# QBUS3820
# Machine Learning and Data Mining in Business
# Group Assignment

Tianying Sheng - 500527786
Chuang Chen - 500490424
Ally Migdol - 530291718

THE UNIVERSITY OF SYDNEY

# Contents

# 1 Introduction

This report aims to increase the marketing efficacy for two of our clients, a bank and a fashion store, by uncovering actionable insights from comprehensive data analyses. Companies employ various marketing strategies, including phone promotions and emails, to attract and retain customers. As a team of business analysts in a marketing consulting firm, we have leveraged the potential of two distinct marketing datasets, applied sophisticated predictive algorithms, and unearthed underlying patterns that will help our clients make informed and cost-effective business decisions in an ever-changing commercial world.

The bank's dataset, collected from historical marketing campaigns, incorporates customer demographics, economic indicators, and customer subscribing status to phone campaigns. Meanwhile, the fashion store's dataset records customer behaviors, spending habits, and responses to email marketing promotions. Through rigorous analysis of these datasets, we aim to establish an accurate machine learning model to predict future customer behaviors and provide data-driven recommendations in line with our loss scenario. Notably, due to the substantial imbalance between negative and positive responses in both datasets, F1 and AUC have been selected as robust evaluation metrics that work effectively for tackling such imbalanced problems.

We performed exploratory data analysis to identify preliminary patterns and bivariate relationships among predictors, which provides relevant information to our feature engineering and model building through interactive data visualization. A streamlined methodology was devised to automatically compare diverse encoding and transformation techniques thus accelerating the data-driven experimentation of different models. Additional approaches, such as merging sparse values and constructing binary indicators, were implemented to enhance the overall quality and reliability of the data.

The model building process was iterative, involving numerous trial and errors to identify potential algorithms with superior statistical performance. Logistic regression was chosen as the benchmark model for its simplicity, interpretability, and low computational demand. After a preliminary comparison, we selected the most promising candidate from each algorithm type, which was then fine-tuned to increase the likelihood of achieving a better model with optimised hyperparameters and the lowest possible loss for both clients.

The stack model emerged as the best one overall, minimizing cost while providing a high standard of predictive accuracy in line with our loss matrix. Through data mining, we extracted key insights into customer behaviors. Customers who engaged with previous campaigns tend to respond positively to more recent marketing promotions, this is true for bank and store. These findings demonstrate a degree of customer loyalty in both business scenarios, which might suggest the importance of developing customer retention strategies. Curiously, in the bank's dataset, the timing of campaign initiation also played a significant role, with those launched during stable economic periods yielding significantly better results than those held in unstable environments.

For the sake of reproducibility and consistency of the results, we set the random state to either 7 or 42. The random states differ in documents because the team cooperated in a distributed way, using different code documents on separate computers.

# 2 Problem formulation

The project takes into account two clients, a bank and a store, and aims to improve the marketing efficiency of both. This achieved by utilising decision theory, which informs the client for their best course of action. The decision theory assumes that the agent has a set of possible actions to choose from, and each possible action has costs and benefits that depend on the state of nature (Lecture 2). The state of nature for both our cases is whether the customer responds to the marketing campaign or not.

The overall goal of this report is not only to distinguish between a customer that would respond to the

marking campaign but also provide the businesses with insights into what section of the demographic they should target. Thus, when they do perform the marketing campaign, a larger portion of the people they contact will respond to the campaign compared to without the filtering.

Our first hypotheses is that people who have previously done business with the bank or the store will be more likely to respond to the marketing campaigns. Our second hypotheses is that ensemble model which combines multiple models will produce better prediction than using a single model.

## 2.1 Bank

The bank utilises telemarketing (phone calls) to reach potential customers, as per-call conversion rates are significantly higher than other channels. However, telemarketing requires additional resources such as more employees, training, and time to reach each potential client. Thus, the bank has to decide on their selection of participants to call within the campaign with more thought. The agent is the telemarketer who decides which customers to call or not. The state of nature is if the customer subscribes or not to the term deposit. Every customer called without interest in subscribing wastes the bank's resources, and by calling these customers, the bank may not call others who are willing to subscribe. The state of nature is if the customer would subscribe to the term deposit. Therefore, the bank needs an accurate model that will predict a customer's willingness to subscribe based on previous data to maximise profit. Given the goal of the project, we will solve the problem with supervised learning through its strong ability to estimate the probability of something unknown given a provided data set (Lecture 2). Then from the strongest model we will extract characteristics of customers that are more likely to subscribe to a term deposit or any other interesting features which will then help the bank make a more informed decision when selecting who to call.

### 2.1.1 Loss matrix

The loss matrix for the bank data set was determined by first calculating the average call length per customer which was found to be 4.30 minutes by averaging the durations column of the data set. The cost per customer was found by first finding the average hourly rate of outsourcing a telemarketer in Australia which was around \$40 per hour (Cloudtask.com, 2020). Then by multiplying the salary of the telemarketer with the percentage of time per hour a customer takes we get \$2.87 per customer. We also found that the median savings of an Australian in all ages was \$3559 according to Westpac, since it is unlikely that the customer will deposit all their savings into a term deposit, we assumed that the customer on average puts 80% of their savings into a term deposit if they choose to accept it which puts the amount deposited to be \$2847. According to the reserve bank of Australia, the net interest margin is 2%, therefore the bank makes approximately $2847 * 0.02 - 2.87 = \$54.07$. Therefore the loss for any customer that we don't call but that would have subscribed (False negative) is \$54.07 and the loss for calling a customer that does not subscribe (False positive) is \$2.87. The loss for correctly classifying is 0. Therefore the loss matrix is as follows when simplified ($\frac{54.07}{2.87} \approx \frac{19}{1}$):

| | | Predicted | |
|---|---|---|---|
| | | Not Subscribed | Subscribed |
| Actual | Not Subscribed | 0 | 1 |
| | Subscribed | 19 | 0 |

## 2.2 Store

The store client reaches customers through promotional Email campaigns. The agent, the email campaign system, has the situation of sending out promotional campaigns with the decision of which customers to send or not. Therefore, the state of nature is whether the customer would respond to the email campaign. However, not all customers sent the promotions will respond, and therefore the store avoids unnecessary costs by sending the campaign to uninterested consumers. To properly understand the costs and business gained from email campaigns, a classification model will predict whether a customer responded to the promotion. The goal is to, given a customer, classify whether

to send the promotional email campaign or not based on the notation of the customer responding to the promotion and to find characteristics of a potential customer.

### 2.2.1 Loss matrix

The loss matrix for the store data set took into account the average return on investment (ROI) from email marketing campaigns which is approximately $54 for every $1.50 spent (Kirsch, 2023). We also took into account acquisition costs such as costs for generating traffic, opt-in forms, and cost of content from the email campaign which added $1.35 per email sent (Editorial Team, 2023). Therefore, the loss of sending the email campaign to a non-responding customer is $2.85, and the loss of not sending the email to a responding customer is $54. Therefore, the loss matrix is as followed when simplified is ($\frac{54}{2.85} \approx \frac{19}{1}$):

| | | Predicted | |
|---|---|---|---|
| | | Not Responded | Responded |
| Actual | Not Responded | 0 | 1 |
| | Responded | 19 | 0 |

## 2.3 Metrics to determine success

Our proposed solution's success is determined by evaluating the total loss of each model through our carefully designed loss matrices. Initially, we established a baseline loss by considering the scenario where both the bank and store sent the marketing campaign to every individual on the list. The baseline losses amounted to 9780 for the bank and 5418 for the store.

To gauge the overall effectiveness of our models, we proceeded to calculate the loss for each model and compared it against the respective baseline. A model is considered successful if it achieves a lower loss than the baseline. Moreover, a lower loss value indicates a superior performance, highlighting the model's enhanced capabilities in minimizing our client's costs.

# 3 Exploratory Data Analysis

Exploratory data analysis (EDA) is the process of describing and visualising a dataset to understand it, identify patterns, detect issues, and check assumptions (Lecture 3). EDA gains insights and informs feature engineering and modelling. By exploring the relationships and predictors, the designed models can best accurately predict the clients' campaigns aligning with our business loss scenario.

## 3.1 Bank Data

**Data Error and Leakage**
Before modelling, the data set was first cleaned to fill in missing values and remove duplicate entries using the clean_df function which puts all the column headers in the same format, standardising missing values and removing duplicate rows. In order to avoid leakage, we examined each variable. Data leakage occurs when training a model that uses information unavailable for prediction in a production environment (Lecture 3). According to the data dictionary, the duration variable contains information that we are trying to predict. The duration of a call is unknown until after the call has ended with the customer, and by that point whether or not the customer subscribed will be known anyways. We also make our predictions before calling a customer so duration is not available during that time. Therefore, we decided to drop the duration column to decrease outliers and an over-optimistic estimation of generalisation performance. To reduce data leakage, the variable "campaign" was also removed. Since the campaign is still occurring, the number of ongoing contacts performed is part of the data we are trying to predict.

**Continuous Predictors**
Univariate distribution plots check for high skewness, multi-modality, high proportion of zeros, and

high kurtosis which may indicate non-normally distributed data. Dataprep showed that pdays (number of days passed by after the client was last contacted), emp_var_rate (employment variation rate), cons_price_idx (consumer confidence index), euribor3m (euribor 3 month rate), and nr_employed (number employed) each appear skewed (Figure B.i). In addition, age has outliers which will affect the performance of some machine learning models and therefore needs feature engineering.

**Bivariate Distribution**

Measure of dependence highlights the relationship between the continuous variables with the customer response rate. Regplot showed the bivariate distribution by combining a linear regression with a scatter plot to highlight the variables' relationships (Figure 3.i). Weak relationships include emp.var.rate, cons.price.idx, euribor3m, cons.con.idx, and age. They each show little impact on the subscription rate. On the contrary, pdays shows a strong linear relationship between the numbers of days passing and a decrease in subscriptions. As the number employed increases, the subscription rate decreases.

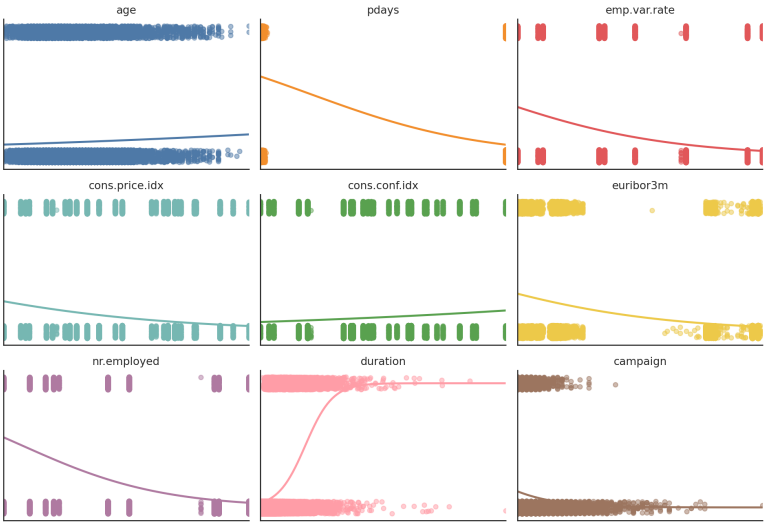



Figure 3.i: Bivariate Relationships

**Discrete and Categorical Predictors**

Percentage plots visualise the data as percentages of a whole to show the distribution and proportion of each categorical predictor ((Figure B.v)). The majority of customers are married, use cellular for contact compared to telephones, and are contacted for the first time in a marketing campaign. Comparing the distribution of the variables to stacked bar charts that take in account the percentage to subscribe shows that the majority groups may not necessarily result in more subscriptions. The stacked bar charts allow us to visualise the relationship between the client subscription and the remaining variables (Figure B.iii). While the month of May has the highest customers with 34%, it resulted in the lowest amount of subscriptions within the year (Figure B.iiig).

Customers also show a higher probability to subscribe after the bank previously contacted them more than two times. lthough the bank did not previously contact 85% of customers, contacted 12% one time previously, and nearly 3% for two times or more (Figure B.iih). The data show that 81% of customers do not have personal loans (Figure B.iia). However, in terms of the subscription rate, no loan customers had nearly the same likelihood to subscribe than customers with a loan or unknown (Figure B.iiia). Therefore, while the majority of respondents may fit in a certain category, it does not necessarily mean a higher subscription rate.

**Mutual Information**

Mutual information shows the average amount that two outcomes occur together compared to their outcome as independent variables (Zhang et al., 2021). It provides information not only about the linear relationship between the variables, but the non-linear relationships as well shared between any two random variables. Therefore, looking at the mutual information for the bank variables between

subscriptions, euribor_3m has the highest strength association of .0751 (Table 9). The variables cons_conf_idx, cons_price_idx, and nr_employed follow with close predictive power. On the contrary, housing, loan, and day of the week have the weakest relationships with scores less than .004 and .001. Variables such as housing, loan, and day of the weak provide very little information for the subscription outcome, and therefore may face less prioritisation in modelling, .

## 3.2 Store Data

**Data Error and Leakage**
The data set was also cleaned to fill in missing values and remove duplicate entries using the clean_df function which puts all the column headers in the same format, standardising missing values and removing duplicate rows. The number of marketing promotions on file, Promos, was also removed because it will lead to data leakage from the model predicting if the customer will respond to the email marketing campaigns.

**Continuous Predictors**
The store data consists of majority numerical predictors, resulting in 39 continuous variables. Dataprep stated high skewness in 30 of the variables with 13 variables having over 50% zeros, indicating the need of feature engineering before building certain linear models. After creating dummy indicators for variables with high percentages of zeros, dataprep no longer classified ccspend, gmp, fredays, hi, days, and classes as skewed.

**Discrete and Categorical Predictors**
The percentage plot showed key insights on the distribution of each predictor. At 67%, the majority of customers did not use coupons and approximately 17% used one (Figure B.va). The number of customers using more than two coupons decreases to less than 3%. Yet, when looking at the response success rate, according to the exploratory data analysis, as the number of used coupons increase, the probability of responding to promotion seems to significantly increase. However, when the number of coupons exceeded 12, the increasing rate slowed down, with some extreme patterns in categories that are too sparse due to a small sample size for high levels of coupons. Number of stores customers visit has a similar distribution, with the response rate increasing as the number of stores visited increases. Stores also have the majority of customers between one to three stores, but the pattern of response increasing shows prominent (Figure B.ivh). Mailed has a more uniformed distribution and also shows the pattern that an increase of mail sent to the customers, also increases their likelihood to respond to the campaigns(Figure B.ivd).

**Pearson Correlation Analysis and Mutual Information**
To analyse the relationship between the continuous variables, we conducted a Pearson correlation (Figure 3.ii). The test highlighted that there is a strong positive correlation between mon (Total net sales) and ccspend (Amount spent at CC store), fre (Number of purchase visits), styles (Total number of individual items purchased by the customer), and classes (Number of different product classes purchased). Number of purchase visits, amount spent at each store, and number of items purchased by the customer each directly impact the total net sales of the store, and therefore may arise potential multicollinearity issues for the linear model. For example, ltfreday (lifetime average of days between visits) and fre days (number of days between purchases) show very similar information as both focus on days between purchase and therefore had a higher correlation. There is also a strong negative relationship between markdown (Markdown percentage on customer purchases) and fredays (Number of days between purchases).

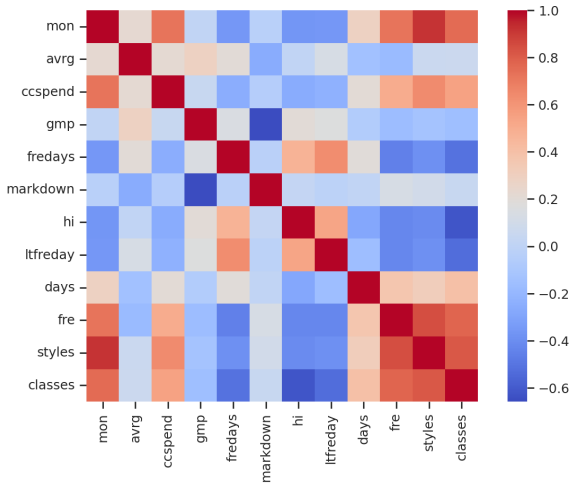



Figure 3.ii: Pearson Correlation

As previously stated, mutual information is an essential insight for classification as it provides information not only about the linear relationship between the variables, but all relationships shared between any two random variables. The mutual information score showed mon (total net sales), avg (average amount spent per visit), and ccspend (amount spent at CC store) are highly associated with customer responses (Table 10). With positive and large scores, each of the two outcomes occur much more frequently than they would compared to random chance and therefore may face higher prioritisation in the modelling.

## 4 Feature Engineering

Feature engineering is a crucial step in any machine learning project, as it transforms variables into formats better suited for various learning algorithms, ultimately enhancing prediction performance. During the initial stages, we implemented an iterative process of trial and error to identify the optimal combination of models and encoders. To achieve this efficiently, we designed two functions based on tutorial code that automatically compared different encodings and transformations. The first test_encodings() was dedicated to categorical feature imputation, while the second test_transformation() focused on numerical feature scaling.

In addition to these advanced feature engineering comparisons, we conducted basic data processing for each dataset, which contained cleaning, type inference, informative feature construction, merging sparse data, and imputing missing values. These engineering techniques were developed based on our comprehensive understanding of the two datasets, effectively preparing the features to adapt to varied models.

It is noteworthy that different learning algorithms possess distinct properties. Some are sensitive to outliers and skewness, requiring feature encoding as well as specific assumptions, as seen in logistic regression and other linear models. On the other hand, some algorithms are robust to monotone transformations, outliers, and missing values, such as tree-based models including decision trees, random forests, and boosting. Therefore, we implemented diverse feature engineering techniques tailored to the specific model and requirements.

Before we started, the two data sets from bank and fashion store clients were randomly split into training, validation, and testing sets by setting the random state to be 7, with respectively 64%, 16%, and 20% of the original data. This can ensure the reproducibility of the evaluations of our feature engineering and modelling.

### 4.1 Encoding Methodology

The first function is **test_encodings()**, which is designed to concurrently assess the performance of various categorical variable encoding techniques on a machine learning model. It accepts three arguments: algo, nominal, and features.

- algo is a learning algorithm that will be used to fit and predict the data.
- nominal is a list of categorical variables that will be tested.
- variables is a list of invariant base features to include in the model.

The function effectively performs a comparative analysis among six encoding methods: One-Hot, Target, Leave-One-Out, GLMM, CatBoost, and Ordinal encodings. Before calling the function, we transformed each categorical feature into distinct numerical columns in accordance with various encoders.

This approach starts by initializing an empty dictionary results_df_dic where the results for each encoding method will be stored. For each nominal predictor, it creates a DataFrame called results with rows for each encoding type and columns for each performance metric, including F1-score, AUC, Sensitivity, and so on. The function then iterates through each encoding method, generating new feature sets accordingly.

Each time, the model is fit using the training data and then used to make predictions on the validation set. Finally, the results table for each categorical variable is added to the results_df_dic dictionary with the variable's name as the key. The function then returns this dictionary containing all outputs. The train, valid, y_train, and y_valid were predefined by random splitting before the function is called.

At the end, the performance results were sorted by key metrics such as F1 scores or AUC, allowing us to identify the optimal combination for each learning algorithm. This automatic methodology significantly reduces the time spent on feature engineering, making rapid iterations through encoding, modelling, and evaluation processes achievable. We present an exemplary output of test_encodings() for education variable in logistic regression (Table 1). Since each time we only change one single variable, so the differences in statistic might be small but meaningful.

The second function, **test_transformations()**, is similar to the first one, despite a focus on continuous variables as opposed to categorical ones. For each learning algorithm, this streamlined approach enables us to evaluate a variety of transformations for continuous variables, including Logarithm, Yeo-Johnson, Box-Cox, Normalization, Standardization, with No Transformation serving as the baseline. It's important to note that some variables may contain negative values. Our methodology automatically detects these variables and excludes the Log and Box-Cox transformations - only accommodate positive values - for them.

This function also supports three parameters: algo, continuous, and variables. The changes from test_encodings() is the continuous parameter, which is a list of continuous variables to be tested. Following the similar process, each continuous variable is transformed into different formats and stored in the dataframes. We then execute the function and compare transformations for each algorithm. Some exemplary outputs are similar to the test_encodings() output (Table 1), except the encoders replaced by d transformations.

| Encoders | F1-score | AUC | Sensitivity | Precision | Specificity | Error Rate | Cross-entropy |
|---|---|---|---|---|---|---|---|
| One-Hot encoding | 0.2394 | 0.6993 | 0.9473 | 0.1370 | 0.1541 | 0.7474 | 0.3322 |
| GLMM encoder | 0.2365 | 0.6996 | 0.9628 | 0.1348 | 0.1240 | 0.7719 | 0.3322 |
| Target encoder | 0.2355 | 0.6976 | 0.9643 | 0.1341 | 0.1174 | 0.7774 | 0.3322 |
| Leave-one-out encoder | 0.2355 | 0.6976 | 0.9643 | 0.1341 | 0.1174 | 0.7774 | 0.3322 |
| CatBoost encoder | 0.2355 | 0.6976 | 0.9643 | 0.1341 | 0.1174 | 0.7774 | 0.3322 |
| Ordinal encoder | 0.2355 | 0.6954 | 0.9643 | 0.1341 | 0.1174 | 0.7774 | 0.3323 |

Table 1: Example of test_encodings() on education predictor for Logistic Regression

## 4.2 Bank Data

After inspecting the bank dataset, we observed that the majority of predictors were categorical. The response variable was converted to a binary format, with 1 representing yes. Each feature was grouped into types, including continuous, nominal, binary, discrete, and ordinal.

### 4.2.1 Categorical Predictors

Beyond the automatic test_encodings() method, we applied various imputation techniques, such as creating dummy indicators and merging sparse categories. For instance, a significant proportion of 999 values in pdays - the number of days that passed by after the client was last contacted from a previous campaign. To address this, we created a binary indicator, pdays_999, to denote whether an observation contains 999 in pdays. Similarly, for the previous predictor, we observed that values greater than 3 were sparse, contributing less than 1% of total observations. To create a representative feature, we combined all values greater than 3 into a single group 3. We encoded a new ordinal feature for education, based on domain knowledge. One-Hot encoding is for all binary variables, while for linear models such as logit models, we applied dummy encoding to avoid multicollinearity. Furthermore, at the experimentation stage, since almost all learning algorithms require numerical predictors, we applied One-Hot encoding to all the other nominal variables (dummy for linear models). This approach expedited the building up of benchmark models, allowing us to rapidly filter out the best candidates before hyperparameter optimization.

In contrast, only a small fraction of variables in our bank data were continuous. Since different learning algorithms have different standards of data format, we will discuss detailed feature engineering approaches for each respective model in the Methodology section.

## 4.3 Store Data

For the store dataset, most predictors are numerical, with one string type binary variable - valid phone or not (valphon). During the basic cleaning process, we simply encoded it into an integer dummy variable and converted all column names to lowercase. Notably, despite the numerical data type of the Microvision lifestyle cluster type (clustype) predictor, it should be treated as a categorical feature. This is because the column contains multiple distinct values representing separate categories or clusters, rather than a meaningful continuous range. Therefore, we evaluated different categorical encoders for the clustype predictor in various models using the auto function we designed.

Upon inspecting the data, we classified each independent variable into different groups: continuous, discrete, ordinal, and nominal.

### 4.3.1 Continuous Predictors

During the exploratory data analysis, we observed that most continuous predictors were skewed, with a high proportion of 0s present. To address this issue and facilitate future analysis for non-tree-based models like logistic regression and neural networks, we created dummy indicators to preserve the information about 0s in the original columns. For columns with a 0 proportion less than 25%, we imputed the 0s using the medians of non-zero values, as most non-zero values were also skewed. This approach allowed us to maintain the original information of each covariate while ensuring data quality for auto-comparison between different continuous data transformations in the store data.

However, some predictors contained a large number of 0s (i.e., greater than 25%). In these cases, we discarded the original predictors and retained only the respective dummy indicators. This ensured we did not introduce biases or distortions in the data, whether subjectively or inadvertently. For the rest of the continuous, we simply deployed test_transformations() for each model.

### 4.3.2 Discrete and Ordinal Predictors

In regards to discrete predictors, we treated those with a large number of distinct values (more than 30), such as the number of purchase visits (fre) and the number of days a customer has been on file (days), as continuous variables since they represent a continuous range. For predictors with fewer distinct values, mostly ordinal, we merged categories that were too sparse to reveal informative relationships or patterns.

For instance, our exploratory data analysis indicated that as the number of coupons increased, the likelihood of a response to a promotion also increased significantly. However, once the number of coupons exceeded 12, the increasing rate slowed down, with some extreme patterns observed in the sparse categories. Our team considered these small groups to be non-representative and combined them with the 12-coupon group. We employed a similar approach for the pc_calc20, stores, mailed, and responded predictors, merging the sparse as needed.

Plus, although clustype was transformed into a numeric value, each value represents a distinct category. In this case, we regarded clustype as a nominal variable and compared various feature encoders for model implementation.

Despite the data issues mentioned earlier, it is worth noting that tree-based learning algorithms inherently handle outliers and missing values. They are insensitive to monotonic transformations and can efficaciously recognize the meaning of rank predictors, which frequently appear in store data. Given these advantageous properties, our team has an optimistic expectation that tree models, such as random forest and boosting, will perform well in this context. Therefore, in addition to the encoding techniques used for other algorithms like logistic regression and neural networks, we have also prepared a version of the data with basic imputations specifically for tree-based models.

# 5 Methodology

## 5.1 Evaluation metrics

In the exploratory data analysis, we noted the number of negative responses significantly outweighed the positive responses in the two datasets. Given this imbalance, using accuracy as an evaluation metric would be inappropriate. The score could be misleadingly high if the model merely categorises all observations to the majority class. Although certain techniques, such as reweighting each class or oversampling, might alleviate this issue, they could also result in distortions in the class probability estimates. Hence, we opted for alternative evaluation measures like the F1-score and AUC to more correctly and effectively assess model performance.

The F1 score is a strong metric for imbalanced data. In our business scenario, false negatives (missing subscribed customers) have a higher cost than false positives (targeting unsubscribed customers). Admittedly, sensitivity should be taken into account, however, it could potentially result in the model becoming overly sensitive to false negatives and overlooking the imbalanced data. Thus, the F1 score emerges as a more suitable evaluation metric, as it offers dependable measures for both minority and majority classes while balancing the trade-off between sensitivity (recall) and precision. This allows our model to generalize well, maintain high predictive accuracy, and effectively address the imbalance.

Another robust evaluation metric is AUC (or ROC-AUC), the Area Under the Receiver Operating Characteristic Curve, which measures the model's ability to balance the true positive rate (sensitivity) and false positive rate (specificity) (Tutorial 2). Although superior to accuracy in cases of imbalance, AUC may present some limitations where specificity for heavily imbalanced datasets is pulled down due to a large number of true negatives (Czakon, 2023). Therefore, theoretically, we should prioritize the F1 score while taking AUC as the second-place evaluation metric. Whereas, during the practical experimentation phase, our team assessed algorithms on both the F1 score and AUC, in response to different situations. We also fine-tuned models in terms of cross-validation and various scoring metrics, including negative log loss (log-likelihood), F1 score, and AUC, and finally evaluated each by validation metrics.

In doing so, we increased the likelihood of yielding models with higher statistical performance (high F1, high AUC, or low loss), aligning with our ultimate goal: identifying the best model that exhibits outstanding predictive accuracy while minimizing the cost for our real-world clients.

## 5.2 Bank Dataset

### 5.2.1 Best Linear Model

**LogisticGAM & Logistic Regression**

The best linear model for the bank turned out to be the LogisticGAM (Generalised Additive Model), which permits the training of non-linear functions for each predictor, offering both high predictive performance and interpretability (Lecture 6). This outcome was surprising because our initial objective was to identify the optimal logistic regression model.

After obtaining the best logistic regression model with Ridge regularization ($L2$ penalty) using standardized data, we attempted to fit the top 6 variables with the highest absolute coefficient values into the GAM model. The reason behind this is that incorporating all predictors simultaneously into GAM models is unrealistic. Manual adjustment of each smoother is not only time-consuming, but it may also not produce competitive results when compared to more sophisticated learning algorithms such as boosting, which can accommodate all predictors at once. However, the LogisticGAM outperformed our expectations, emerging as the best linear model with the smallest loss in the specific business context. We will next start by elaborating on the process of building up the best logistic model.

**Checking Assumptions**

Logistic regression utilizes the principle of maximum likelihood for training probabilistic models for classification. Unlike tree-based models, logistic regression, a parametric linear model, has rigorous requirements of inductive bias, assuming binary response, linearity between predictors and log odds, and the absence of multicollinearity.

In the case of the bank dataset, checking for multicollinearity is relatively straightforward due to the majority of categorical variables and the seeming irrelevant patterns of the remaining continuous predictors, as indicated by the data dictionary and correlation results in EDA. To avoid multicollinearity, we specifically excluded the One-Hot method while applying the test_encodings() function to identify the optimal encoding combinations, for the initialised logistic regression model (Table 2). Plus, the multicollinearity can be alleviated by implementing $L1$ or $L2$ regularisations.

Linearity assumptions may be violated in the dataset. For the remaining 6 continuous predictors (see Figure 3.i bivariate plot), we noticed that a weak linear relationship exists between each continuous predictor and the response. To address this issue, we employed the auto test_transformations() function on each continuous variable in order to identify the best transformers (Table 2). Despite these transformations, some degree of nonlinearity persists, and in such cases, we selected the option with the highest statistical performance, an aspect greater explained in the limitation section. Considering that we will later standardize the data for Lasso and Ridge regularizations, we only compared the Log, Box-cox, Yeo-Johnson, and no transformation in this instance. It is important to note that some variables may contain negative values. Our methodology automatically identifies these variables and excludes the Log and Box-Cox transformations for them.

| Nominal | AUC | F1-score | Continuous | AUC | F1-score |
|---|---|---|---|---|---|
| job | GLMM | GLMM | age | No transform | No transform |
| marital | GLMM | GLMM | emp_var_rate | No transform | Yeo-Johnson |
| education | GLMM | GLMM | cons_price_idx | Log | Log |
| month | Leave-One-Out | GLMM | cons_conf_idx | No transformation | Yeo-Johnson |
| day_of_the_week | GLMM | GLMM | euribor_3m | Log | No transform |
| | | | nr_employed | Log | Yeo-Johnson |

Table 2: Bank - Best encoders/transformations for Logistic Regression

Per with the feature engineering results for the initialised model, we started our iterative experiment covering a broad range of possible encoding-transformation combinations, attempting to satisfy the assumptions and data standard for logistic regression, as well as provide solid results to help the bank client make decisions.

**Trial and Error**
Training linear logit models demand less computational cost compared to more complex algorithms. We tuned the inverse regularisation strengths within a logarithmic scale (Cs = 50), and employed cross-validation to assess a vast dimension of parameters. For instance, we learned the models across original, normalized, and standardized training data to test the impact of scaling on model performance. We compared different optimization techniques (solvers), such as 'newton-cg', a variant of Newton's method utilizing second-order optimization with fast convergence and higher computational cost, and 'saga', a modification of first-order Stochastic Gradient Descent with computation efficiency (Lecture 5). We also tuned hyperparameters on varying scoring methods, including F1, AUC, and negative log loss. The exploration of a diverse set of possibilities was inherently fun. The logistic regression model, fine-tuned by minimizing the negative log loss via adding a $L2$ penalty term, turned out with the best statistical validation result. We then instinctively plug the top 6 standardised predictors with the largest absolute coefficient values in the LogisticGAM. To our surprise, the additive model emerged as the champion of bank predicting in the linear family.

**Hyperparameter Optimization**
We established the model using the pyGAM API, which supports built-in smoothing splines with a predetermined number of knots and uses $L2$ regularization to penalize the model's roughness (Tutorial 6). As we mentioned earlier, the linearity assumption of our logit models might be violated. The LogisticGAM model overcomes this problem as it allows each basis function to be nonlinear and the algorithm performs well for individual terms. We standardized the predictors with transformations from the previous logistic model, preparing them for tuning regularization hyperparameters on each basis function.

Initially, we built the model with 5 linear terms and 1 factor term, and tuned each model by maximizing the log-likelihood using TPE Bayesian optimization, as errors appeared when implementing the Generalized Cross-Validation (GCV) score. Allocating a timeout of 120 seconds is sufficient to identify a good configuration in this instance. After fitting the initial version of the GAM, we removed two variables: *nr_employedLog* and *euribor_3mLog*. We set the significance level to be $\alpha = 0.05$. The first had a p-value exceeding 0.1, while the second had a p-value higher than 0.05 but lower than 0.1, we did not reject the null hypotheses ($H_0$) that the coefficients of these two variables are insignificant.

With a maximum of 2 smoothing terms $s$, numerous trials and errors were performed on different smoothers for each predictor to prevent overfitting. As a result, we achieved the optimal combination of basis functions that provides high performance in terms of F1 score and AUC, along with reasonable confidence intervals in the plots of each partial dependent function, see appendix Figure B.viii. Competing with the best logit model, the fine-tuned LogisticGAM displayed distinguished statistics and lower loss. As a result, we conclude that the best linear function is the LogisticGAM Table 3.

| Feature Function | Lambda | Rank | p-value |
|---|---|---|---|
| l(*emp_var_rate*) | [37.871] | 1 | $2.80e-04$ |
| s(*cons_price_idxLog*) | [0.0001] | 20 | $0.00e+00$ |
| s(*monthGLMM*) | [0.0001] | 20 | $0.00e+00$ |
| f(*contact_telephone*) | [0.0151] | 2 | $1.11e-16$ |
| intercept | | 1 | $2.52e-01$ |

Table 3: LogisticGAM - Best linear model for bank

### 5.2.2 Best Non-Linear Model

**Boosting**

The best non-linear model for the bank dataset is the Light Gradient Boosting Machine Booster (LightGBM Booster). In contrast to linear models, gradient boosting, a nonparametric model often deploying small decision trees as weak learners, performs exceptionally well in predictions. This can be attributed to its flexibility and advantageous tree properties, such as insensitivity to monotone transformations and robustness to outliers (Lecture 7). It can automatically handle missing values, regularize parameters, and innately support categorical features. These advantages enable boosting families to excel with various data types without requiring extensive effort for feature encoding or scaling. Nevertheless, with the goal of identifying the best model for accurately predicting customer behavior for our bank client, we still compared different encoding methods for nominal features, including One-Hot, Target, Leave-One-Out, GLMM, CatBoost, and Ordinal, to find the best combination of boosting models and encoding techniques.

**Trial and Error**

During the experimentation phase, we conducted both vertical and horizontal comparisons of different models and their corresponding feature encoders. We began by exploring the best boosting models from the boost family, using pre-set hyperparameters and one-hot encodings to construct each benchmark boost model. As a result (see Table 11 in appendix), XGBoost and LightGBM emerged as robust boost candidates with the top 2 high F1 scores.

To identify the optimal encoders for these two candidates, we deployed the auto encoding function defined above to compare varied transformations of categorical variables. The optimal encoders (Table 4) for each model are:

| Features | XGBoost | LightGBM | LightGBM 2nd |
|---|---|---|---|
| job | CatBoost | One-Hot | One-Hot |
| Marital | CatBoost | CatBoost | GLMM |
| Education | One-Hot | Ordinal | Ordinal |
| Month | One-Hot | GLMM | One-Hot |
| Day of the Week | One-Hot | One-Hot | CatBoost |

Table 4: Best encoders for each boosting model

We then incorporated the encoded features into the two models, tuning each model's hyperparameters by setting a timeout of 3600 seconds (1 hour) with about 100 trials completed. We only tuned each candidate for an hour due to the limited GPU resources available in Colab. In addition to horizontal encoding comparison, we vertically evaluated Booster and Classifier models with the same tuning settings. The former requires the dataset to be converted into LightGBM or XGBoost data format, whereas the latter does not. As a result, the best candidate model is the LightGBM Booster, which performs highly efficient leaf-wise growth. We again tested various feature encoders for the LightGBM Booster. Rather than using pre-set hyperparameters, we employed the tuned hyperparameters from previous LightGBM Booster tuning results in the evaluation of different encoders. Finally, after iterative trials and errors, we identified the most effective combination (see **Table** 5.1.2.2) and proceeded to the next step in detail - hyperparameter optimization.

**Hyperparameter Optimization**

Optimizing the LightGBM Booster is straightforward. We adapted the code from Tutorial 8, using the TPESampler with a fixed seed of 7 from the Optuna library to optimize the model for approximately 660 trials within a time budget of $10,000$ seconds. Since the goal is to classify customer behaviors and minimize the bank's loss, the objective function was set to minimize binary logarithmic loss using gradient-boosted decision trees (GBDT) as the weak learners.

According to the lecture, the most important hyperparameters for boosting are the number of trees, learning rate, and maximum tree depth. They correspond to num_boost_round, learning_rate, and

num_leaves in the initial configuration. We employed 10-fold cross-validation.

- During cross-validation, we specified the maximum number of trees/boosting iterations to 1000 (num_boost_round = 1000) and applied early stopping (50 rounds) to the training if there was little improvement in the validation error. This approach efficiently saves computational resources and prevents overfitting and overly complex LightGBM models.
- By fixing the learning rate at 0.01, we increased the shrinkage of predictions from previous trees, i.e., giving less weight to the trees already present in the model, and allowing more room for additional trees to better fit the data. According to tutorial 8, a useful trick for tuning boosting models is to keep the learning rate fixed at a lower level, such as 0.01 in this case, which enables more weak learners for improved performance. However, we also need to consider the computational cost, so initially, a higher learning rate was set to speed up the hyperparameter optimization process and then reduced to 0.01 for the final boosting candidate.
- Number of leaves was set to range from 2 to 64. As each weak learner is a binary decision tree, the depth of each tree is set to range from 1 to 8 ($log_2 2$ to $log_2 64$). Tuning this parameter prevents overfitting because boosting additively combines weak models, which should ideally be non-complex weak learners, with a depth less than or equal to 8.
- Regarding $L1$ and $L2$ regularizations, lambda_l1 and lambda_l2 were tuned within an expanding range. By adding a penalty term to the loss function, they regularize the model and balance the trade-off between generalization and complexity, i.e. achieving good model performance while preventing overfitting.
- Other parameters such as feature_fraction (fraction of features to be randomly selected for each tree), bagging_fraction (fraction of data to be used for each bagging iteration), and min_data_in_leaf (minimum number of samples required to be in a leaf) were also considered to further improve the optimization result.

Eventually, we obtained the best parameters see Table B. By comparing the LightGBM Booster with other types of models using the validation set, we concluded that the best non-linear model for bank clients is the LightGBM Booster.

### 5.2.3 Model Stack

To determine the best model stack for the bank dataset, firstly the best encodings of the features for each model displayed in the result table (Table 5) were found using the methodology stated in the encoding methodology section of the report above. Then after the best encodings are found for each model, the individual models have their hyperparameters tuned using Bayesian hyperparameter tuning with the Optuna python package. The number of trials and the timeout for tuning each model differs as through trial and error we have found that some models require more time to tune and some require more trials to reach an optimal solution which is why we adjusted them accordingly instead of setting them all to be the maximum to save time and computer resources. Some models that did not have their hyperparameters tuned were also added into the model such as untuned decision tree, untuned random forest etc. We added these models because as seen in bagging, weak models together can also lead to accurate predictions.

The set of models that could be included in the model stack is as follows: [**random forest, random forest untuned, decision tree, decision tree untuned, bagging, bagging untuned, gradient boosting, histogram-based gradient boosting, catBoost, LightGBM, XGBoost, logistic regression**].

**The process for creating the model stack is as follows:**

1. Select a k-fold split on the training data.
2. Select n models from our list of models

3. Select 1 model to be the meta model from our list of models
4. For each of the k folds:
   (a) Evaluate model on the training data of the fold and store the prediction probabilities made as a column vector
   (b) Convert the probabilities into a prediction of either class 0 or 1 with our decision threshold of $\frac{1}{20}$.
5. Append all the prediction column vectors together into a matrix called meta_x.
6. Train the meta model with the data meta_x, test_y as input.

**The process for making predictions with the model stack is as follows:**

1. Pass each of the n base models the testing data and they will produce an array of probabilities for class 1 since it is a binary classification problem.
2. Convert the probabilities into a prediction of either class 0 or 1 with our decision threshold of $\frac{1}{20}$.
3. Concatenate the prediction as column vectors into a single matrix.
4. Pass that matrix into the meta model which will then produce the probabilities of being in class 1.
5. Convert the probabilities into an actual prediction using the decision threshold of $\frac{1}{20}$.
6. Calculate the F1-score on the probabilities.

**Hyperparameter Optimization**

The only hyperparameters to optimize for the model stack is which models go into the model stack and which model will be the meta model. There is no need to tune the individual models here as they are already tuned or are purposely weak models such as the untuned decision tree. The optimal model stack is found by using Bayesian tuning with Optuna, we let Optuna suggest the number of folds for the k-fold splitting, which models go into the stack, and which model will be the meta model. Then we build the model stack based on the suggestions and let Optuna repeat this process and tune according to the f1-score over 200 trials and an individual timeout of 50000 seconds per trial.

The best models that are chosen by Optuna to be in the model stack are: **[logistic regression, lightGBM, catBoost, bagging untuned, random forest untuned, decision tree untuned, XGBoost]** and the meta model is **gradient boosting**.

## 5.3 Fashion Store Dataset

### 5.3.1 Linear Model

The best linear model for the fashion store data is logistic regression with Lasso ($L1$) regularisation, which imposed shrinkages on the absolute values of coefficients, setting insignificant ones to zero, and inducing sparsity (Lecture 4). In contrast to the bank data, the fashion store dataset contains many numerical predictors that are highly correlated with each other, suggesting multicollinear issues, according to the data dictionary, exploratory data analysis, and Pearson correlation test. Despite standardizing the data and applying regularization on estimators to mitigate this issue, logit models still suffer from severe multicollinearity, which can lead to unreliable inference regarding the impact of individual predictors on the response variable and overfitting in the store dataset.

Aware of this, the team used the Variance Inflation Factor (VIF) to detect and screen out predictors with severe multicollinearity (Table 12). For example, the predictor gross margin percentage (gmp) had a VIF of 43.72, indicating high correlations with many other predictors, so it was removed. For predictors that were correlated with each other, such as total net sales (mon) and the total number

of individual items purchased (styles), with respective VIF of 26.47 and 20.03, a new column was created by calculating the average net sales per item for each customer. This approach preserved the information while mitigating multicollinearity and overfitting.

The remaining steps in the linear modeling process were similar to those used for the bank data. The linearity assumption was confirmed by examining bivariate plots (see **Appendix** EDA), as many continuous predictors demonstrated good linear patterns with the log odds of the response. Test_encodings() and test_transformations() were also conducted to automatically compare different feature engineering methods. The best encoders and transformations for each predictor can be found in the **appendix**.

LogisticGAM hyperparameter optimization was also performed. However, in this case, logistic regression with Lasso ($L1$) regularization emerged as the best model, demonstrating a better F1 score and AUC in the validation results, as well as a lower loss for store clients in our real-world business scenario.

### 5.3.2 Non-Linear Model

Boosting, again, emerges as our best non-linear model. For the bank dataset, the champion is LIGHTGBM Booster, while for the fashion store, it's lightGBM Classifier. The modeling process is similar to before, but there are some notable differences. All features in the store dataset are numerical and do not require pre-transformation. This simplifies the feature engineering process for boosting models, as they are robust to outliers, missing values, and insensitive to monotonic transformations. Essentially, we do not need to perform superfluous feature engineering for the store data when using boosting models.

Whereas, the clustype variable, even though it has been transformed into an ordinal number, lacks rank and range meaning because each number represents a category. To enhance the learning accuracy, we specify clustype as a categorical variable for some boost models.

Next, we compared benchmark models to identify the best candidates based on high F1 or AUC scores. Subsequently, we performed hyperparameter optimization for each candidate, setting a timeout of 3600 seconds to increase the likelihood of obtaining more competitive models. In the end, the LightGBM Classifier achieved the lowest loss and turned out as our best non-linear model for the store dataset.

### 5.3.3 Model Stack

For the best model stack in the store data set, the methodology we applied was the same as in the bank data set. The best models that were chosen by Optuna to be in the model stack are: [**histogram-based boosting, random forest untuned, XGBoost, logistic regression, CatBoost, decision tree, LightGBM**] and the meta model was **LightGMB classifier**.

## 5.4 Mulit-layer preceptron

For MLP we first built a simple MLP consisting of 3 hidden layers with 40 units in each and the activation functions being Relu between the layers and sigmoid for the output layer. We then tested various encodings of nominal variables with this simple MLP to determine which encoding works best for which variable. We then used Optuna to optimise the layers, the number of units in each hidden layer, and the activation functions. Throughout our testing, we found that the optimisation result tends to be inconsistent which is a result of the inconsistency when training the MLP. Sometimes we would get decent predictions on par with the other models, other times the predictions will be very poor. This is on the basis that we don't change any of the variables in the model and just retrain it. This inconsistency is due to the nature of GPU training as stated in the Pytorch documentation. There was no real way around this obstacle as it was just too time consuming to train on the CPU, even though GPU sped up the process significantly it was still very time consuming just to train the model once with 1000 epochs. Due to the nature of this inconsistency, we tried to optimise the model a few times with Optuna with the epochs set to 200 for each training cycle and then picked the best

optimisation results. The results we obtained we sub-optimal and worse than the other best models, therefore we decided not to include the detailed MLP methodology in this section.

# 6 Results

Note: No model mean that we predicted everything to be class 1 which is equivalent to the bank and the store sending everyone the marketing campaign without filtering.

## 6.1 Bank Dataset

| Model | Precision | Sensitivity | Specificity | Error Rate | AUC | F1-score | Loss Matrix |
|---|---|---|---|---|---|---|---|
| No Model | 0.1213 | 1 | 0 | 0.8787 | 0.5 | 0.2164 | 9780 |
| Logistic Regression (L2) | 0.1627 | 0.8961 | 0.3462 | 0.5920 | 0.7824 | 0.2749 | 8860 |
| Logistic GAM | 0.1682 | 0.8667 | 0.3923 | 0.5487 | 0.7778 | 0.2818 | 8756 |
| LightGBM boosting | 0.1760 | 0.8790 | 0.4310 | 0.5150 | 0.8010 | 0.2930 | 8794 |
| XGBoost | 0.1720 | 0.8890 | 0.4070 | 0.5340 | 0.7990 | 0.2880 | 8645 |
| Decision Tree | 0.1811 | 0.8333 | 0.4799 | 0.4773 | 0.7794 | 0.2975 | 9362 |
| Random Forest | 0.1625 | 0.9074 | 0.3543 | 0.5786 | 0.8004 | 0.2756 | 8690 |
| Bagging (Decision Tree) | 0.1223 | 0.9985 | 0.0109 | 0.8693 | 0.7911 | 0.2302 | 9711 |
| Model stack | 0.1707 | 0.8852 | 0.4063 | 0.5356 | 0.8057 | 0.2862 | 8620 |
| MLP | 0.1655 | 0.8970 | 0.3757 | 0.5611 | 0.7947 | 0.2795 | 8747 |

Table 5: Validation accuracy for Bank

## 6.2 Store Dataset

| Model | Precision | Sensitivity | Specificity | Error Rate | AUC | F1-score | Loss Matrix |
|---|---|---|---|---|---|---|---|
| No Model | 0.1694 | 1 | 0 | 0.8305 | 0.5 | 0.2891 | 5418 |
| Logistic Regression (L1) | 0.2299 | 0.9540 | 0.3386 | 0.5559 | 0.8033 | 0.3705 | 4813 |
| Logistic GAM | 0.1655 | 0.8682 | 0.3792 | 0.5401 | 0.7818 | 0.3753 | 8756 |
| LightGBM Classifier | 0.2640 | 0.9710 | 0.4480 | 0.4640 | 0.8530 | 0.4150 | 3601 |
| XGBoost | 0.2710 | 0.9600 | 0.4740 | 0.4440 | 0.8530 | 0.4230 | 3686 |
| Decision Tree | 0.2861 | 0.8688 | 0.5580 | 0.3894 | 0.8026 | 0.4460 | 5150 |
| Random Forest | 0.2666 | 0.9602 | 0.4612 | 0.4542 | 0.8522 | 0.4173 | 3755 |
| Bagging (Decision Tree) | 0.2510 | 0.9448 | 0.4249 | 0.4871 | 0.8387 | 0.4157 | 4275 |
| Model stack | 0.1582 | 0.9200 | 0.3240 | 0.6037 | 0.8533 | 0.4300 | 3712 |
| MLP | 0.2623 | 0.9584 | 0.4502 | 0.4637 | 0.8457 | 0.4118 | 3853 |

Table 6: Validation accuracy for Store

### 6.2.1 Discussion and Benchmark Model

The logistic regression model is selected to be our benchmark model. This model was selected because of three main features, its interpretability, its low computational cost compared to all the other models and the ease in which it is to implement. From the result tables we can see that the best linear model is the logistic GAM, the best non-linear model is the LightGBM, the best model stack and model average is the model stack with a combination of the following models: [**logistic regression, lightGBM, catBoost, bagging untuned, random forest untuned, decision tree untuned, XGBoost**] and the meta model is **gradient boosting**. These models are again the best for the store data set. The reason random forest or model stack is not the best non-linear model is because our team grouped those under model stack due to their nature of combining the predictions from multiple models. The best model overall is the model stack for both the bank and the store data set.

The reason why model stack works so well is because it is a combination of all the best single models we have, and by combining them into a stack we can leverage the different architectures of different models and therefore capture a wider range of patterns and improve the overall generalisation ability of the model. The combination of the different models also leverages the strength of each model, where

one model makes a mistake other models may perform better and the meta model might detect this and produce better predictions. The model stack is also less sensitive to noise and outliers in data.

### 6.2.2 Limitations

Although we fine-tuned the hyperparameters of all models using either 5-fold or 10-fold cross-validation, each model was measured by only a single validation set. This was primarily because certain models, such as the LightGBM Booster that requires a specialized data format, presented difficulties in adapting the code for cross-validation, specifically using the cross_val_score() function. We sought to preserve consistency in the evaluation process across all models and ensure result comparability, and thus, the evaluation was based on a single validation set. However, this approach might not be as precise as that achieved via cross-validation, which averages results across each validation fold, thereby providing a more robust measure. Therefore, the outcome from a single validation set might be skewed due to differing class distributions following the random splitting of train, validation, and test sets.

- No systematic methodology for variable selection for logistic regression and GAM
- Fixed threshold tau & Not calibrated probability
- logistic linearity for bank & multicolinearity and loss information for store

## 7 Data Mining

The metric we used to interpret feature importance is the SHAP plot and the feature importance plot. SHAP stands for Shapley Additive explanations which is a game theory approach for explaining the output of machine learning models (Lundberg, 2017). The more positive the SHAP value the more influence the feature has on predicting the 1 class, and the more negative the SHAP value the more influence it has on predicting the 0 class.

### 7.1 Bank

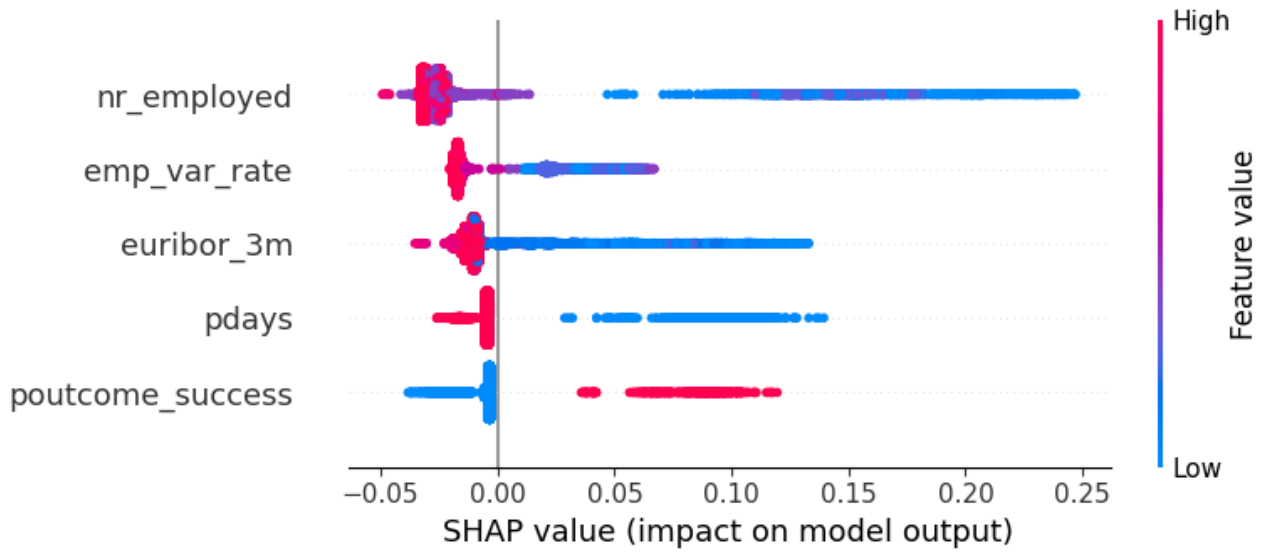



Figure 7.i: SHAP plot of random forest on bank data set

In the SHAP plot above we have picked out the features that have the most impact on the class predictions. The full SHAP plot is available in the appendix at Figure B.vi. From the above SHAP plot, the most influential feature is the number of people employed in the bank (nr_employed), it is curious to us that customers appear to be more willing to subscribe to banks with fewer employees

and are more likely to not subscribe if the bank had a high number of employees. The employment variation rate (emp_var_rate) is another indicator of whether the customer will subscribe or not, when the employment variation is low the customer shows more willingness to subscribe and when the employment variation rate is high the customer leaning more towards not subscribing. This is logical because the employment variation rate is usually an indication of the current economy, high employment variation rates typically indicate large layoffs and hires which might correlate to a dip in the economy when people are less confident to commit to a term deposit. Whether the customer previously subscribed (poutcome_success) is another indicator of whether a customer will subscribe or not, when poutcome_success is a high value of 1 meaning they previously subscribed, they tend be more confident to subscribe again to a term deposit versus when they have not subscribed before which leave them less confident and more likely to say no to a term deposit. Another feature we like to mention is the 3 month Euro Interbank Offered Rate (euribor_3m) which is the interest rate that Eurozone banks borrow money from each other. This indicator shows that with higher euribor_3m customers are more likely to decline a term deposit and with low euribor_3m customers are more likely to subscribe, this is most likely because, with high euribor_3m, the banks will also raise the interest rate on funds they lend to their customers such as mortgages. Thus during a time with high interest rates people are less likely to subscribe to a term deposit as they have less money.

The generalised linear regression model also highlighted the significant variables for the bank that correlates with the subscription rate. The coefficient value indicates the estimated change in the log-odds of each variable per unit increase for the subscription rate. We analysed the coefficients with statistical significance ($p < 0.05$) and found the following with the greatest correlation on subscription rate (Table 7). Past satisfied clients showed up as the most responsive customers to marketing campaigns. "poutcome_success" has a coefficient of 1.08, indicating that customers with a successful outcome in a previous marketing campaign are more likely to respond positively. Seeing that previous customers show a more positive outcome to future campaigns suggests that businesses should focus time on retention rather than only on new customers.

| | Coefficient |
|---|---|
| const | -104.232 |
| emp_var_rate | 0.7371 |
| cons_price_idx | 1.1094 |
| cons_conf_idx | 0.0303 |
| pdays_999 | -0.7348 |
| default_unknown | -0.2611 |
| contact_telephone | -0.7003 |
| month_mar | 1.0793 |
| poutcome_success | 1.0815 |

Table 7: GLM Coefficient Score

After careful analysis of these features, it becomes evident that several factors strongly correlate with customers' inclination to subscribe to a term deposit, particularly in relation to the prevailing economic conditions at the time of contact. Notably, when the economy is thriving, customers exhibit a higher likelihood of subscribing compared to periods of economic downturn. Factors such as elevated interest rates and job uncertainties significantly influence customer responses, leading to a higher likelihood of rejecting the offer. We also found that past customers of the bank are more responsive to the marketing campaign.

### 7.2 Store

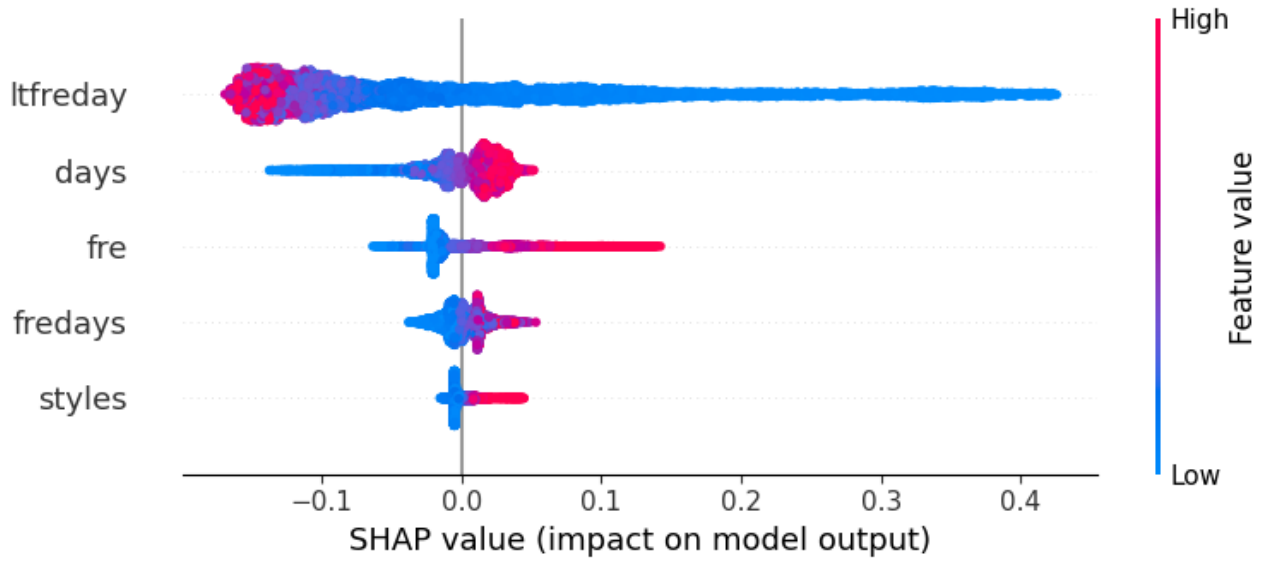



Figure 7.ii: SHAP plot of random forest on store data set

Similar to the bank data set the SHAP plot above shows the most influential features. we can see that the feature with the most correlation with the class predictions is the lifetime average of days between visits (ltfreday) for a single customer. When the customer has a low interval between visits they are more likely to respond to the online marketing email and when they have a large interval between each visit they are more likely to not respond to the marketing email. This could be due to the fact that when they have a low interval between visits they are familiar with the store and when they receive the email they have a lower probability of thinking it is a scam and therefore more confident to respond to it. The days the customer has been on file also contribute to the customer responding, when they have been on file for a longer period of time logically they would have been sent more emails from the company and are therefore more likely to respond to at least one of the emails versus someone who has only been sent one email. When the number of purchase visits (fre) is high the customer is more likely to respond to the email campaign, and when the number of purchase visits is low the customer is more likely not to respond to the email campaign. The final feature we would like to mention for the store data set is the total number of items the individual has purchased (styles), when the number of items purchased by the customer is high they are more likely to just purchase another one and thus respond to the email campaign. However, if the number of items purchased is low this means they are not comfortable with making a lot of purchases which pushes them to not respond to the marketing campaign.

Furthermore, the generalised linear model highlighted several additional influential features (Table 8). With the digitalisation of shopping and the impact of the COVID-19 pandemic, customer trends toward traditional shopping have changed. Web shoppers (web) show more willingness to respond positively to the email promotion campaign of the store with a coefficient of 0.58. Therefore, businesses such as the store may benefit from specialised online email promotion campaigns. Furthermore, credit card users also show as more responsive, with a positive association with responding to the campaigns.

| | Coefficient |
|---|---|
| const | -2.0606 |
| fre | 0.0680 |
| ccspend | 0.3829 |
| web | 0.5871 |
| valphon_y | .2333 |
| omonspend_0 | -0.2543 |
| smonspend_0 | -.6636 |
| markdown_0 | -2770 |
| responserate_0 | -.2075 |

Table 8: GLM Coefficient Score

### 7.3 Combined insights

For the bank data set, customers show more willingness to subscribe when the economy is stable and they have a more disposable income to put into the term deposit or the customers have previously subscribed. We can see this effect more in the mutual information table in the appendix Table 10 where the features with high mutual information usually are not related to the customer's characteristics but rather the state of the economy.

For the store data set, however, customers that tend to respond to the marketing campaign are those that frequently shop and also frequently purchase an item when they go shopping.

Therefore we can see that for the bank data set it is mainly when to run a marketing campaign rather than who to target. The client should focus mainly on running the campaign when the economy is stable and doing well and prioritising the customers that already have responded to a term deposit before.

By combining the two data sets results we can see that the customers that tend to respond to the marketing campaigns are those that are already previous customers and that the campaigns should run at a time of economic stability.

## 8 Conclusion

As a team of business analysts in a marketing consultancy, we have successfully provided essential quantitative insights, enabling both clients to make data-driven and cost-effective business decisions, by analyzing their untapped customer bases through comprehensive machine learning algorithms.

Our analysis involving both SHAP and GLM methods revealed a likely association of the success of marketing campaigns with economic environment, past customer behavior, and personal attributes. The bank data suggested that factors such as employment rates, interest rates, and consumer confidence may have a significant influence on the tendency of customer subscriptions. Likewise, customers with a history of subscriptions showed a higher likelihood of repeated subscriptions, indicating some degree of customer loyalty pattern.

Therefore, we recommend both clients develop effective marketing strategies considering economic contexts and focusing on engaging with their existing and frequent customers. For the bank, a stable economic phase, particularly with lower interest rates, would be a favorable time to launch marketing campaigns. Meanwhile, the store should aim to contact and email active online shoppers and those showing consistent shopping behaviors. These approaches will not only improve current effectiveness and loss of marketing campaign but also profoundly benefits two businesses future profits.

Further research would be needed to improve the performance of the overall modeling process. An error analysis could help us detect recurring false patterns or types of errors, leading to potential enhancements in predictive accuracy, which could be achieved by refining our feature representation or by focusing on a select portion of the training data. Access to more powerful computing resources would also allow us to train and tune models more efficiently. Some of our tuning sessions, such as

those for the MLP and model stack, exceeded 24 hours due to current computing resource limitations. Probability calibration could also be considered, which adjusts the predicted class probabilities to better align with the actual values. This approach could be beneficial due to the imbalanced nature of the datasets and the cost-sensitive analyses in the business scenario we defined.

The exploration of diverse potential algorithms was intrinsically fun. Knowing there is no single right solution to complex problems, the team adopted a wide range of iterative experimentation to provide data-driven insights for the bank and store partners. We business analysts enjoyed the whole process of playing around with different machine learning algorithms and cooperating with each excellent team member.

With our proposals, we believe the bank and the fashion store can optimize their marketing strategies as commercial institutions to achieve their campaign effectiveness, uncovering untapped opportunities among existing customers, and ultimately, developing long-lasting strategies to thrive in this ever-changing marketing world.

# A References

Czakon, J. (2023, May 9). F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose? neptune.ai. Retrieved May 21, 2023, from https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

How Much Does It Cost to Outsource a Call Center? (2020, December 1). Cloudtask.com. https://cloudtask.com/blog/how-much-does-it-cost-to-outsource-a-call-center#:~:text=Locally%20outsourced%20call%20center

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021b). An Introduction to Statistical Learning. Retrieved May 18, 2023, from https://www.statlearning.com/

Lundberg, S., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. Retrieved May 18, 2023, from https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into Deep Learning. Retrieved May 18, 2023, from https://d2l.ai/index.html

# B Appendix

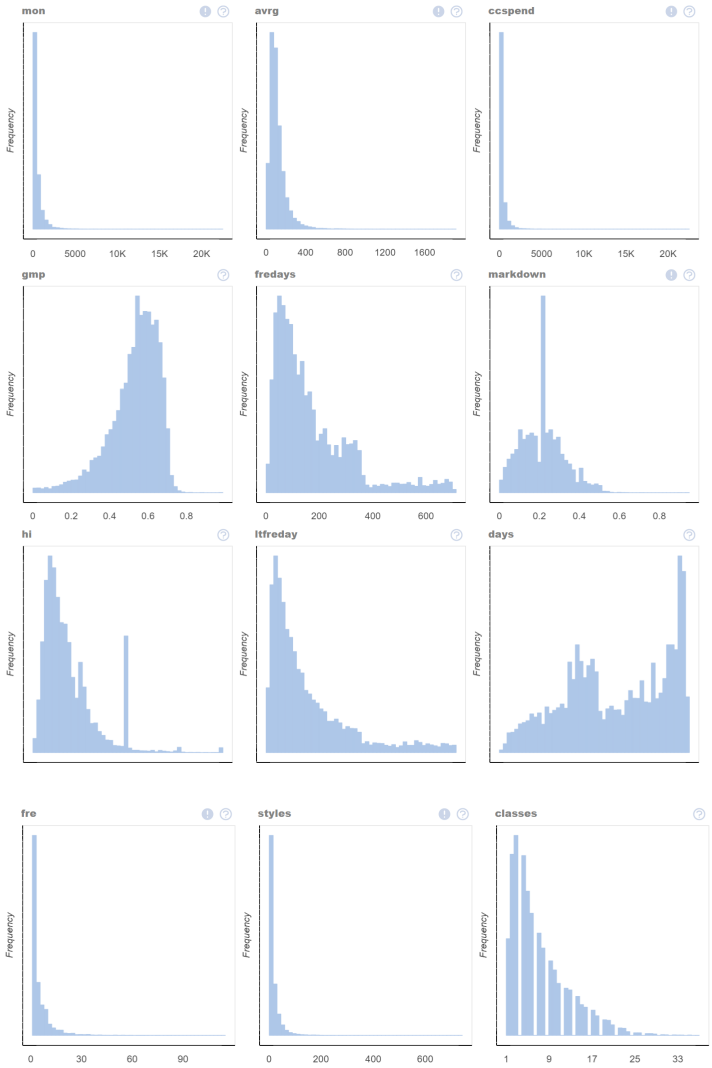
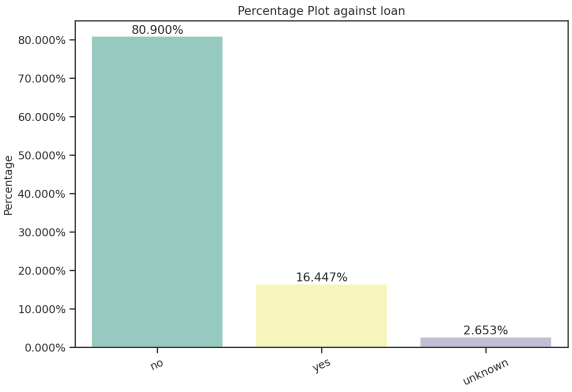
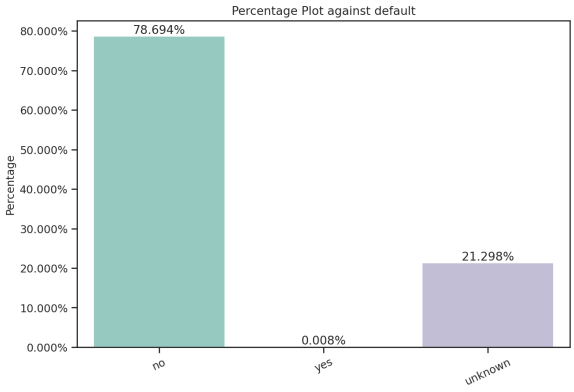


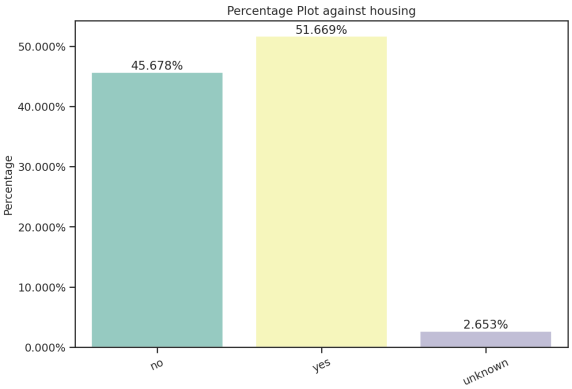

Figure B.i: Bank Continuous EDA



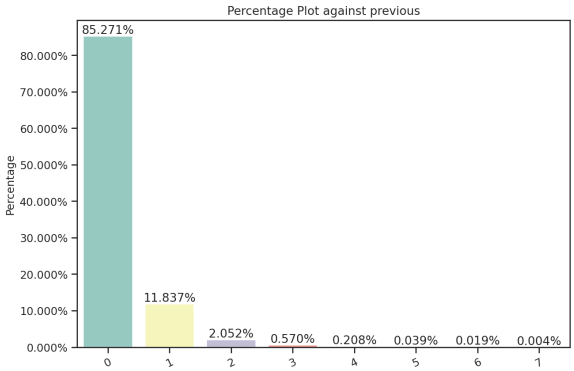

(a) Loan



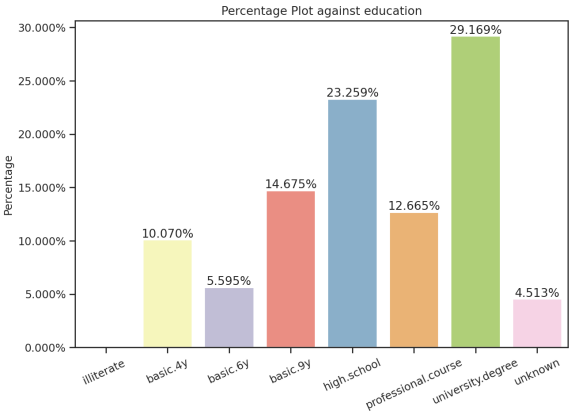

(b) Default



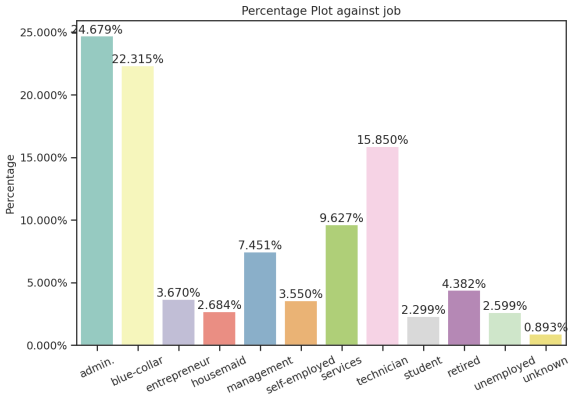

(c) Housing



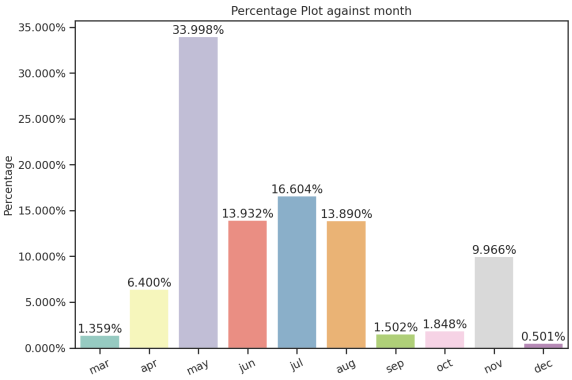

(d) Previous



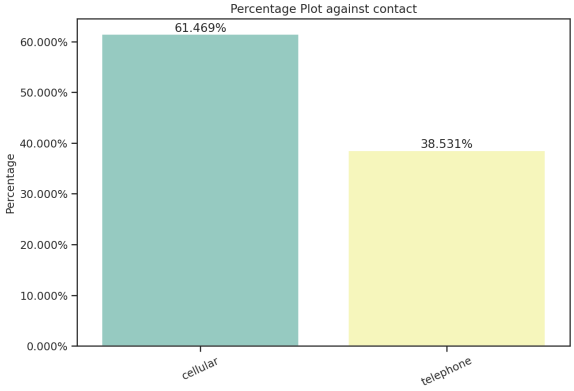

(e) Education



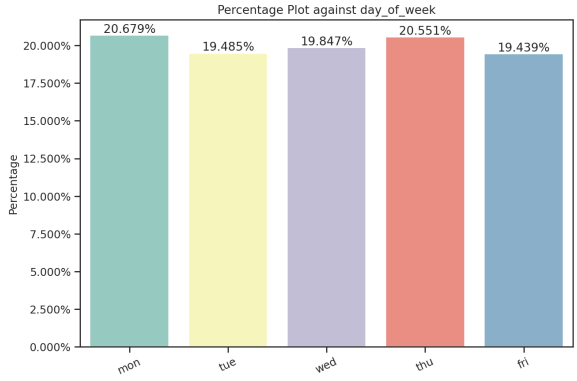

(f) Job



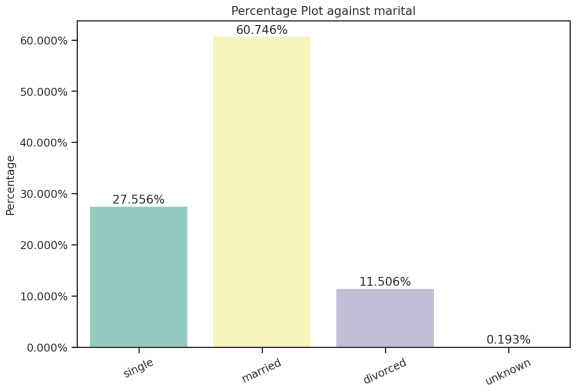

(g) Month



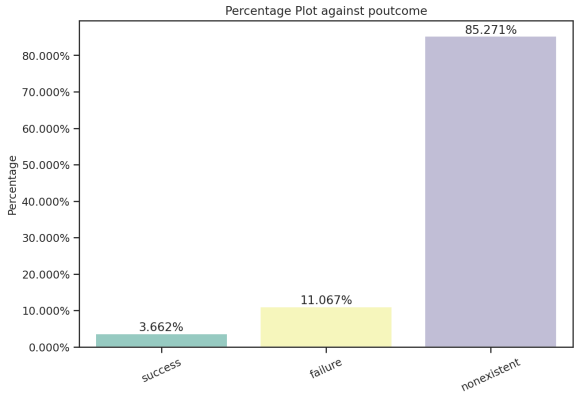

(h) Contact




(i) Day of Week

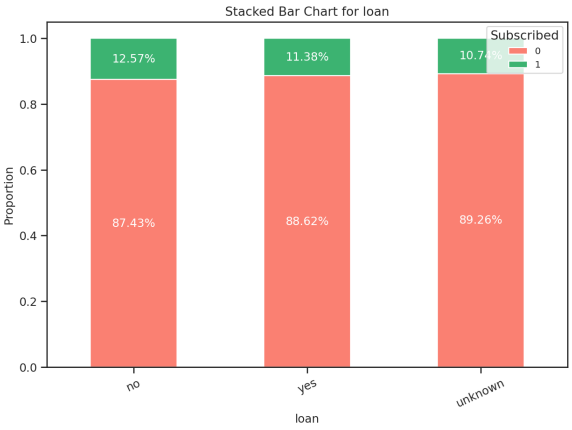



(j) Marital Status

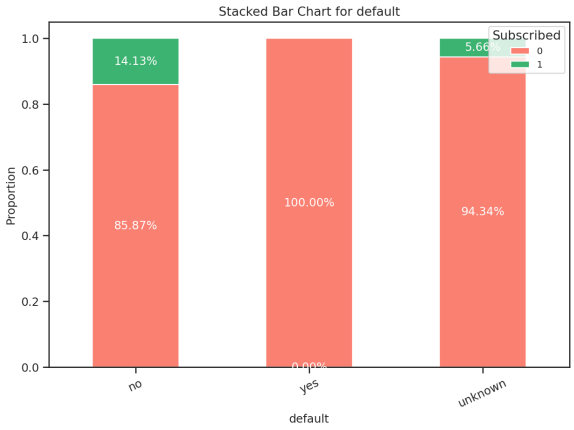



(k) Poutcome

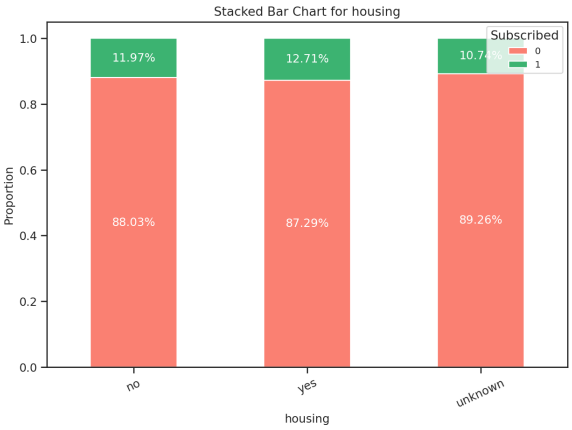
Figure B.ii: Bank Categorical EDA: Percentage Plot



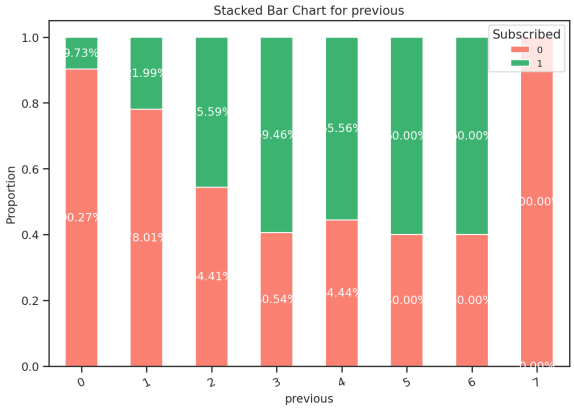

(a) Loan



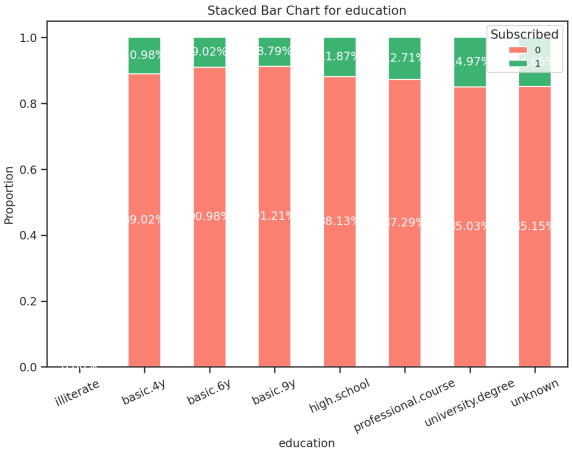

(b) Default



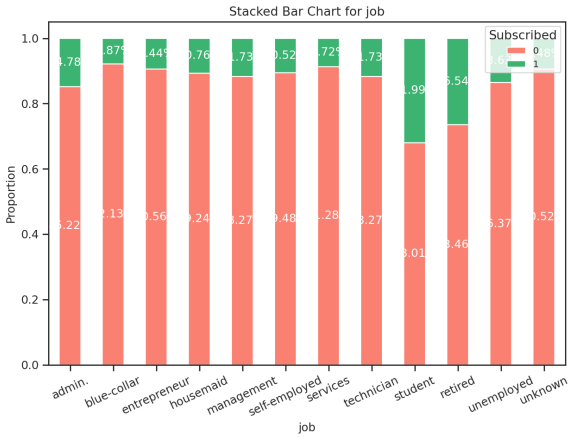

(c) Housing



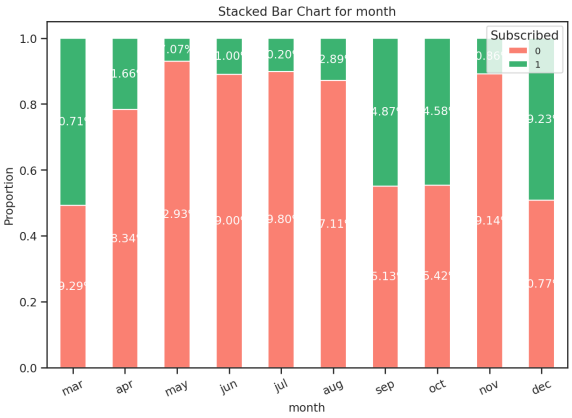

(d) Previous



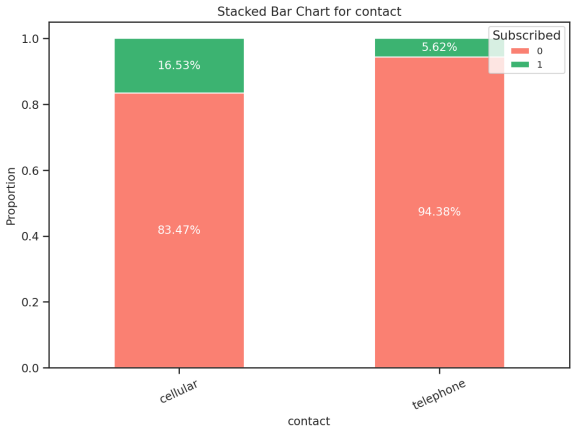

(e) Education



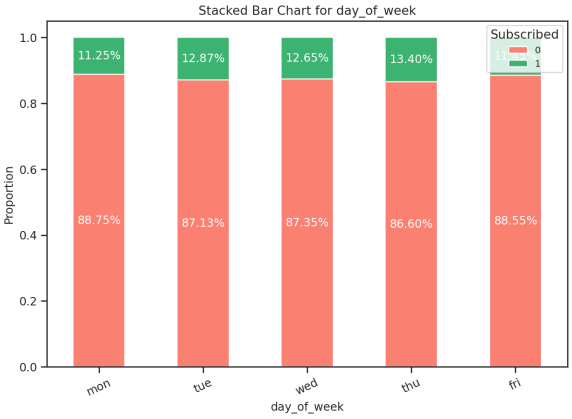

(f) Job



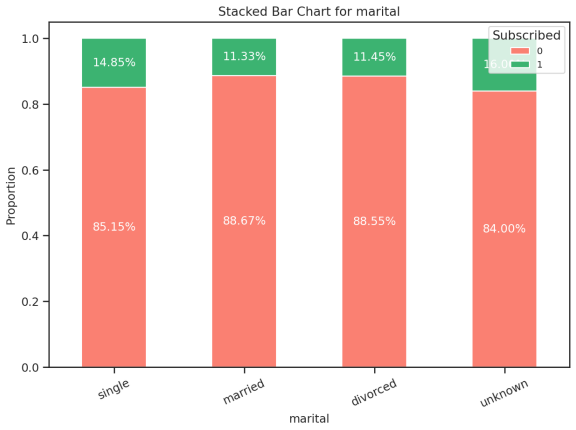

(g) Month



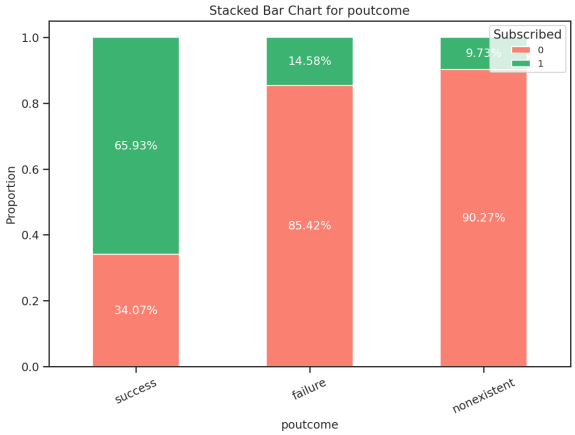

(h) Contact




(i) Day of Week




(j) Marital Status

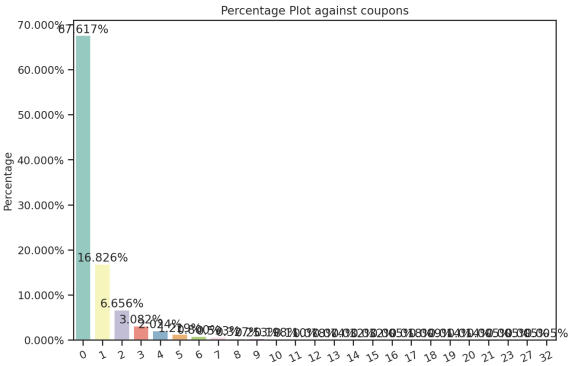



(k) Poutcome

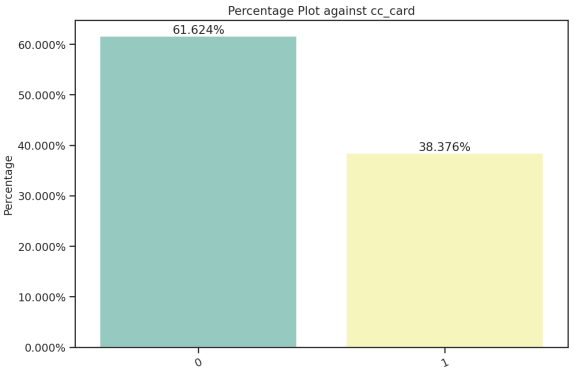
Figure B.iii: Bank Categorical EDA: Stacked Bars

| | Mutual Information |
|---|---|
| euribor_3m | 0.0751 |
| cons_conf_idx | 0.0710 |
| cons_price_idx | 0.0678 |
| nr_employed | 0.324 |
| emp_var_rate | 0.0554 |
| pdays | 0.0404 |
| poutcome | 0.0314 |
| month | 0.0266 |
| previous | 0.0194 |
| age | 0.0158 |
| contact | 0.0144 |
| job | 0.0100 |
| default | 0.0065 |
| education | 0.0027 |
| marital | 0.0016 |
| day_of_week | 0.0004 |
| loan | 0.0001 |
| housing | 0.0001 |

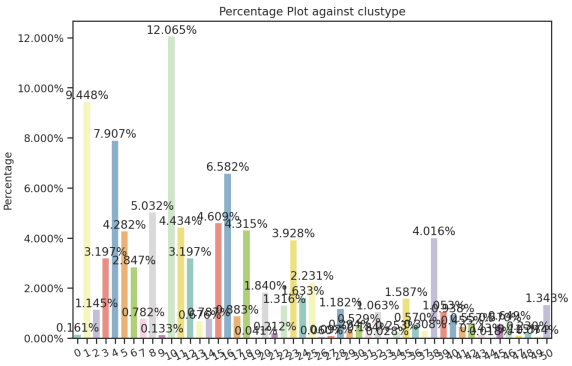
Table 9: Bank Mutual Information



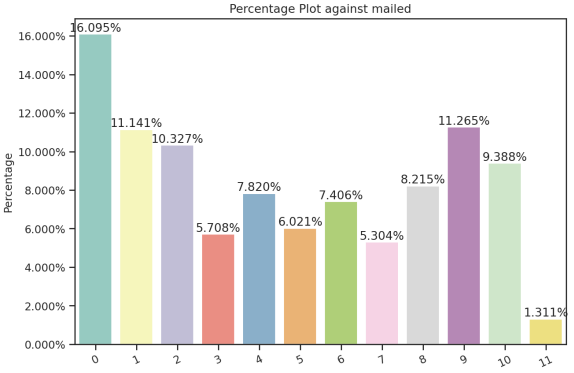

(a) Coupon



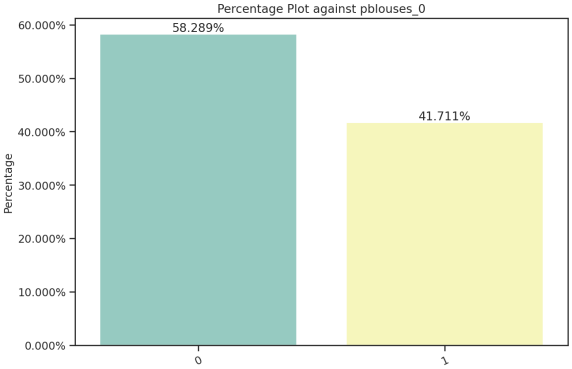

(b) cccard



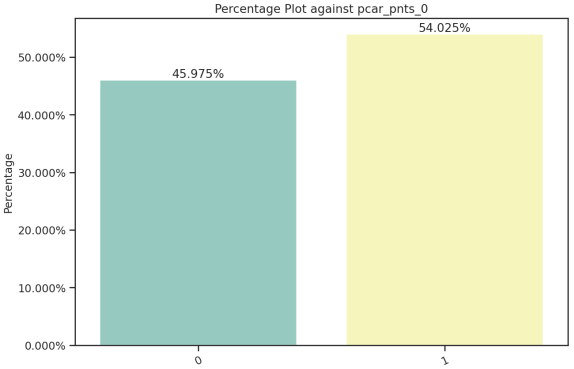

(c) clustype



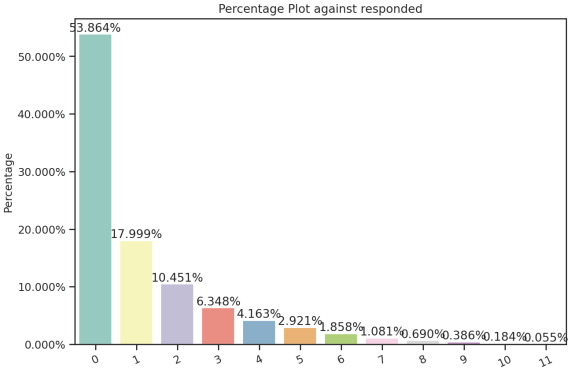

(d) mailed



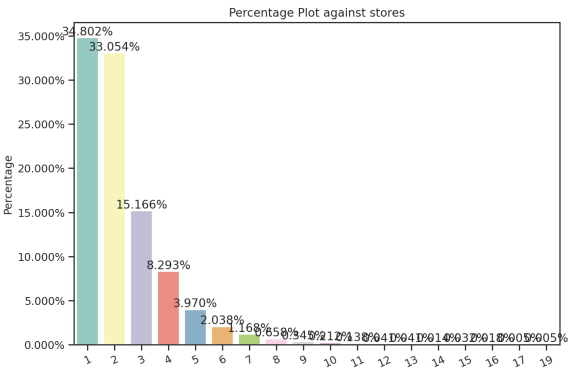

(e) pblouses0



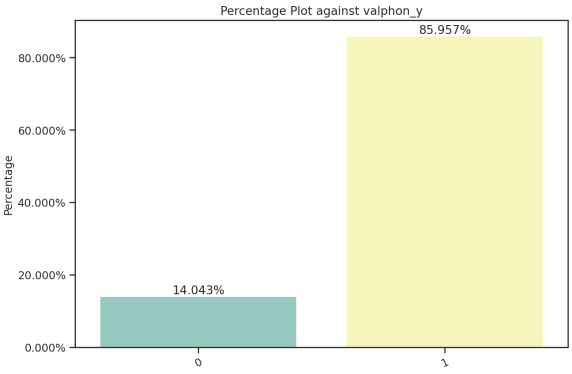

(f) pcarpnts0



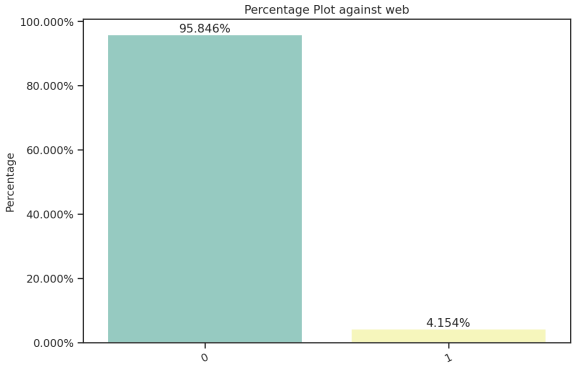

(g) responded



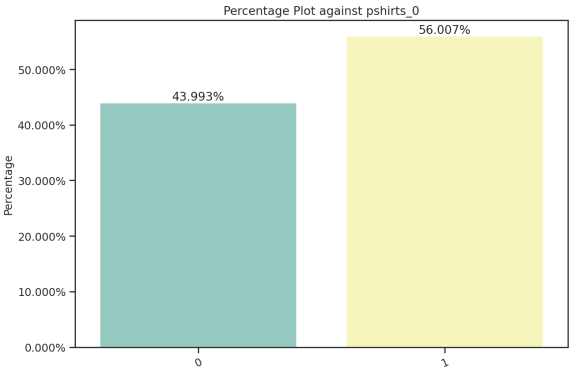

(h) store



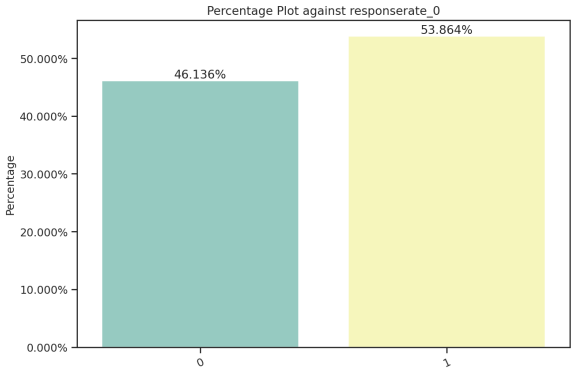

(i) valphony



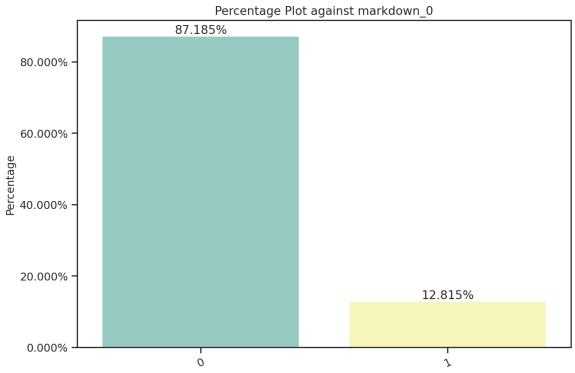

(j) web




(k) Pshirts0

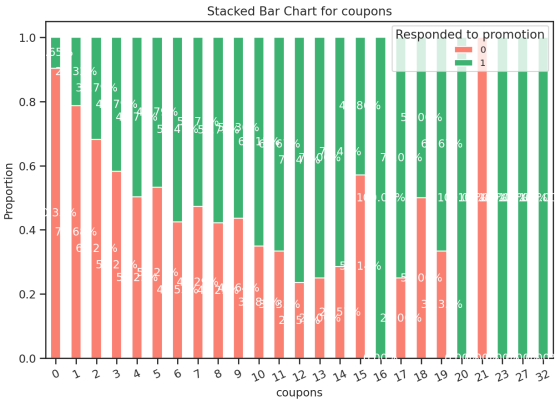



(l) responserate0

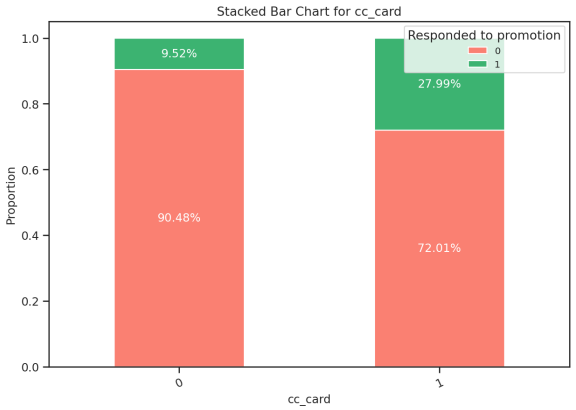



(m) markdown0

Figure B.iv: Store Categorical EDA: Percentage Plot

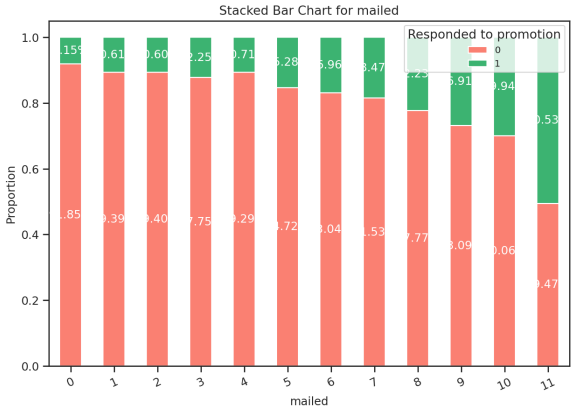



(a) Coupon

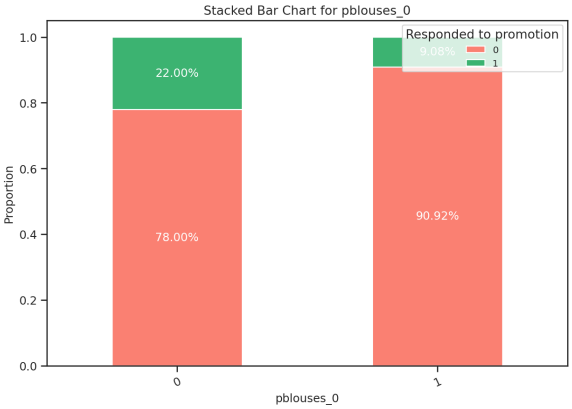



(b) cccard

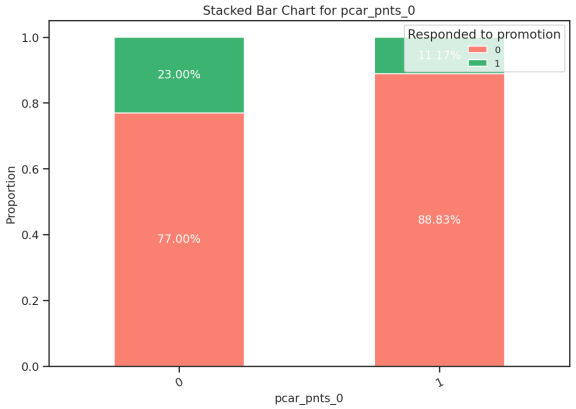

(c) clustype



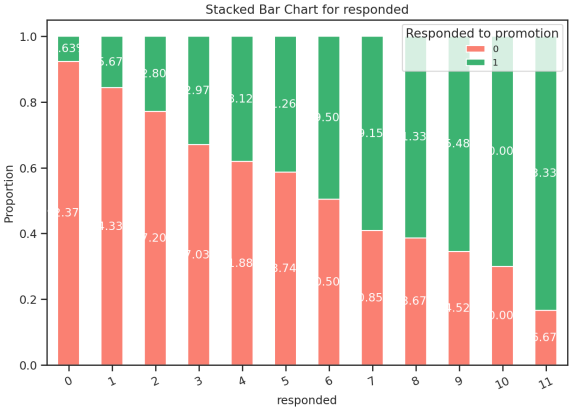

(d) mailed



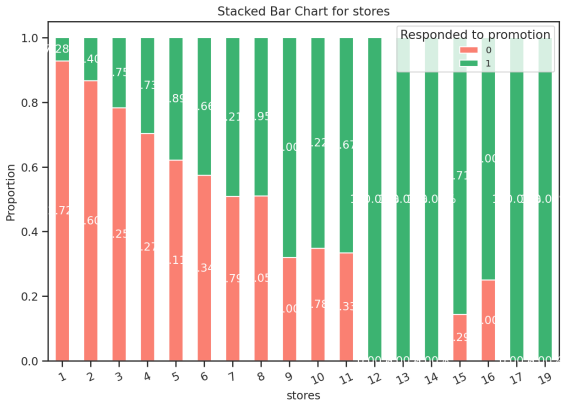

(e) pblouses0



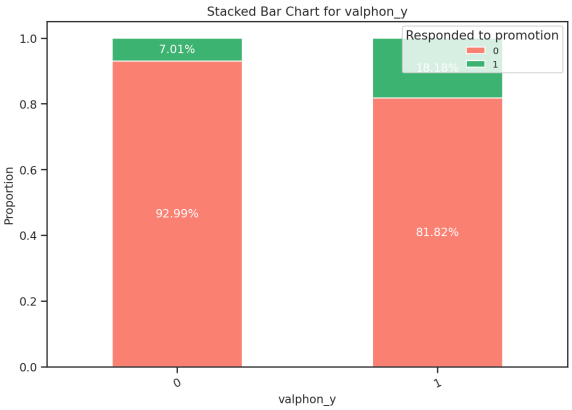

(f) pcarpnts0



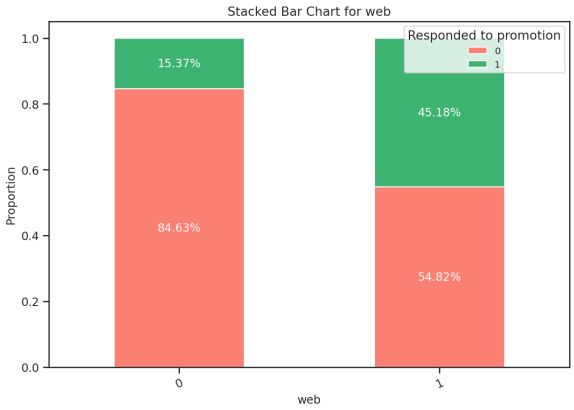

(g) responded



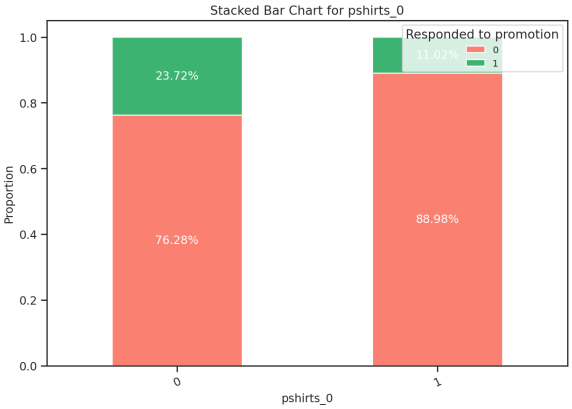

(h) store



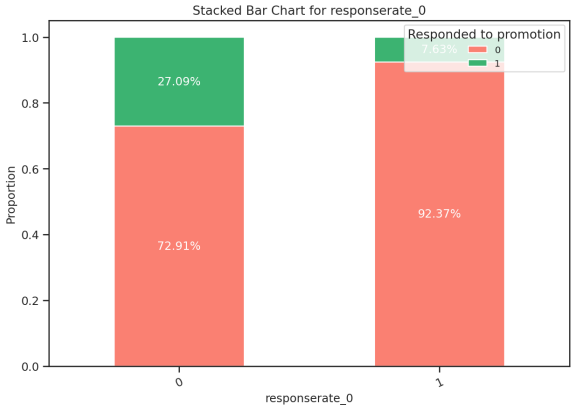

(i) valphony



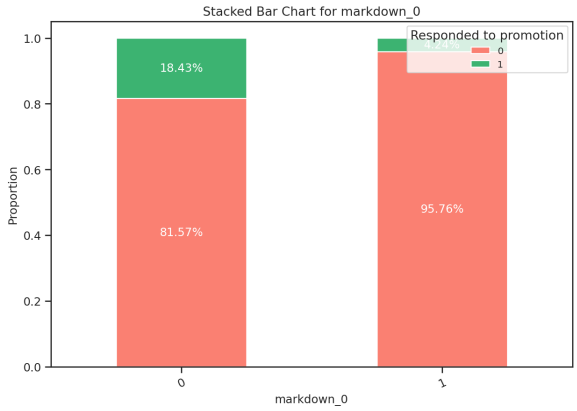

(j) web




(k) pshirts0




(l) responserate0




(m) markdown0

Figure B.v: Store Categorical EDA: Stacked Bars

| | Mutual Information |
|---|---|
| mon | 0.389 |
| avrg | 0.329 |
| ccspend | 0.324 |
| ltfreday | 0.324 |
| hi | 0.222 |
| fredays | 0.212 |
| fre | 0.083 |
| styles | 0.079 |
| classes | 0.064 |
| responded | 0.055 |
| days | 0.049 |
| coupons | 0.049 |
| stores | 0.040 |
| responserate_0 | 0.036 |
| percret_0 | 0.030 |

Table 10: Store Mutual Information

| | sensitivity | F1-score | Error Rate | Specificity | Precision | AUC | Cross-entropy |
|---|---|---|---|---|---|---|---|
| XGBoost | 0.2700 | 0.3660 | 0.1160 | 0.9710 | 0.5670 | 0.7530 | 0.3270 |
| LightGBM | 0.2590 | 0.3690 | 0.1100 | 0.9800 | 0.6450 | 0.7770 | 0.3000 |
| CatBoost | 0.2480 | 0.3570 | 0.1110 | 0.9800 | 0.6370 | 0.7730 | 0.3030 |
| CatBoost no encoding | 0.2400 | 0.355 | 0.308 | 0.984 | 0.6600 | 0.6820 | 0.2970 |
| GBM | 0.2230 | 0.3380 | 0.1090 | 0.9860 | 0.6960 | 0.780 | 0.2960 |
| Histogram-Based GBM | 0.1890 | 0.2980 | 01110 | 0.9890 | 0.7010 | 0.777 | 0.2960 |

Table 11: Metric for different boost methods

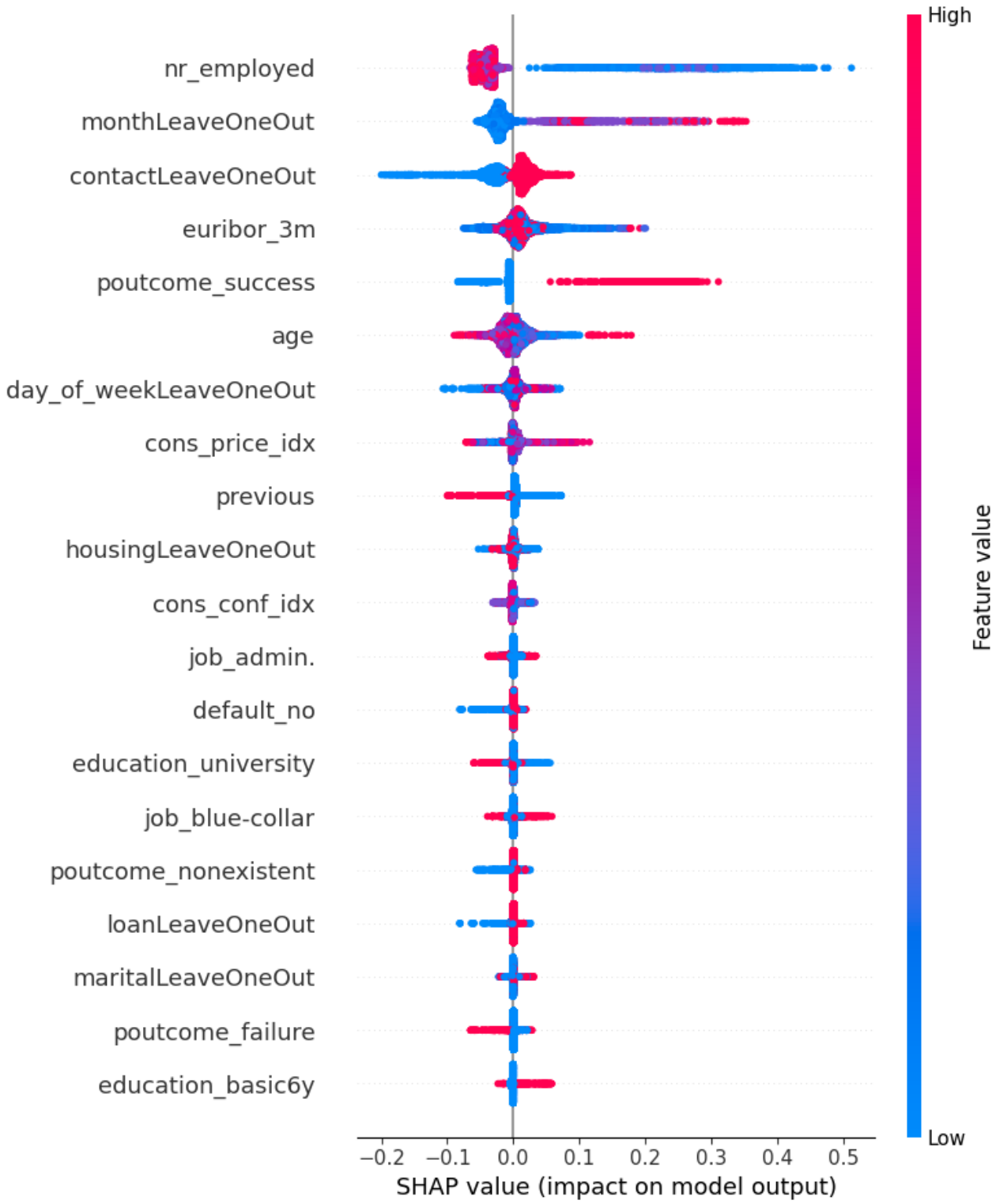



Figure B.vi: Full shap feature plot for bank

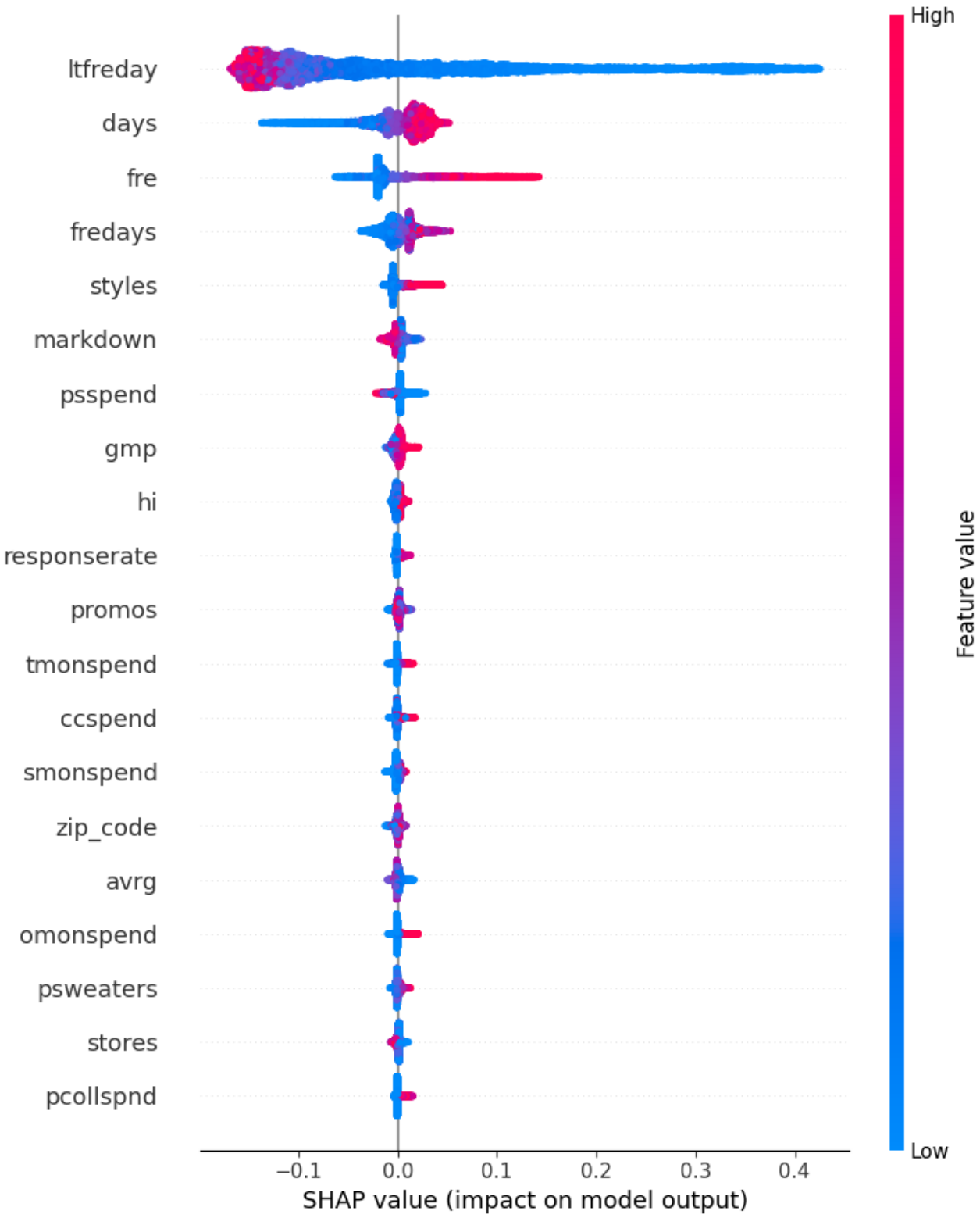



Figure B.vii: Full shap feature plot for store

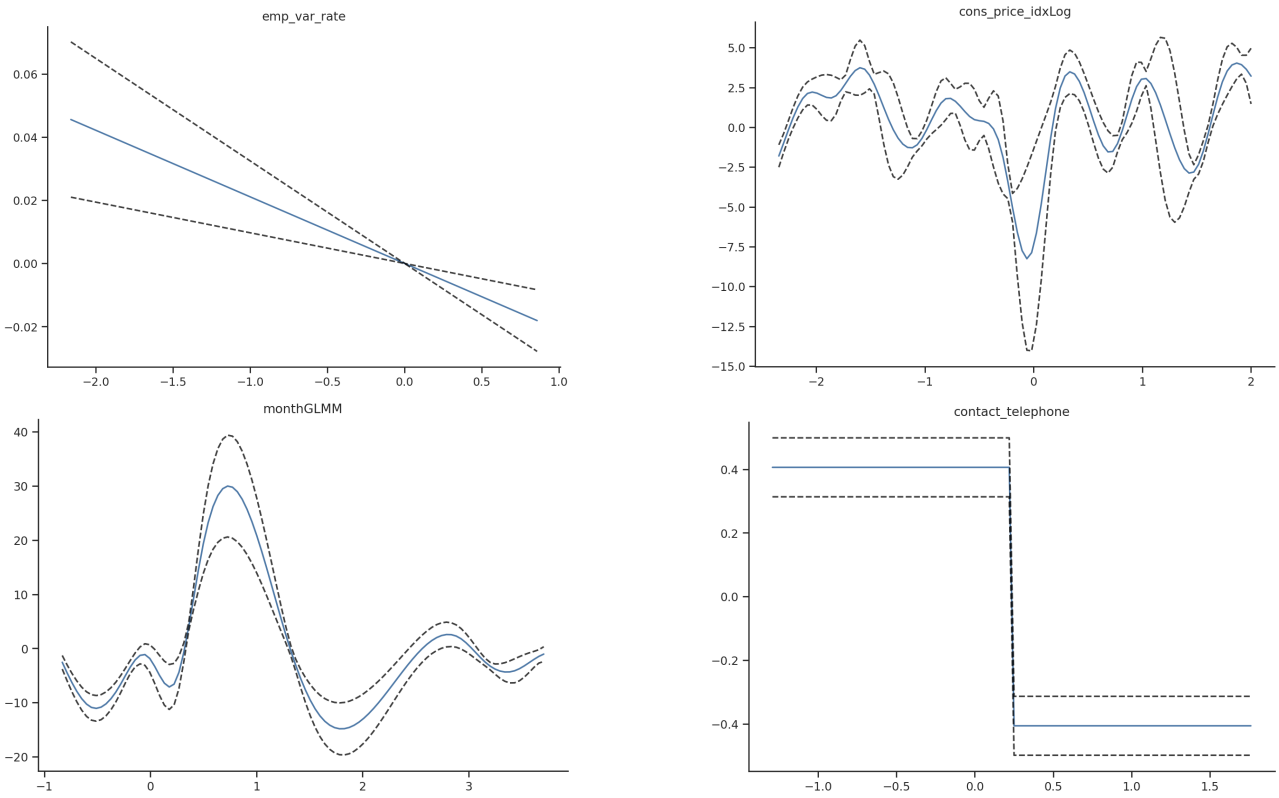



Figure B.viii: Partial dependence plot for GAM

| Variables | VIF |
|---|---|
| styles | 26.46451 |
| classes | 25.86513 |
| amspend_0 | 20.92434 |
| mon | 20.03423 |
| days | 18.72705 |
| pouterwear_0 | 14.14121 |
| fre | 11.57381 |
| axspend_0 | 10.10729 |
| avg_sale_per_item | 9.45545 |
| stores | 9.11770 |
| psuits_0 | 9.06318 |
| mailed | 8.56851 |
| hi | 8.40614 |
| responded | 8.39563 |
| fredays | 7.97119 |
| valphon_y | 7.96082 |

Table 12: VIF for logistic regression

- 'objective': 'binary'
- 'boosting_type': 'gbdt',
- 'learning_rate': 0.01,
- 'num_leaves': 44,
- 'lambda_l1': 2.3940244060036417,
- 'lambda_l2': 8.243998317794096e-08,
- 'bagging_fraction': 0.7165092950708541,

- 'bagging_freq': 3,
- 'feature_fraction': 0.5253096835972446,
- 'min_data_in_leaf': 17,
- 'feature_pre_filter': False,
- 'verbosity': -1