a client.