Bank Marketing Case Study
Brandi Rodriguez

I. Executive Summary

Banks today that are faced with growing pressure to increase profits and reduce costs, in addition to fierce competition within the industry, must find innovative ways to reach the right customer in the most efficient and cost-effective way. Well-known classification modeling techniques are used in this case study to predict which clients are likely to subscribe to a bank term deposit. The focus of this study is on classification modeling and centers around data related to the direct marketing campaigns of a Portuguese banking institution which is used to train Logistic Regression and Linear Discriminant Analysis models. The fitted logistic regression model outperformed all other models, with a 90.64% accuracy rate and 77.3% AUC. *'Emp.var.rate'*, *'contact'*, '*month*' and '*poutcome*' were the most significant predictors.

II. The Problem

Like many financial institutions, and companies in general, the bank is challenged with trying to target the right customer in the most cost-effective manner. A bank that understands its customers better can gain an advantage over its competitors. This case study highlights the value of data analytics as a tool to inform product innovation, marketing strategy and decision making. A predictive model can easily be deployed by a bank to analyze a large number of clients quickly and efficiently to help optimize its telemarketing campaigns. Having a reliable and accurate customer response model is critical for marketing success since an increase or decrease in accuracy of 1% could have significant impact on their profits (Olson).

The data collected by the bank contains attributes related to the client, campaigns, and socio-economic indicators. The variable "y" indicates whether the client purchased a long-term deposit, which is the target variable of this study. It is leveraged to train two predictive models with the goal of predicting whether a client is likely to subscribe. The study seeks to answer the question of:

- Which technique yields the best predictiveness?
- Which variables are most important to the model's predictive accuracy?
- Will a client subscribe to a term deposit in future marketing campaigns?

To accomplish this, we'll start with a brief introduction into the literature related to the processes utilized in this study, followed by a detailed description of the procedures taken for this study including an analysis of the variables, a discussion on the data cleaning and model training and testing process, then concluding with a comparison of model performance including: AIC, Accuracy, Sensitivity, Specificity, and AUC.

III. Review of Related Literature

Response Modeling is a predictive modeling approach used by marketers that predicts future response probabilities to customers based on their history with the company (Coussemant). Several classification methods such as logistic regression, linear and quadratic discriminant analysis, naïve Bayes, neural networks, and decision trees – each with their own different set of assumptions and limitations – can be used to predict customer responsiveness. In their article, "A Data-Driven Approach to Predict the Success of Bank Telemarketing" Moro, Cortez, and Rita – who first analyzed the dataset – apply logistic regression, decision tree, neural network and SVM models to the bank dataset to predict the success of telemarketing calls for selling bank long-term deposits. They found neural network model achieved the highest AUC (79.4%) and outperformed all other models with an accuracy rate (93.37%). When considering the importance of individual features, it was interesting that the 3-month Euribor rate (euribor3m) was considered the most relevant attribute, with a relative importance around 17% (Moro, Cortez, Rita). Moro et al found that a lower Euribor corresponded to a higher probability for deposit subscriptions, which is surprising because most banks align their deposit rates offered with an index such as the Fed Funds or ECB index. Moro et al provided additional research into why this occurred, which emphasizes the inclusion of "the human touch" to evaluate, provide research and give more context when performing datamining projects such as this.

IV. Methodology

In this case study, logistic regression and linear discriminant analysis models are built featuring different data transformations and variables. The results will be compared using AIC, accuracy, sensitivity, and specificity rates, and AUC. The data used for this study comes from the UCI Machine Learning Repository. It was collected from a Portuguese bank's telemarketing campaigns and contains a total of 4,119 observations and 21 variables including bank client attributes (i.e. *age, job, marital status, education, default, housing, loan*), campaign attributes (*contact, month, day_of_week, duration, campaign, pdays, previous, poutcome*) and social-economic attributes (*employment variation rate, consumer price index, consumer confidence index, Euribor 3-month rate, and number of employees*). The binary target variable is '*y*', with 'yes' meaning the client subscribed to a term deposit, and 'no' if they did not. **Appendix A** contains a detailed description of the dataset.

The data is used to train logistic regression and linear discriminant analysis models. A top-down strategy was used, where different variations of the full model are trained (i.e. with certain variables scaled vs unscaled, outliers imputed vs. not imputed, columns dropped vs retained, etc.). The best performing full model then gets refitted using only the significant predictors. The dataset is imbalanced (89% "no" and 11% "yes"), which may distort the algorithms and its predicting performance. This often causes models to fit the majority class better to improve the overall accuracy, but the goal is to be more successful at identifying people who will subscribe to a term deposit, rather than the overall power of prediction. Thus, the ROC curve and AUC will also be used as a performance metric in addition to accuracy rate. An incremental analysis was performed, starting with a full logistic model, followed by models with several different variations to the predictors. A new dataset is created for each variation and some are later "recycled," as seen in **Appendix B** which provides a detailed table of the various models, datasets created, significant predictors, performance metrics, and findings. **Appendix C** contains a table with descriptions of each of the unique datasets created for this analysis.

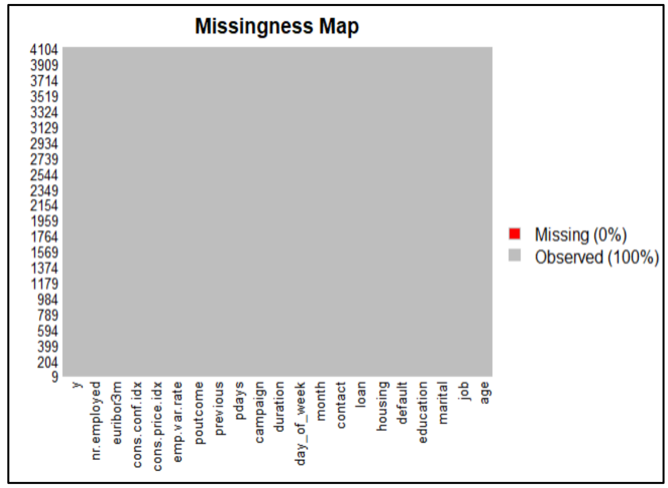
The goal of logistic regression is to maximize the likelihood that a random data point gets classified correctly. It uses Bayes Theorem to estimate probabilities, producing discrete binary outcomes between 0 and 1 which are then transformed to a '0; or '1' using a threshold classifier (Machine Learning Mastery). For this analysis, the threshold is initially set to .50. The optimal threshold is then calculated, but when applied to the model, it significantly reduced accuracy and specificity, while boosting sensitivity. This model has the highest sensitivity rate so far. The logistic model operates under the following assumptions that can lead to inaccurate or erroneous results if they are violated: the predictors must have a linear relationship with the log odds of the binary response variable and must not present multicollinearity. To avoid this, a check was performed for correlation and multicollinearity on the final model. The predictors must be independent and follow a Gaussian distribution. Variables that do not follow a Gaussian distribution can be transformed with a log or boxplot transformation. It also assumes homoscedasticity, that there are no outliers in the continuous predictors, and a large sample size. As one of the most popular algorithms for binary classification, it is used as the baseline model. Although logistic regression models are easier to interpret, they can be rigid and sometimes cannot adequately model complex nonlinear relationships (Moro, Cortez, Rita). It's also vulnerable to overfitting and can easily be outperformed by more complex (albeit less interpretable) models. Furthermore, it can become unstable when classes are well separated, as well as when there are too few observations from which to estimate a parameter.

As an alternative, an LDA model will be trained. Unlike logistic regression, which is traditionally limited to only two-class classification problems, LDA is a more flexible, robust method that can perform multiple class classification. It seeks to maximize class separation by finding the largest separation between class means, while also giving the smallest variance within each class. It uses estimates of the mean and variance from the data for each class to make predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made. Although more complex models often achieve better accuracy, the increased complexity can make the model more difficult to interpret. To overcome this, a sensitivity analysis can be performed by analyzing the response of a model when a given input is varied through its domain (Moro, Cortez, Rita). LDA assumes the data follows a normal distribution and each variable has class-specific means with equal variances. It is a more robust technique than logistic regression and does not require dummy encoding of categorical variables.

V. Data

After reading in the data, a copy was created with duplicates removed, and columns reordered to start with the response, followed by continuous numeric variables, then discrete numeric variables, and then factor variables. There dataset has 4,119 observations and 21 features. The descriptive statistics and structure of the data were reviewed using summary() and str() function. None of the categorical variables need to be coerced to factor variables. It is apparent that 'default' won't be a useful variable, as it has three levels with only 1 observed "yes" and all others as "no" or "unknown." 'Pdays' has a large proportion of 999s, which means the client was not previously contacted, so it may be a candidate for removal in the training dataset.

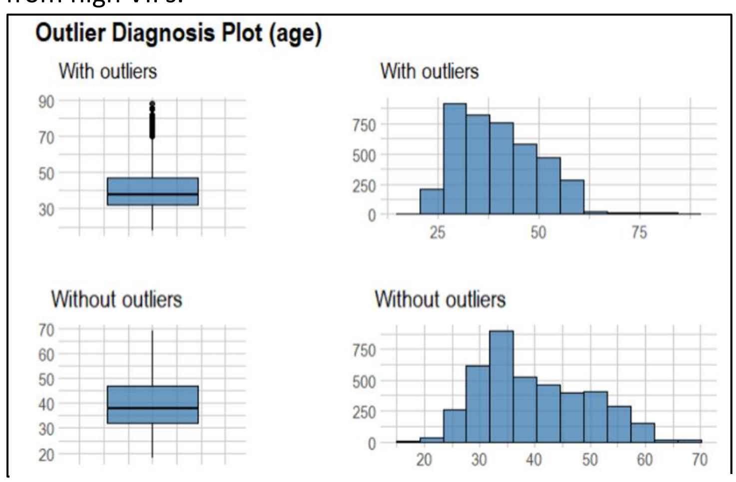
Missing values were checked using the missmap() function from the Amelia package showed the dataset was complete with no missing values. Exploratory Data Analysis was performed to discover patterns and correlations before applying any machine learning algorithms. The DataExplorer package was used for quick data visualization (see **Appendix D**). The bar charts plotted of all categorical variables make it quickly apparent that the majority of clients have not subscribed to a term deposit. A large portion of clients have administrative and blue-collar jobs, are married, and have a university degree. Several of the categorical predictors contain a level labelled "unknown." These will later be treated as NAs. There were slightly more clients who had a home loan, while a large majority did not have a personal loan. More were contacted by cell phone and most of the prior contacts with them took place during summer months. 'Default' does not appear to be a useful feature. It has three levels, yet only 1 'yes'. It's a good candidate for removal.



A larger portion of those last contacted in October, September, March, and December subscribed to a term deposit, while those contacted in the summer months were less likely to subscribe. This is interesting because the summer months were when most of the clients were contacted, while significantly fewer prior contacts were made in October, September, March and December. Those who were contacted by cell phone rather than telephone, as well as those who had a 'success' as the outcome of the prior marketing campaign were more likely to have subscribed to a term deposit (*'poutcome'*). The proportion of those who subscribed to a deposit appears to be the same, regardless of day of week (*'day_of_week'*), whether they had a home loan (*'housing'*), or personal loan (*'loan'*). Those contacted by cellular had a larger proportion that subscribed to a term deposit (*'contact'*).

For the numeric variables, 'age' is centered around age 30 to 40 with a right skewed distribution. 'Campaign' is right skewed as well, with most clients contacted 5 or less times during the current campaign. All values for 'pdays' appear to have taken a value of 0 or '999', with an overwhelming majority as '999'. Because so many were '999', this feature is a candidate for removal from the final data for model training. When correlations were evaluated, all socio-economic features proved to be highly correlated. Emp.var.rate and euribor3m had the strongest correlation (.97), followed by euribor3m and nr.employed (.94) and emp.var.rate with nr.employed (.90). These predictors were kept in the dataset and would be removed later on if multicollinearity was detected from high VIFs.

The presence of outliers can skew the basic statistics used to separate classes in LDA such as the mean and standard deviation. Instead of removing outliers, they were replaced with imputed values to retain the useful information contained from the rest of the variables for each of the observations containing outliers. This was applied to *'m7'* (a logistic regression model) and an alternative fitted LDA model. Outliers can be transformed to follow a Gaussian distribution by using log and root transformations for exponential distributions and Box-Cox for skewed distributions. The outliers were imputed using the imputate_outlier function from the dlookr package using the 'capping' method which imputes the upper outliers with 95 percentile and the bottom outliers with 5 percentile.



An example of how a variable's distribution is reshaped by reducing outliers.

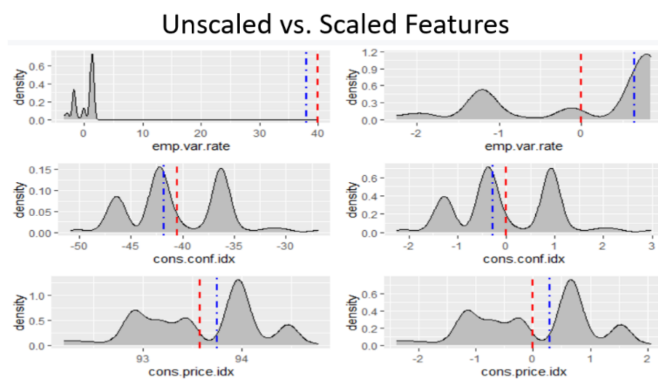
Call '*duration*' denotes the duration of the last contact, in seconds. This variable highly affects the response variable (when duration is 0, y is 'no.' However, duration is only known after call has been executed. Although it can provide additional insights to the Bank if used under a different context, it was discarded for this case study as the intent is to create a predictive model. '*Pdays*' was removed, given the majority of values were 999's indicating most clients had not been contacted previously, and the variable *previous* (# of contacts performed before this campaign) may serve as a more useful alternative. '*Default*' did not provide much useful information, having only 1 "yes", and all others "no" or "unknown", so it was subsequently removed. It was previously noted that several categorical variables were coded as "unknown." In fact, a fifth of all values for '*default*' were labelled unknown. '*Education*', '*housing*', '*loan*', '*job*', and '*marital*' also had unknown values that were treated as NAs and subsequently removed. Lastly, '*emp.var.rate*' was removed due to the high correlation it had with some '*euribor3m*' and '*nr.employed*', reducing the full dataset from 4,119 to 3,811 observations and from 21 variables to 17.

```
raw_data %>%
  summarise_all(list(~mean(. == "unknown"))) %>%
  gather(key = "variable", value = "Unknown_Percent") %>%
  arrange(-Unknown_Percent) %>%
  head(10)
```

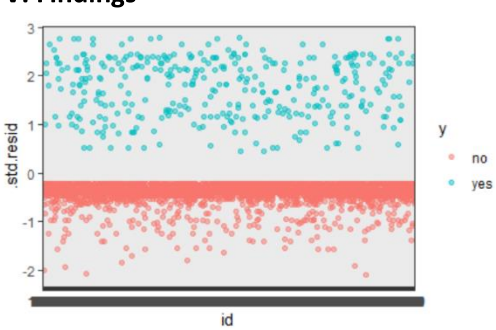
	variable <chr>	Unknown_Percent <dbl>
1	default	0.194950231
2	education	0.040543821
3	housing	0.025491624
4	loan	0.025491624
5	job	0.009468318
6	marital	0.002670551
7	age	0.000000000
8	contact	0.000000000
9	month	0.000000000
10	day_of_week	0.000000000

An important step for classification using logistic regression is the use of dummy variables. In the article "Regression Analysis with Categorical Variables", Venkataramana explains why the use of categorical independent variables in a regression model involves the application of dummy coding. The number of dummy variables encoded must be one less than the number levels a categorical variable has and the dummy variable coefficients are interpreted in relation to the base or reference group (Venkataramana). The *dummy_cols* function from the *fastDummies* package was used to **create** the dummy variables needed. The categorical variables are then dropped from the original data and the dummy variables are added (with special care to discard one of the dummies for each categorical variable to avoid redundancy since one var already becomes the reference group for regression and to avoid perfect multicollinearity).

The final dataset has 4,119 observations and 53 variables. Lastly, feature scaling was also introduced in the logistic regression model, m8 and alternative LDA models**,** as the range of variables measured on different scales may differ a lot. Using the original scale may put more weight on variables with larger ranges, resulting in disproportionate influence. Feature scaling can be used to bring all values to the same magnitudes to solve this issue. Feature scaling is important when the machine learning model is fundamentally based on a distance matrix, also known as a distance-based classifier such as: KNN, SVM, Logistic Regression and Neural Networks. If an algorithm is not distance-based, feature scaling is unimportant, including Naïve Bayes, LDA, and Tree-Based models (Clare Liu). The data should be scaled after the dataset is split into training and testing sets, and should be done to each individually. Doing it to the entire combined dataset could lead to a biased estimation if information from the testing set ends up included in the training set after being split.

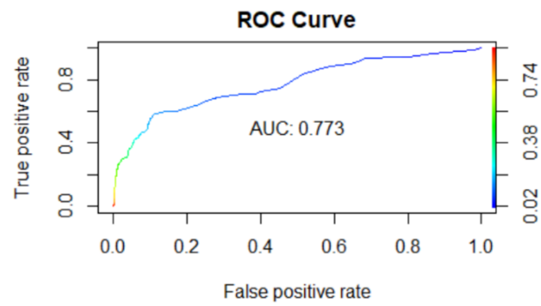


V. Findings



The baseline model was a full logistic regression model with all variables and no feature engineering. It resulted in one of the highest accuracy rates (888.94%) and AUC (76.90%) and the significant predictors were '*emp.var.rate*', '*age*', '*contact*', '*month*' and '*poutcome*'. Additional models were trained using different variations to the dataset to see which transformations help improve the model's predictiveness (refer to **Appendix C** for full list of datasets and descriptions). The results were then compared to the baseline, while highlighting the models that performed the best on certain metrics. For example, m3 – a logistic model where the variable '*age*' is binned into three age-based groups – resulted in the highest AUC of 77.3%. Encoding the categorical variables as dummy variables did not have any impact in this case. Other interesting findings were noted, such as the fact that in m2, dummy variable encoding did not impact performance metrics. '*m8*', a full logistic regression model with '*age'* binned and feature scaling, produced the highest accuracy (89.55%), Specificity (55.26%) and AUC (77.30%) up to that point. Sensitivity was also relatively high (91.21%). However, it did have one of the highest AICs.

Next, logistic regression models were fit on the significant predictors (*logit.final=glm(y~emp.var.rate + age + contact + month + poutcome, data=train9, family = binomial*) from the baseline model, using the same feature engineering from m8, the highest performing full logistic model (the test and train datasets were an exact copy of those used in m8, which bins '*age*' and scales features). This model outperformed all others, with an accuracy rate of 90.64% and an AUC of 77.30%. Several LDA models were trained that produced subpar results relative to the final logistic regression model, '*logit.final*'. The assumptions of the logistic regression were then evaluated, checking the standard residuals, Cook's distance for outliers, VIFs for detecting multicollinearity, etc. (see final code output in **Appendix E**) before plotting the ROC Curve and AUC.



The final logistic regression model selected tells us there's a negative association between '*emp.var.rate*' and those who subscribe. The odds ratio of .644 suggests that holding all other predictors fixed, we expect to see about a 64% decrease in the odds of subscribing to a term deposit for a one unit increase in '*emp.var.rate'*. The .678 odds ratio of '*contactelephone'* tells us with all else held fixed, we can expect to see about a 67% decrease in subscribing from those contacted by telephone rather than cell phone. We can expect to see the largest increase in the odds of subscribing when a client is contacted in the month of March, followed by December, then September. Likewise, we can expect to see a substantially large increase in subscriptions when the outcome of the previous marketing campaign was a success. Although bank managers and marketing executives can't control '*emp.var.rate'*, they can control other explanatory variables such as what age groups to contact, whether to contact their telephone or mobile phone, which month, and awareness that a positive outcome from a prior campaign will positively influence the likelihood of the client to subscribe to a new campaign.

```
#Compute odds ratios using the exponential function
```{r}
OR = exp(logit.final$coefficients)
round(OR, 3)
```

      (Intercept)   emp.var.rate   age35-50   age50+   contacttelephone
            0.067   0.644   1.040   1.411   0.678
         monthaug   monthdec   monthjul   monthjun   monthmar
            1.235   3.780   1.441   2.119   6.295
         monthmay   monthnov   monthoct   monthsep poutcomenonexistent
            0.667   0.740   1.884   2.224   1.152
  poutcomesuccess
            7.579
```

V. Conclusion

In this study, a dataset from a large Portuguese bank was analyzed and logistic regression and LDA was proposed as an approach for targeting bank telemarketing clients. The goal was to predict the success of subscribing a long-term deposit using attributes that were known before a call is executed. During the model phase a top-down approach was used, a final model was selected based on AIC, final set of 5 features were used and the two models were compared. Ultimately, the best results were obtained from the logistic regression model that binned '*age*' and included feature scaling. The odds ratios were interpreted and give bank managers valuable insight into the features they can control to increase the odds of subscribing to a term deposit. Logistic regression and LDA are two popular statistical tools, but there are many new novel predictive modeling and classification techniques that could also be used, such as: decision tree, neural networks, support vector machines and KNN. For more insight, the bank could still create a model using the features that were discarded for this analysis for future model tuning. Additional modeling related to probability of successfully cross-selling other products could be useful. Lastly, other algorithms could be deployed, including one of the several extensions and variations of LDA such as:

- Quadratic Discriminant Analysis (QDA): Each class uses its own estimate of variance (or covariance when there are multiple input variables).
- Flexible Discriminant Analysis (FDA): Where non-linear combinations of inputs are used such as splines.
- Regularized Discriminant Analysis (RDA): Introduces regularization into the estimate of the variance.

References:

- Machine Learning Blog: https://machinelearning-blog.com/2018/04/23/logistic-regression-101/
- StatisticsSolutions: https://www.statisticssolutions.com/assumptions-of-logistic-regression/
- Machine Learning Mastery: https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/
- James Chen: https://medium.com/@jameschen_78678/which-customers-are-more-likely-to-respond-to-banks-marketing-campaigns-3f00c512268d
- Jason Brownlee: https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/
- Keboola: https://www.keboola.com/blog/logistic-regression-machine-learning
- Clare Liu: https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html
- Olson: https://www.sciencedirect.com/science/article/pii/S0167923612001881
- Coussemant: https://www.sciencedirect.com/science/article/pii/S095741741500456X
- Moro, Cortez, Rita: A Data-Driven Approach to Predict the Success of Bank Telemarketing
- Venkataramana: Regression Analysis with Categorical Variables

APPENDIX A: VARIABLE OVERVIEW

Column Name	Description	Type
age	Age of the client	Numeric
job	Client's occupation	Categorial: • admin • blue-collar • entrepreneur • housemaid • management • retired • self-employed • services • student • technician • unemployed • unknown
marital	Marital status **Note:** divorced means divorced or widowed	Categorial: • divorced • married • single • unknown
education	Client's education level	Categorial: • basic.4y • basic.6y • basic.9y • high.school • illiterate • professional.course • university.degree • unknown
default	Indicates whether the client has credit in default	Categorial: • no • yes • unknown
housing	Indicates whether the client has a housing loan	Categorial: • no • yes

		• unknown
loan	Indicates whether the client as a personal loan	Categorial: • no • yes • unknown
contact	Type of contact communication	Categorial: • cellular • telephone
month	Month that last contact was made	Categorial: • jan • feb • ⋮ • dec
day_of_week	Day that last contact was made	Categorial: • mon • tue • wed • thu • fri
duration	Duration of last contact in seconds **Note:** This attribute highly affects the output target (e.g., if duration=0 then y=no). Yet, the duration is not known before a call is performed. Also, after the end of the call, y is obviously known.	Numeric
campaign	Number of contacts performed during this campaign for this client (including last contact)	Numeric
pdays	Number of days since the client was last contacted in a previous campaign **Note:** 999 means client was not previously contacted	Numeric
previous	Number of contacts performed before this campaign for this client	Numeric
poutcome	Outcome of the previous marketing campaign	Categorial: • failure • nonexistent

		• success
emp.var.rate	Employment variation rate (quarterly indicator)	Numeric
cons.price.idx	Consumer price index (monthly indicator)	Numeric
cons.conf.idx	Consumer confidence index (monthly indicator)	Numeric
euribor3m	Euribor 3-month rate (daily indicator)	Numeric
nr.employed	Number of employees (quarterly indicator)	Numeric

Data Structure

```
'data.frame':	4119 obs. of  21 variables:
 $ emp.var.rate  : num  -1.8 1.1 1.4 1.4 -0.1 -1.1 -1.1 -0.1 -0.1 1.1 ...
 $ cons.price.idx: num  92.9 94 94.5 94.5 93.2 ...
 $ cons.conf.idx : num  -46.2 -36.4 -41.8 -41.8 -42 -37.5 -37.5 -42 -42 -36.4 ...
 $ euribor3m     : num  1.31 4.86 4.96 4.96 4.19 ...
 $ nr.employed   : num  5099 5191 5228 5228 5196 ...
 $ age           : int  30 39 25 38 47 32 32 41 31 35 ...
 $ duration      : int  487 346 227 17 58 128 290 44 68 170 ...
 $ campaign      : int  2 4 1 3 1 3 4 2 1 1 ...
 $ pdays         : int  999 999 999 999 999 999 999 999 999 999 ...
 $ previous      : int  0 0 0 0 0 2 0 0 1 0 ...
 $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 2 8 8 8 1 8 1 3 8 2 ...
 $ marital       : Factor w/ 4 levels "divorced","married",..: 2 3 2 2 2 3 3 2 1 2 ...
 $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 3 4 4 3 7 7 7 7 6 3 ...
 $ default       : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 1 1 1 2 1 2 ...
 $ housing       : Factor w/ 3 levels "no","unknown",..: 3 1 3 2 3 1 3 3 1 1 ...
 $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 2 1 1 1 1 1 1 ...
 $ contact       : Factor w/ 2 levels "cellular","telephone": 1 2 2 2 1 1 1 1 1 2 ...
 $ month         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 5 5 8 10 10 8 8 7 ...
 $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 1 1 5 1 2 3 2 2 4 3 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 1 2 2 1 2 ...
 $ y             : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Data Summary

```
  emp.var.rate        cons.price.idx  cons.conf.idx      euribor3m      nr.employed        age           duration     
 Min.   :-3.40000   Min.   :92.20   Min.   :-50.8   Min.   :0.635   Min.   :4964   Min.   :18.00   Min.   :   0.0  
 1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7   1st Qu.:1.334   1st Qu.:5099   1st Qu.:32.00   1st Qu.: 103.0  
 Median : 1.10000   Median :93.75   Median :-41.8   Median :4.857   Median :5191   Median :38.00   Median : 181.0  
 Mean   : 0.08497   Mean   :93.58   Mean   :-40.5   Mean   :3.621   Mean   :5166   Mean   :40.11   Mean   : 256.8  
 3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4   3rd Qu.:4.961   3rd Qu.:5228   3rd Qu.:47.00   3rd Qu.: 317.0  
 Max.   : 1.40000   Max.   :94.77   Max.   :-26.9   Max.   :5.045   Max.   :5228   Max.   :88.00   Max.   :3643.0  
                                                                                                                   
    campaign          pdays          previous                 job            marital                  education   
 Min.   : 1.000   Min.   :  0.0   Min.   :0.0000   admin.      :1012   divorced: 446   university.degree  :1264  
 1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.0000   blue-collar: 884   married :2509   high.school        : 921  
 Median : 2.000   Median :999.0   Median :0.0000   technician : 691   single  :1153   basic.9y           : 574  
 Mean   : 2.537   Mean   :960.4   Mean   :0.1903   services   : 393   unknown :  11   professional.course: 535  
 3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.0000   management : 324                   basic.4y           : 429  
 Max.   :35.000   Max.   :999.0   Max.   :6.0000   retired    : 166                   basic.6y           : 228  
                                                   (Other)    : 649                   (Other)            : 168  
    default          housing            loan             contact          month       day_of_week        poutcome   
 no     :3315   no     :1839   no     :3349   cellular :2652   may    :1378   fri:768     failure    : 454  
 unknown: 803   unknown: 105   unknown: 105   telephone:1467   jul    : 711   mon:855     nonexistent:3523  
 yes    :   1   yes    :2175   yes    : 665                    aug    : 636   thu:860     success    : 142  
                                                               jun    : 530   tue:841                       
                                                               nov    : 446   wed:795                       
                                                               apr    : 215                                 
                                                               (Other): 203                                 
   y       
 no :3668  
 yes: 451  
```

APPENDIX B: MODEL RESULTS

**For efficiency, alternative models were "recycled" by executing code for a prior model but substituting the variables when needed (i.e. df, train, test datasets). The temporary model's performance metrics were then recorded. If an alternative model resulted in better performance metrics, then it was retained in the final code submission.*

Phase 1: Establish a Baseline Model

QUESTION / OBJECTIVE	MODEL	FORMULA	RETAINED	DATASETS	FEATURE ENGINEERING	SIGNIFICANT PREDICTORS	AIC	BIC	ACCURACY	SENSITIVITY	SPECIFICITY	AUC	FINDINGS
Establish a baseline model	**m1**	m1=glm(y~., data=train1, family = binomial)	**Y**	**train1** **test1**	**None**	**emp.var.rate, age, contact, month, poutcome**	**1860.8**	**2178.0**	**88.94%**	**91.15%**	**48.84%**	**76.90%**	**N/A**

Phase 2: Full Logistic Model with Feature Engineering

QUESTION / OBJECTIVE	MODEL	FORMULA	RETAINED	DATASETS	FEATURE ENGINEERING	SIGNIFICANT PREDICTORS	AIC	BIC	ACCURACY	SENSITIVITY	SPECIFICITY	AUC	FINDINGS
Q1. Is it necessary to create dummy variables?	**m2**	m2=glm(y~., data=train2, family = binomial)	**Y**	df2 train2 test2	Dummy variables	emp.var.rate, age, contact, month, poutcome	1860.8	2178.0	88.94%	91.15%	48.84%	76.90%	Both models performed the same. Continue next steps with df1 (no dummy variables).
Q2. Would binning 'age' help?	**m3**	m3=glm(y~., data=train3, family = binomial)	**Y**	df3 (copy of df1) train3 test3	Binning 'age'	emp.var.rate, age, contact, month, poutcome	1865.6	2189.2	88.94%	91.15%	48.84%	**77.30%**	Binning age had a negative effect on AIC, and had no impact to accuracy, sensitivity, or specificity. However, there was an improvement to AUC.
Q3. Does removing unknown variables improve the model?	**m4**	m4=glm(y~., data=train4, family = binomial)	**Y**	df4 (copy of df1) train4 test4	Remove "unknowns"	emp.var.rate, cons.conf.idx, job, contact, month, poutcome	1443.4	1716.6	87.54%	89.78%	45.16%	74.20%	Produced the lowest AIC, but caused reduction to accuracy & AUC.
	m4 alternative	m4=glm(y~., data=train4, family = binomial)	**N**	df4 (copy of df3) train4 test4	Binning 'age' + Remove "unknowns"	emp.var.rate, cons.price.idx, cons.conf.idx, job, contact, month, poutcome	1446.4	1725.4	87.70%	89.93%	46.88%	74.10%	When m4 was trained with df4 as a copy of df3 instead of df1 (so the model incorporates binning age and removing unknown variables), it produced a lower AIC and slighltly higher performance metrics, but AUC was lower. Because the alternate had a lower AUC than the original m4, it will not be retained in the final code.
Q4. Does removing pdays and default improve model?	**m5**	m5=glm(y~., data=train5, family = binomial)	**N**	df5 (copy of df1) train5 test5	Drop 'pdays' & 'default'	emp.var.rate, age, contact, month, poutcome	1855.8	2154.7	89.31%	**91.29%**	52.38%	76.90%	Produced a lower AIC, and an improvement ito sensitivity and specificity, but same AUC
	m5 alternative	m5=glm(y~., data=train5, family = binomial)	**Y**	df5 (copy of df3) train5 test5	Binning age + Drop 'pdays' & 'default'	emp.var.rate, age, contact, month, poutcome	1860.8	2165.9	89.31%	91.19%	52.50%	77.20%	When df5 incorporates the feature engineering from df3 (binning age), as well as removing 'pdays' and 'default', it produced a higher AIC & BIC, had no impact to accuracy, a negative impact to sensitivity, and a positive impact to specificity and AUC. Because the alternate has a higher AUC, it will be retained in the final code.
Q5. Does binning, removing unknown variables, and removing pdays and default improve the model?	**m6**	m6=glm(y~., data=train6, family = binomial)	**Y**	df6 (copy of df3) train6 test6	Binning 'age' + Remove "unknowns"+ Drop 'pdays' & 'default'	emp.var.rate, cons.price.idx, cons.conf.idx, job, contact, month, poutcome	1443.4	1710.4	87.70%	89.93%	46.88%	74.10%	Produced a better AIC, but accuracy, sensitivity, specificity, and AUC went down.
Q6. Does imputing outliers improve the model?	**m7**	m7=glm(y~., data=train7, family = binomial)	**N**	df7 (copy of df1) train7 test7	Impute outliers	emp.var.rate, age, contact, month, poutcome	1861.4	2178.6	89.19%	91.19%	51.22%	76.90%	AIC and BIC increased. Resulted in second highest accuracy, sensitivity, and specificity seen so far.
	m7 alternative	m7=glm(y~., data=train7, family = binomial)	**Y**	df7 (copy of df3), train7 test7	Binning 'age' + Impute outliers	emp.var.rate, age, contact, month, poutcome	1865.2	2188.5	89.19%	91.18%	51.22%	77.20%	The alternate model, which incorporates the binning of 'age' that occurs in df3, resulted in a higher AIC and no improvement to any of the performance metrics except for AUC. Because the alternate has a higher AUC, it will be retained in the final code.
Q7. Does feature scaling help improve the model?	**m8**	m8=glm(y~., data=train8, family = binomial)	**N**	df8 (copy of df1) train8 test8	Scaling	emp.var.rate, age, contact, month, poutcome	1860.8	2178.0	89.06%	91.06%	50.00%	76.90%	Performed similar or better than the baseline model for all metrics.
	m8 alternative	m8=glm(y~., data=train8, family = binomial)	**Y**	df8 (copy of df3) train8 test8	Binning 'age' + Scaling	emp.var.rate, age, contact, month, poutcome	1865.9	2189.2	**89.55%**	91.21%	**55.26%**	**77.30%**	Removing unknowns and scaling numerical features seemed to have the largest payoff. This model had the highest accuracy, second highest sensitivity rate, the highest specificity and AUC.
	m8 alternative	m8=glm(y~., data=train8, family = binomial)	**N**	df8 (copy of df6) train8 test8	Binning 'age' + Remove "unknowns"+ Drop 'pdays' & 'default' + scaling	emp.var.rate, cons.price.idx, cons.conf.idx, job, contact, month, poutcome	**1443.0**	**1710.4**	87.86%	89.95%	48.39%	74.10%	

Phase 3: Logistic Model Fitting

QUESTION / OBJECTIVE	MODEL	FORMULA	RETAINED	DATASETS	FEATURE ENGINEERING	SIGNIFICANT PREDICTORS	AIC	BIC	ACCURACY	SENSITIVITY	SPECIFICITY	AUC	FINDINGS
Fitted logistic model with significant predictors from baseline model	**m9 aka logit.final**	logit.final=glm(y~emp.var.rate + age + contact + month + poutcome, data=train9, family = binomial)	**Y**	df9 (copy of df8) train8 test8	Binning 'age' + Scaling	emp.var.rate, contact, month, poutcome	1871.6	1969.2	**90.64%**	**91.62%**	**68.57%**	**77.30%**	The fitted model outperforms all other models thus far, resulting in the highest accuracy, sensitivity, and specificity rate as well as AUC (tied with m8).
Fitted logistic model using **optimal threshold**	**logit.final**	logit.final=glm(y~emp.var.rate + age + contact + month + poutcome, data=train9, family = binomial)	**Y**	df9 (copy of df8) train8 test8	Binning 'age' + Scaling	emp.var.rate, contact, month, poutcome	-	-	**71.57%**	**94.95%**	**23.13%**	**77.30%**	Applying the optimal threshold significantly reduced accuracy and specificity, while boosting sensitivity. This model has the highest sensitivity rate so far.

Phase 4: Full LDA Model

QUESTION / OBJECTIVE	MODEL	FORMULA	RETAINED	DATASETS	FEATURE ENGINEERING	SIGNIFICANT PREDICTORS	AIC	BIC	ACCURACY	SENSITIVITY	SPECIFICITY	AUC	FINDINGS
Full LDA Model	**lda.full**	lda.full = lda(y~., data=lda.train)	**N**	df1, df2, df3, df5, df7, df8, df9	None	N/A	-	-	N/A	N/A	N/A	-	Warning message: variables are collinear
Full LDA Model	**lda.full**	lda.full = lda(y~., data=lda.train)	**N**	df4	Remove "unknowns"	N/A	-	-	86.73%	94.12%	32.43%	-	df4 and df6 are the only models that did not return a warning message indicating variables are collinear. What distinguishes these datasets from the rest is that unknown values were removed. The addition of binning 'age' and dropping 'pdays' and 'default' did not improve performance metrics.
Full LDA Model	**lda.full**	lda.full = lda(y~., data=lda.train)	**Y**	df6	Binning 'age' + Remove "unknowns"+ Drop 'pdays' & 'default'	N/A	-	-	86.73%	94.12%	32.43%	-	

Phase 5: LDA Model Fitting

QUESTION / OBJECTIVE	MODEL	FORMULA	RETAINED	DATASETS	FEATURE ENGINEERING	SIGNIFICANT PREDICTORS	AIC	BIC	ACCURACY	SENSITIVITY	SPECIFICITY	AUC	FINDINGS
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**Y**	df1	Binning 'age' + Scaling	N/A	-	-	89.06%	96.45%	28.89%	-	
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df2	Dummy variables	N/A	-	-	88.46%	98.77%	0.04%	-	The only variables included in this model were emp.var.rate and rate. Contact, month, and poutcome were removed as part of the data processing for df2. This led to a much higher sensitivity, but at the cost of an extremely low specificity rate.
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df3	Binning 'age'	N/A	-	-	89.06%	96.45%	28.89%	-	Similar results to df1. This suggests binning doesn't have an impact.
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df4	Remove "unknowns"	N/A	-	-	87.06%	96.32%	18.92%	-	Performed worse than the fitted model with no data transformation
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df5	Binning age + Drop 'pdays' & 'default'	N/A	-	-	89.06%	96.45%	28.89%	-	Similar results to df1. This suggests binning doesn't have an impact. Neither would dropping pdays and default because the model didn't include these variables.
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df6	Binning age + Drop 'pdays' & 'default'	N/A	-	-	87.06%	96.32%	18.92%	-	Performed worse than the fitted model with no data transformation
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df7	Binning 'age' + Impute outliers	N/A	-	-	89.06%	96.45%	28.89%	-	Similar results to df1. This suggests binning doesn't have an impact and neither does imputing outliers.
Fitted LDA Model	**lda.fit**	lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train)	**N**	df8	Binning 'age' + Impute outliers	N/A	-	-	89.06%	96.45%	28.89%	-	Similar results to df1. This suggests binning doesn't have an impact and neither does feature scaling.

APPENDIX C: DATASETS CREATED

df	description	Samples	Variables
raw_data	Original data set	4119	21
df1	Removed duplicates, rearranged columns, drop duration	4119	20
df2	A copy of df1 + encoded with dummy variables	4119	53
df3	A copy of df1 + age binned into 3 groups	4119	20
df4	A copy of df1 + unknowns converted to NA, then removed	3090	20
df5	A copy of df3 + drop 'pdays' and 'default' variables	4119	18
df6	A copy of df3 + remove unknowns + drop 'pdays' and 'default' variables	3090	18
df7	A copy of df3 + impute outliers	4119	20
df8	A copy of df3 + scaling numerical features	4119	20
df9	A copy of df8 with no further changes	4119	20

APPENDIX D: EXPLORATORY DATA ANALYSIS

Figure 1: No Missing Values

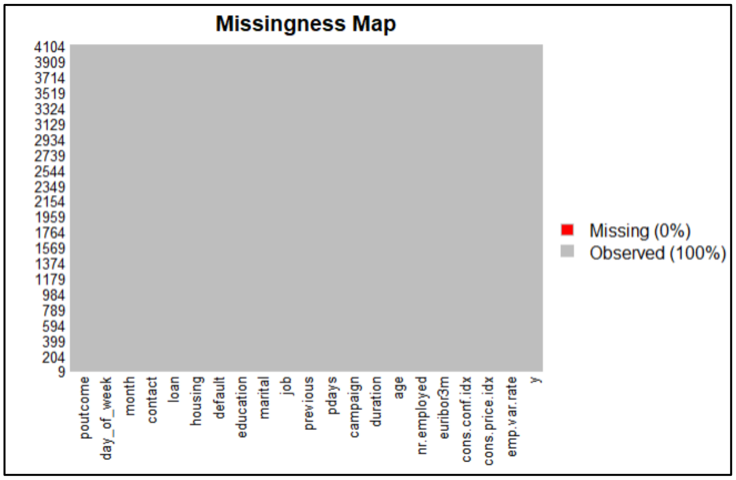


Figure 2: Categorical Predictors

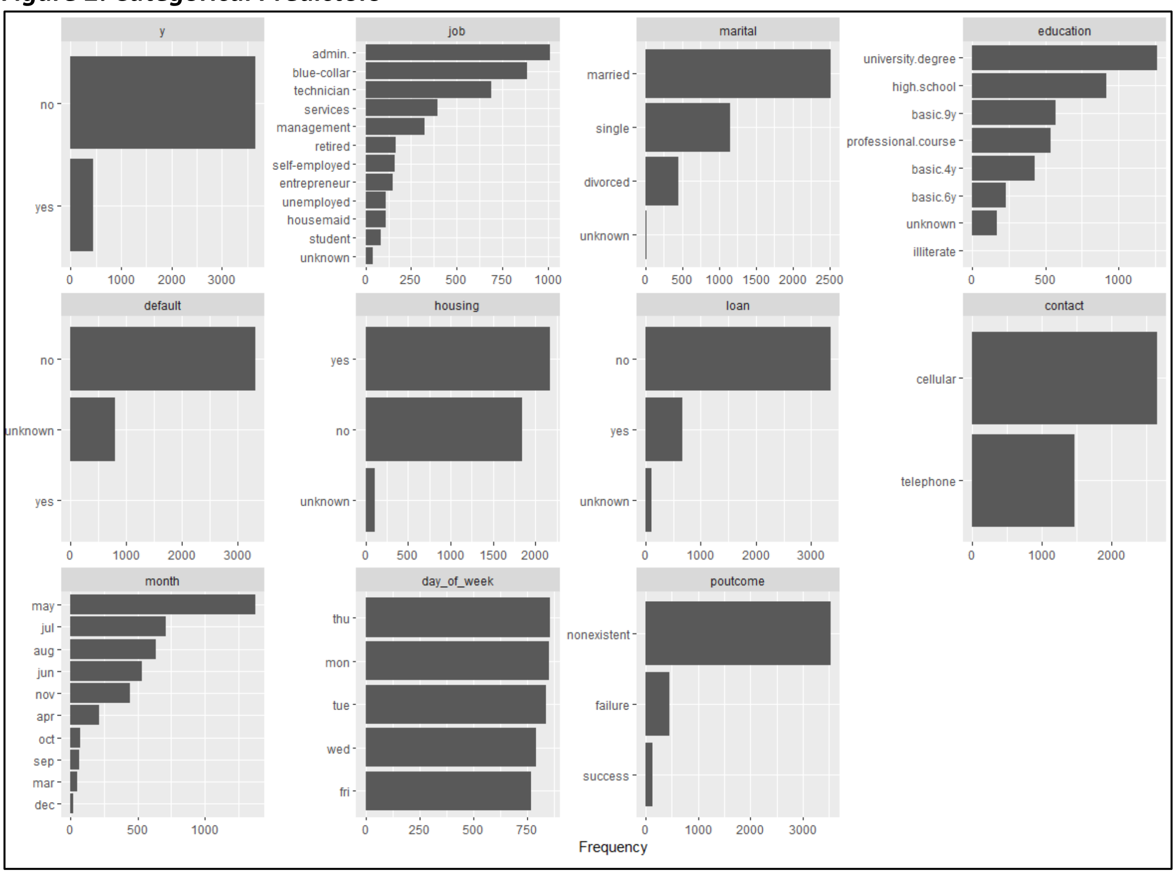


Figure 3: Categorical Predictors by Y

Figure 4: Categorical Predictors by Y

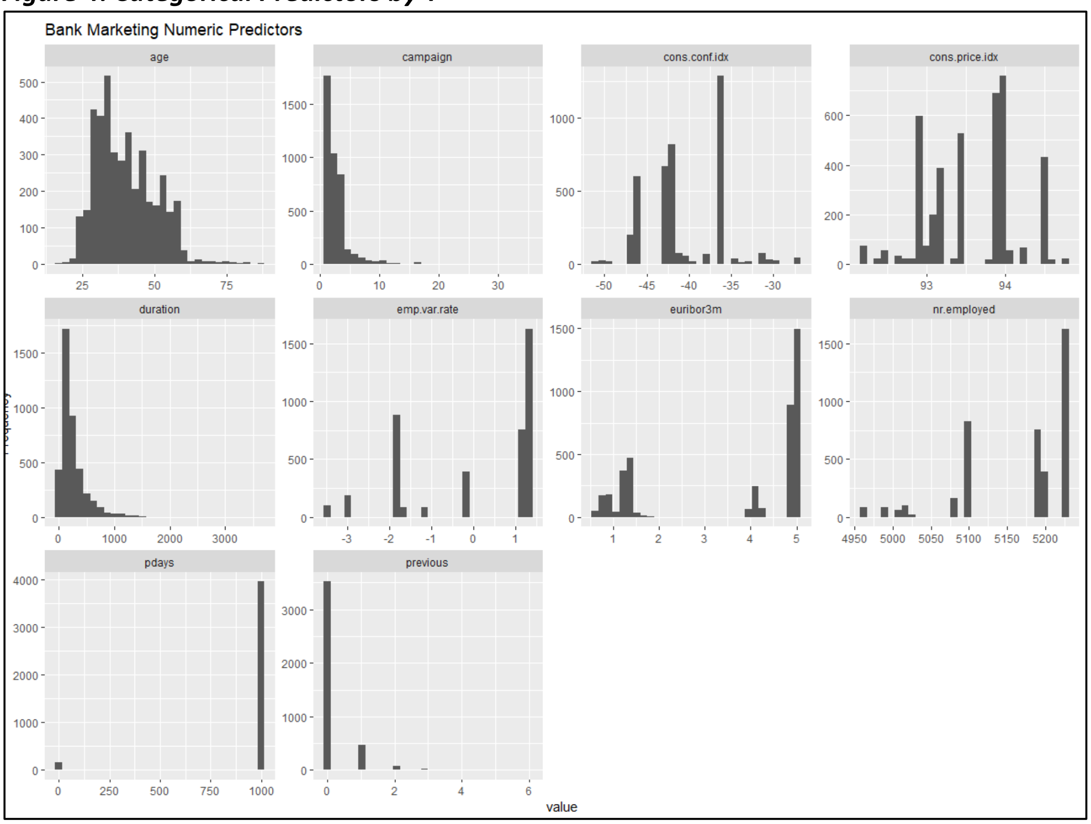
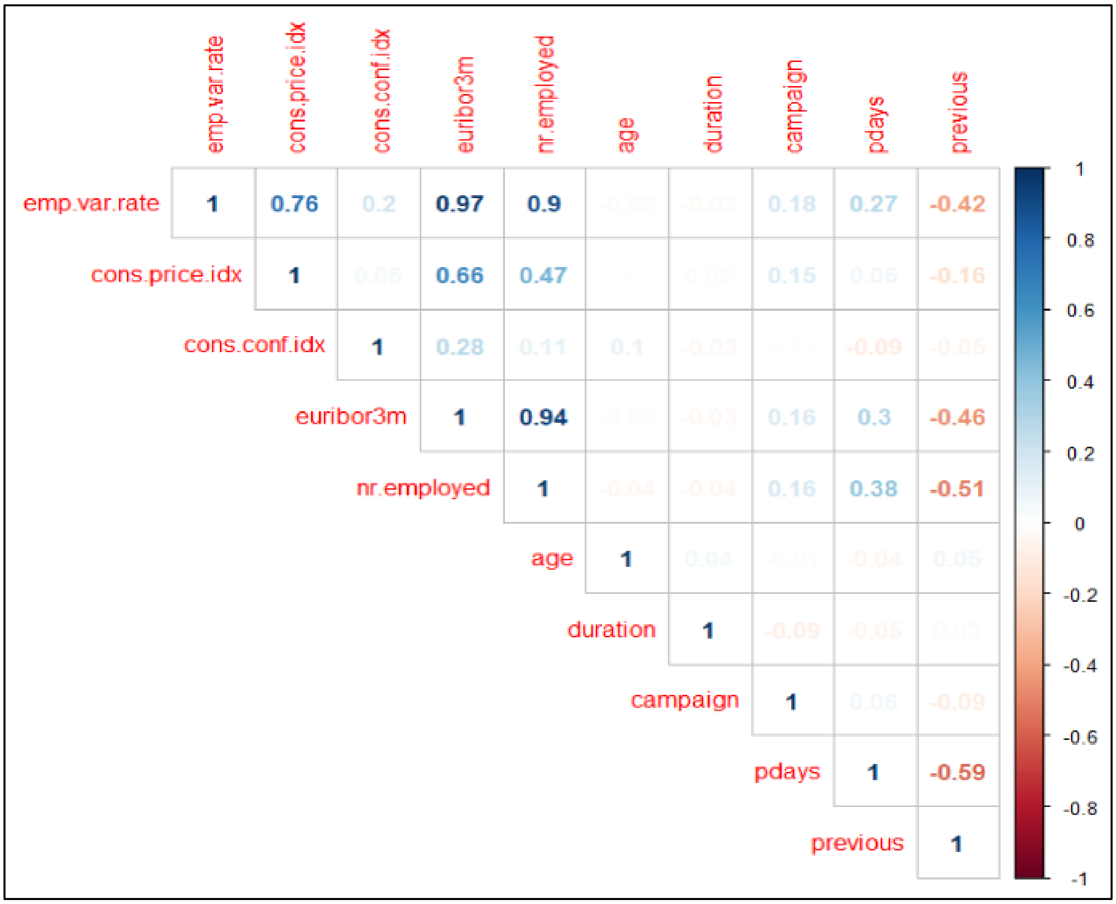


Figure 5: Correlation



APPENDIX E: R CODE

````
---
title: "Bank Marketing Case Study"
author: "Brandi Rodriguez"
date: "February 2021"
output:
  pdf_document: default
---

#LOAD DATA
```{r message=FALSE, warning=FALSE}
rm(list = ls())

#import libraries
library(tidyverse)
library(caret)

#read data
setwd(getwd())
raw_data = read.table('bank-additional.csv', sep=";", header=TRUE)

#remove duplicate records
df1 = distinct(raw_data)

#rearrange columns
df1 = raw_data[, c(21, 16:20, 1, 11:14, 2:10, 15)]

#convert categorical variables to factor variables
df1$job = as.factor(df1$job)
df1$marital = as.factor(df1$marital)
df1$education = as.factor(df1$education)
df1$default = as.factor(df1$default)
df1$housing = as.factor(df1$housing)
df1$loan = as.factor(df1$loan)
df1$contact = as.factor(df1$contact)
df1$month = as.factor(df1$month)
df1$day_of_week = as.factor(df1$day_of_week)
df1$poutcome = as.factor(df1$poutcome)
```

#DATA STRUCTURES AND SUMMARY
```{r}
str(df1)
summary(df1)
```
#MISSING VALUES
The data appeared to have no missing values, but several variables had a level encoded as "unknown."
````

```{r message=FALSE, warning=FALSE}
library(Amelia)
missmap(df1, col = c("red", "gray"))
```

#EXPLORATORY DATA ANALYSIS
```{r message=FALSE, warning=FALSE}
library(DataExplorer)
introduce(df1)
```

```{r}
plot_bar(df1, nrow = 3L, ncol=4L, title = "Bank Marketing Categorical Predictors")
```
The majority of clients have not subscribed to a term deposit. A large portion of clients have administrative and blue-collar jobs, are married, and have a university degree. Several of the categorical predictors contain a level labelled "unknown." These will later be treated as NAs. There were slightly more clients who had a home loan, while a large majority did not have a personal loan. More were contacted by cell phone and most of the prior contacts with them took place during summer months. 'Default' does not appear to be a useful feature. It has three levels, yet only 1 'yes'. It's a good candidate for removal.

```{r}
plot_bar(df1, by="y", nrow = 3L, ncol=4L, title = "Bank Marketing Categorical Predictors by Y")
```
It looks like a larger percent of those last contacted in October, September, March, and December subscribed to a term deposit, while those contacted in the summer months were less likely to subscribe. This is interesting because the summer months were when most of the clients were contacted, while significantly less prior contacts were made in October, September, March and December. Those who were contacted by cell phone rather than telephone, as well as those who had a 'success' as the outcome of the prior marketing campaign were more likely to have subscribed to a term deposit ('poutcome'). The proportion of those who subscribed to a deposit appears to be the same, regardless of day of week ('day_of_week'), whether they had a home loan ('housing'), or personal loan ('loan'). Those contacted by cellular had a larger proportion that subscribed to a term deposit ('contact').

```{r}
plot_histogram(df1, title = "Bank Marketing Numeric Predictors")
```
'Age' is centered around age 30 to 40 with a right skewed distribution. 'Campaign' is right skewed as well, with most clients contacted 5 or less times during the current campaign. All values for 'pdays' appear to have taken a value of 0 or '999', with an overwhelming majority as '999'. Because so many were '999', this feature is a candidate for removal from the final data for model training.

#CORRELATION
```{r}
library(corrplot)
corrplot(cor(df1[, 2:11]), method = "number", type = "upper")
```

The socio-economic features are highly correlated.
  .97: emp.var.rate and euribor3m
  .94: euribor3m and nr.employed
  .90: emp.var.rate and nr.employed
emp.var.rate: employment variation rate - quarterly indicator
euribor3m: euribor 3 month rate - daily indicator
nr.employed: # of employees
For now, these features will be kept in the dataset and may potentially be dropped later on in the analysis if they prove to have high VIFs, indicating the presence of multicollinearity.

#DROP COLUMNS
Need to drop 'duration', because it highly affects the response (i.e. if duration = 0, then y = 'no). Plus, our goal is to create a predictive model and 'duration' is not known before a call is performed.
```{r}
df1 = subset(df1, select = -c(duration)) #can add more columns to drop (separate each with a comma)
```

#DUMMY VARIABLES
In logistic regression models, encoding variables as dummy variables allows easy interpretation and calculation of the odds ratios, and increases the stability and significance of the coefficients (https://stats.idre.ucla.edu/wp-content/uploads/2016/02/p046.pdf).
```{r message=FALSE, warning=FALSE}
library(fastDummies)
dummy_create = fastDummies::dummy_cols(df1, remove_first_dummy=TRUE)
#drop original categorical variables now that they've been encoded as dummy variables
#keep original response variable "y" as a factor (remove dummy variable "y_yes" created)
df2 = subset(dummy_create, select=-c(job, marital, education, default, housing, loan, contact, month,
day_of_week, poutcome, y_yes))
```

#SPLIT TRAIN AND TEST DATASETS
Since there were so few observations where y = "yes", I'm going to do a 80/20 split (as opposed to a 70/30 split) to capture more observations with y = "yes" to train the model on.
```{r message=FALSE, warning=FALSE}
#without dummy variables
set.seed(2021)
index = createDataPartition(df1$y, p=0.8, list = FALSE)
train1 = df1[index,]
test1 = df1[-index,]

#with dummy variables
set.seed(2021)
index = createDataPartition(df2$y, p=0.8, list = FALSE)
train2 = df2[index,]
test2 = df2[-index,]
```

#Q1: IS IT NECESSARY TO CREATE DUMMY VARIABLES?

```{r}
m1=glm(y~., data=train1, family = binomial)
m2=glm(y~., data=train2, family = binomial)
```

Evaluate m1:
```{r message=FALSE, warning=FALSE}
#Which predictors are signifcant and calculate model fit statistics
significant_if = summary(m1)$coeff[-1,4]<.05
m1.significant = names(significant_if)[significant_if ==TRUE]

m1.significant
AIC = AIC(m1)
BIC = BIC(m1)
cbind(AIC, BIC)

#make m1 predictions
library(caret)
test1$PredProb = predict.glm(m1, newdata=test1, type = 'response')
test1$Pred.y = ifelse(test1$PredProb >= .5,1,0)
test1$Pred.y = ifelse(test1$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test1$y), as.factor(test1$Pred.y))

#calculate auc
library(ROCR)
library(pROC)
library(car)
pred1 = prediction(predict(m1, test1, type = "response"), test1$y)
auc1 = round(as.numeric(performance(pred1, measure = "auc")@y.values), 3)
auc1
```

Evaluate m2:
```{r message=FALSE, warning=FALSE}
#significant predictors and some model fit statistics
significant_if = summary(m2)$coeff[-1,4]<.05
m2.significant = names(significant_if)[significant_if ==TRUE]

m2.significant
AIC = AIC(m2)
BIC = BIC(m2)
cbind(AIC, BIC)

#make predictions
library(caret)
test2$PredProb = predict.glm(m2, newdata=test2, type = 'response')
test2$Pred.y = ifelse(test2$PredProb >= .5,1,0)
test2$Pred.y = ifelse(test2$Pred.y == 1, "yes", "no")
```

caret::confusionMatrix(as.factor(test2$y), as.factor(test2$Pred.y))

#calculate AUC
pred2 = prediction(predict(m2, test2, type = "response"), test2$y)
auc2 = round(as.numeric(performance(pred2, measure = "auc")@y.values), 3)
auc2
```

#Q2. DOES BINNING IMPROVE PREDICTIVENESS?
Having a smaller number of age groups that are far more statistically significant than each year evaluated separately (*https://stats.idre.ucla.edu/wp-content/uploads/2016/02/p046.pdf).
```{r}
plot_histogram(df1$age)
```

Create bins of approximately equal width:
https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781783989065/1/ch01lvl1sec20/binning-numerical-data
```{r message=FALSE, warning=FALSE}
df3 = df1 #switch to another df for an alternative model
b = c(-Inf, 30, 45, Inf) #create breaks to infer bins
df3$age = cut(df3$age, breaks = b)
summary(df3$age)
levels(df3$age) = c("<35", "35-50", "50+")
summary(df3$age)

#alternative way to bin:
  #df3$age = cut(df3$age, breaks = 4, labels = c("AgeGroup1", "AgeGroup2", "AgeGroup3", "AgeGroup4"))
  #summary(df3$age) #doesn't tell you what ages are in each age group, just a count
```

```{r message=FALSE, warning=FALSE}
#create test and training dataset
set.seed(2021)
index = createDataPartition(df3$y, p=0.8, list = FALSE)
train3 = df3[index,]
test3 = df3[-index,]

#train model
m3=glm(y~., data=train3, family = binomial)

#significant predictors and some model fit statistics
significant_if = summary(m3)$coeff[-1,4]<.05
m3.significant = names(significant_if)[significant_if ==TRUE]

m3.significant
AIC = AIC(m3)

BIC = BIC(m3)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test3$PredProb = predict.glm(m3, newdata=test3, type = 'response')
test3$Pred.y = ifelse(test3$PredProb >= .5,1,0)
test3$Pred.y = ifelse(test3$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test3$y), as.factor(test3$Pred.y))

#calculate AUC
pred3 = prediction(predict(m3, test3, type = "response"), test3$y)
auc3 = round(as.numeric(performance(pred3, measure = "auc")@y.values), 3)
auc3
```

#Q3. DOES REMOVING UNKNOWN VARIABLES IMPROVE MODEL RESULT?
```{r}
colSums(df1 == "unknown")
```

% of Unknowns by Column
```{r}
df1 %>%
  summarise_all(list(~mean(. == "unknown"))) %>%
  gather(key = "variable", value = "Unknown_Percent") %>%
  arrange(-Unknown_Percent) %>%
  head(10)
```
About a fifth of all values for default were unknown.

Create a new dataset with unknowns removed.
```{r message=FALSE, warning=FALSE}
df4 = df1 #switch to another df for an alternative model
df4[df4 == "unknown"] <- NA #convert unknowns to NA
df4 = drop_na(df4) #remove NAs
```

Check the data structures after converting to NAs. The new data frame still has the same number of levels as before, despite removing all unknowns.
```{r}
str(df4)
```

Since the categorical columns originally had "unknown" as a string, R recognizes the NAs as strings. We need to correct this assumption by telling R that it's a column of integers, then convert to factor variables.   (https://www.youtube.com/watch?v=C4N3_XJJ-jU @ 3:00).
```{r message=FALSE, warning=FALSE}

df4$default = as.integer(df4$default)
df4$default = as.factor(df4$default)
df4$education = as.integer(df4$education)
df4$education = as.factor(df4$education)
df4$housing = as.integer(df4$housing)
df4$housing = as.factor(df4$housing)
df4$loan = as.integer(df4$loan)
df4$loan = as.factor(df4$loan)
df4$job = as.integer(df4$job)
df4$job = as.factor(df4$job)
df4$marital = as.integer(df4$marital)
df4$marital = as.factor(df4$marital)
str(df4)
```

Now the # of levels in the categorical variables is 1 less than what it was before. However, the labels for categorical variables that had NAs removed are now replaced with numeric labels.
```{r}
table(df1$default)
table(df4$default)
```

Relabel the levels
```{r message=FALSE, warning=FALSE}
levels(df4$default) = c("no", "yes")
levels(df4$education) = c("basic.4y","basic.6y", "basic.9y","high.school", "illiterate",
"professional.course", "university.degree" )
levels(df4$housing) = c("no", "yes")
levels(df4$loan) = c("no", "yes")
levels(df4$job) = c("admin", "blue-collar", "entrepreneur", "housemaid", "management", "retired",
"self-employed", "services", "student", "technician", "unemployed")
levels(df4$marital) = c("divorced", "married", "single")
```

Double check
```{r}
table(df1$marital)
table(df4$marital)
```

```{r message=FALSE, warning=FALSE}
#create test and training dataset
set.seed(2021)
index = createDataPartition(df4$y, p=0.8, list = FALSE)
train4 = df4[index,]
test4 = df4[-index,]

#train model

~~~
m4=glm(y~., data=train4, family = binomial)

#significant predictors and some model fit statistics
significant_if = summary(m4)$coeff[-1,4]<.05
m4.significant = names(significant_if)[significant_if ==TRUE]

m4.significant
AIC = AIC(m4)
BIC = BIC(m4)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test4$PredProb = predict.glm(m4, newdata=test4, type = 'response')
test4$Pred.y = ifelse(test4$PredProb >= .5,1,0)
test4$Pred.y = ifelse(test4$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test4$y), as.factor(test4$Pred.y))

#calculate AUC
pred4 = prediction(predict(m4, test4, type = "response"), test4$y)
auc4 = round(as.numeric(performance(pred4, measure = "auc")@y.values), 3)
auc4
```
~~~

#Q4: DOES DROPPING PDAYS AND DEFAULT HELP THE MODEL?

Taking a closer look at pdays and default, the vast majority of values observed for pdays was 999 values. This column is a candidate for removal, as well as default, which had 3 levels, "no", "unknown" and "yes", but only one observed "yes."

~~~
```{r}
df5 = df3 #switch to another df for an alternative model
table(df5$pdays)
table(df5$default)
```

```{r message=FALSE, warning=FALSE}
#drop columns
df5 = subset(df5, select = -c(pdays, default))

#create test and training dataset
set.seed(2021)
index = createDataPartition(df5$y, p=0.8, list = FALSE)
train5 = df5[index,]
test5 = df5[-index,]

#train model
m5=glm(y~., data=train5, family = binomial)
~~~

````
#significant predictors and some model fit statistics
significant_if = summary(m5)$coeff[-1,4]<.05
m5.significant = names(significant_if)[significant_if ==TRUE]

m5.significant
AIC = AIC(m5)
BIC = BIC(m5)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test5$PredProb = predict.glm(m5, newdata=test5, type = 'response')
test5$Pred.y = ifelse(test5$PredProb >= .5,1,0)
test5$Pred.y = ifelse(test5$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test5$y), as.factor(test5$Pred.y))

pred5 = prediction(predict(m5, test5, type = "response"), test5$y)
auc5 = round(as.numeric(performance(pred5, measure = "auc")@y.values), 3)
auc5
```
````

#Q5: DOES BINNING AGE, REMOVING UNKNOWN VARIABLES, AND DROPPING COLUMNS IMPROVE THE MODEL?

Incorporate binning of age by making a copy of df3, where binning was first tested

````
```{r message=FALSE, warning=FALSE}```{r}
df6 = df3 #switch to another df for an alternative model

#convert unknowns to NA, then remove
df6[df6 == "unknown"] <- NA
df6 = drop_na(df6)

#reconvert factor variables
df6$default = as.integer(df6$default)
df6$default = as.factor(df6$default)
df6$education = as.integer(df6$education)
df6$education = as.factor(df6$education)
df6$housing = as.integer(df6$housing)
df6$housing = as.factor(df6$housing)
df6$loan = as.integer(df6$loan)
df6$loan = as.factor(df6$loan)
df6$job = as.integer(df6$job)
df6$job = as.factor(df6$job)
df6$marital = as.integer(df6$marital)
df6$marital = as.factor(df6$marital)

#Relabel the levels
levels(df6$default) = c("no", "yes")
````

````
levels(df6$education) = c("basic.4y","basic.6y", "basic.9y","high.school", "illiterate",
"professional.course", "university.degree" )
levels(df6$housing) = c("no", "yes")
levels(df6$loan) = c("no", "yes")
levels(df6$job) = c("admin", "blue-collar", "entrepreneur", "housemaid", "management", "retired",
"self-employed", "services", "student", "technician", "unemployed")
levels(df6$marital) = c("divorced", "married", "single")

#drop columns
df6 = subset(df6, select = -c(pdays, default))

#create test and training dataset
set.seed(2021)
index = createDataPartition(df6$y, p=0.8, list = FALSE)
train6 = df6[index,]
test6 = df6[-index,]

#train model
m6=glm(y~., data=train6, family = binomial)

#significant predictors and some model fit statistics
significant_if = summary(m6)$coeff[-1,4]<.05
m6.significant = names(significant_if)[significant_if ==TRUE]

m6.significant
AIC = AIC(m6)
BIC = BIC(m6)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test6$PredProb = predict.glm(m6, newdata=test6, type = 'response')
test6$Pred.y = ifelse(test6$PredProb >= .5,1,0)
test6$Pred.y = ifelse(test6$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test6$y), as.factor(test6$Pred.y))


#AUC
pred6 = prediction(predict(m6, test6, type = "response"), test6$y)
auc6 = round(as.numeric(performance(pred6, measure = "auc")@y.values), 3)
auc6
```

#Q6: DOES IMPUTING OUTLIERS IMPROVE MODEL RESULTS?
```{r}
df7 = df3 #switch to another df for an alternative model
```
````
````

````
```{r}
str(df7)
```


```{r message=FALSE, warning=FALSE}
library(dlookr)
df7 %>%
  plot_outlier(emp.var.rate) #no apparent outliers

df7 %>%
  plot_outlier(cons.price.idx) #no apparent outliers

df7 %>%
  plot_outlier(cons.conf.idx)

df7 %>%
  plot_outlier(euribor3m)

df7 %>%
  plot_outlier(nr.employed)

df7 %>%
  plot_outlier(campaign)

df7 %>%
  plot_outlier(pdays)

df7 %>%
  plot_outlier(previous)
```
````

The capping method imputes the upper outliers with 95 percentile and imputes the bottom outliers with 5 percentile.

````
```{r message=FALSE, warning=FALSE}
par(mfrow=c(2,4))

df7$emp.var.rate = imputate_outlier(df7, emp.var.rate , method = "capping")
plot(df7$emp.var.rate, main="emp.var.rate")

df7$cons.price.idx = imputate_outlier(df7, cons.price.idx , method = "capping")
plot(df7$cons.price.idx)

df7$cons.conf.idx = imputate_outlier(df7, cons.conf.idx , method = "capping")
plot(df7$cons.conf.idx)

df7$euribor3m = imputate_outlier(df7, euribor3m , method = "capping")
plot(df7$euribor3m)
````
````

````
df7$nr.employed = imputate_outlier(df7, nr.employed , method = "capping")
plot(df7$nr.employed)

df7$pdays = imputate_outlier(df7, campaign , method = "capping")
plot(df7$pdays)
```

```{r message=FALSE, warning=FALSE}
#create test and training dataset
set.seed(2021)
index = createDataPartition(df7$y, p=0.8, list = FALSE)
train7 = df7[index,]
test7 = df7[-index,]

#train model
m7=glm(y~., data=train7, family = binomial)

#significant predictors and some model fit statistics
significant_if = summary(m7)$coeff[-1,4]<.05
m7.significant = names(significant_if)[significant_if ==TRUE]

m7.significant
AIC = AIC(m7)
BIC = BIC(m7)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test7$PredProb = predict.glm(m7, newdata=test7, type = 'response')
test7$Pred.y = ifelse(test7$PredProb >= .5,1,0)
test7$Pred.y = ifelse(test7$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test7$y), as.factor(test7$Pred.y))

#AUC
pred7 = prediction(predict(m7, test7, type = "response"), test7$y)
auc7 = round(as.numeric(performance(pred7, measure = "auc")@y.values), 3)
auc7
```

#Q7. DOES FEATURE SCALING IMPROVE MODEL ACCURACY?
Using the original scale may put more weight on variables with larger ranges, resulting in
disproportionate influence. Feature scaling can be used to bring all values to the same magnitudes to
solve this issue.
```{r message=FALSE, warning=FALSE}
df8 = df3 #switch to another df for an alternative model
#create test and training dataset
set.seed(2021)
````

~~~~
index = createDataPartition(df8$y, p=0.8, list = FALSE)
train8 = df8[index,]
test8 = df8[-index,]
```

```{r message=FALSE, warning=FALSE}
#may need to adjust these variables, depending on the model used to create a copy of for df8
train8 = train8%>%
  mutate_at(c("emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed",
"campaign","previous"), scale)

test8 = test8%>%
  mutate_at(c("emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed",
"campaign", "previous"), scale)
```

```{r message=FALSE, warning=FALSE}
library(gridExtra)
dp1 = ggplot(train1, aes(x=emp.var.rate))+
  geom_density(color = "black", fill = "gray") +geom_vline(aes(xintercept = mean(age)), color = "red",
linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(age)), color = "blue", linetype = 4,
size =1)
dp1

dp2 = ggplot(train8, aes(x=emp.var.rate))+
  geom_density(color = "black", fill = "gray") +geom_vline(aes(xintercept = mean(emp.var.rate)), color =
"red", linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(emp.var.rate)), color =
"blue", linetype = 4, size =1)
dp2


dp3 = ggplot(train1, aes(x=cons.conf.idx))+
  geom_density(color = "black", fill = "gray") +geom_vline(aes(xintercept = mean(cons.conf.idx)), color =
"red", linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(cons.conf.idx)), color =
"blue", linetype = 4, size =1)
dp3

dp4 = ggplot(train8, aes(x=cons.conf.idx))+
  geom_density(color = "black", fill = "gray") +geom_vline(aes(xintercept = mean(cons.conf.idx)), color =
"red", linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(cons.conf.idx)), color =
"blue", linetype = 4, size =1)
dp4

dp5 = ggplot(train1, aes(x=cons.price.idx))+
  geom_density(color = "black", fill = "gray") +geom_vline(aes(xintercept = mean(cons.price.idx)), color =
"red", linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(cons.price.idx)), color =
"blue", linetype = 4, size =1)
dp5
~~~~

```
dp6 = ggplot(train8, aes(x=cons.price.idx))+
  geom_density(color = "black", fill = "gray") +geom_vline(aes(xintercept = mean(cons.price.idx)), color = "red", linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(cons.price.idx)), color = "blue", linetype = 4, size =1)
dp6

grid.arrange(dp1, dp2, dp3, dp4, dp5, dp6, ncol=2)
rm(dp1, dp2, dp3, dp4, dp5, dp6)
```

Check if variance = 1

```{r message=FALSE, warning=FALSE}
sd(train8$campaign)
sd(train8$previous)
sd(train8$cons.price.idx)
sd(train8$euribor3m)
sd(train8$nr.employed)
```

```{r message=FALSE, warning=FALSE}
#train model
m8=glm(y~., data=train8, family = binomial)

#significant predictors and some model fit statistics
significant_if = summary(m8)$coeff[-1,4]<.05
m8.significant = names(significant_if)[significant_if ==TRUE]

m8.significant
AIC = AIC(m8)
BIC = BIC(m8)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test8$PredProb = predict.glm(m8, newdata=test8, type = 'response')
test8$Pred.y = ifelse(test8$PredProb >= .5,1,0)
test8$Pred.y = ifelse(test8$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test8$y), as.factor(test8$Pred.y))

#AUC
pred8 = prediction(predict(m8, test8, type = "response"), test8$y)
auc8 = round(as.numeric(performance(pred8, measure = "auc")@y.values), 3)
auc8
```

Proceed with model 8, which had the highest AUC.

#FITTED MODEL
m8, which removes unknowns and includes feature scaling has outperformed all other variations of the baseline model so far. It will now be fitted with only the significant predictors.
```{r message=FALSE, warning=FALSE}
df9 = df8 #switch to another df for an alternative model
#create test and training dataset
set.seed(2021)
index = createDataPartition(df9$y, p=0.8, list = FALSE)
train9 = df9[index,]
test9 = df9[-index,]

#train fitted model
m9=glm(y~emp.var.rate + age + contact + month + poutcome, data=train9, family = binomial)

#significant predictors and some model fit statistics
significant_if = summary(m9)$coeff[-1,4]<.05
m9.significant = names(significant_if)[significant_if ==TRUE]

m9.significant
AIC = AIC(m9)
BIC = BIC(m9)
cbind(AIC, BIC)

#make predictions and evaluate performance metrics
library(caret)
test9$PredProb = predict.glm(m9, newdata=test9, type = 'response')
test9$Pred.y = ifelse(test9$PredProb >= .5,1,0)
test9$Pred.y = ifelse(test9$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(test9$y), as.factor(test9$Pred.y))

#AUC
pred9 = prediction(predict(m9, test9, type = "response"), test9$y)
auc9 = round(as.numeric(performance(pred9, measure = "auc")@y.values), 3)
auc9
```

```{r message=FALSE, warning=FALSE}
summary(m9)
```

#ESTABLISH FINAL LOGISTIC MODEL
renaming final logistic model and datasets for easier integration #incase we want to view output of code below for a different model.
```{r message=FALSE, warning=FALSE}
logit.final = m9
logit.df = df9
logit.train = train9
logit.test = test9
```

~~~
logit.pred = pred9
logit.auc = auc9
```
~~~

#Compute odds ratios using the exponential function

~~~
```{r message=FALSE, warning=FALSE}
OR = exp(logit.final$coefficients)
round(OR, 3)
```
~~~

The fitted model tells us there's a negative association between emp.var.rate and those who subscribe. The odds ratio of .644 tells us that holding all other predictors fixed, we expect to see about a 64% decrease in the odds of subscribing to a term deposit for a one unit increase in emp.var.rate. The .678 odds ratio of contactelephone tells us with all else held fixed, we can expect to see about a 67% decrease in subscribing from those contacted by telephone rather than cell phone. We can expect to see the largest increase in the odds of subscribing when a client is contacted in the month of March, followed by December, then September. We can expect to see a substantially large increase in subscriptions when the outcome of the previous marketing campaign was a success.

#CHECK ASSUMPTIONS

Now that we have established our final logistic model, we need to check the assumptions.

#Linearity of x with logit of y (applied to numerical variables only).

The plot will show us if there is a linear relationship.

~~~
```{r message=FALSE, warning=FALSE}
#predict probability of y
probabilities <- predict(logit.final, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)

#Select only numeric predictors
mydata = dplyr::select_if(logit.train, is.numeric)
predictors <- colnames(mydata)

#bind the logit and tidy the data for plotting
mydata = mutate(mydata, logit = log(probabilities/(1-probabilities)))
mydata = gather(mydata, key = "predictors", value = "predictor.value", -logit)

#plot
ggplot(mydata, aes(logit, predictor.value))+
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(method = "loess") +
  theme_bw() +
  facet_wrap(~predictors, scales = "free_y")
```
~~~

#Check for Influential variables

~~~
```{r message=FALSE, warning=FALSE}
plot(logit.final, which = 4, id.n = 2) #cook's distance
```
~~~

#Check Standardized Residuals
Extract model results to compute std. residuals
```{r message=FALSE, warning=FALSE}
library(broom) #package containing the augment function needed
logit.final.data <- augment(logit.final)
top_n(logit.final.data, 2, .cooksd)
```

Plot Standardized Residuals
```{r message=FALSE, warning=FALSE}
#add a column to identify rows
id = rownames(logit.final.data)
logit.final.data = cbind(id=id, logit.final.data)

ggplot(logit.final.data, aes(id, .std.resid)) +
  geom_point(aes(color = y), alpha = .5) +
  theme_bw()
```

```{r message=FALSE, warning=FALSE}
# find data points with an absolute standardized residuals above 3: outliers
filter(logit.final.data, abs(.std.resid) > 3) # none exists
```
All absolute standardized residuals were below 3, indicating there are no outliers.

#Check for Multicollinearity
VIFj > 10 means at least 90% of xj is explained by other predictors. Remove until all VIFs are < 10. This is how to handle/treat multicollinearity, which causes predictors to conform to each other and less reliable predictions.
```{r message=FALSE, warning=FALSE}
vif(logit.final)
```

The final model does not display signs of multicollinearity. All VIFs were reasonable.

#ROC Curve and ROC
```{r message=FALSE, warning=FALSE}
#pred and auc were previously calculated and stored as logitpred and logit auc

#calculate statistics
logit.false.rates = performance(logit.pred, "fpr", "fnr")
logit.accuracy = performance(logit.pred, "acc", "err")
logit.perf = performance(logit.pred, "tpr", "fpr")

#plot ROC curve and AUC
plot(logit.perf, colorize = T, main = "ROC Curve")
text(.5, .5, paste("AUC:", logit.auc))
```

~~~
```

#Compute Optimal Threshold
```{r message=FALSE, warning=FALSE}
#first calculate sensitivity
plot(unlist(performance(logit.pred, "sens")@x.values), unlist(performance(logit.pred, "sens")@y.values),
    type="l", lwd=2,
    ylab="Sensitivity", xlab="Cutoff", main = paste("Maximized Cutoff\n","AUC: ",logit.auc))

par(new=TRUE) # plot another line in same plot

#second specificity
plot(unlist(performance(logit.pred, "spec")@x.values), unlist(performance(logit.pred, "spec")@y.values),
    type="l", lwd=2, col='red', ylab="", xlab="")
axis(4, at=seq(0,1,0.2)) #specificity axis labels
mtext("Specificity",side=4, col='red')

#find where the lines intersect
min.diff <-which.min(abs(unlist(performance(logit.pred, "sens")@y.values) -
unlist(performance(logit.pred, "spec")@y.values)))
min.x<-unlist(performance(logit.pred, "sens")@x.values)[min.diff]
min.y<-unlist(performance(logit.pred, "spec")@y.values)[min.diff]
logit.optimal <-min.x #this is the optimal points to best trade off sensitivity and specificity

abline(h = min.y, lty = 3)
abline(v = min.x, lty = 3)
text(min.x,0,paste("optimal threshold=",round(logit.optimal,2)), pos = 4)
```

#Rerun Model with Optimal Threshold
```{r message=FALSE, warning=FALSE}
#make new predictions with optimal threshold and evaluate performance metrics
library(caret)
logit.test$PredProb = predict.glm(logit.final, newdata=logit.test, type = 'response')
logit.test$Pred.y = ifelse(logit.test$PredProb >= .08,1,0)
logit.test$Pred.y = ifelse(logit.test$Pred.y == 1, "yes", "no")
caret::confusionMatrix(as.factor(logit.test$y), as.factor(logit.test$Pred.y))
```
~~~

Applying the optimal threshold significantly reduced accuracy from 90.64% to 71.57% and specificity from 68.57% to 23.13%, while boosting sensitivity of the final model from 91.62% to 94.95%. The final model will revert back to final.logit with a .5 threshold.

~~~
#FULL LDA Model
```{r message=FALSE, warning=FALSE}
library(MASS)
#switch out different df's featuring different data prep/feature engineering to view & record results
lda.data = df6#
lda.train = train6
~~~

````
lda.test = test6
set.seed(2021)

#train model
lda.full = lda(y~., data=lda.train)

#make predictions
predictions.lda.full = predict(lda.full, newdata=lda.test)
summary(predictions.lda.full$class)

#confusion matrix
caret::confusionMatrix(predictions.lda.full$class, lda.test$y)

#AUC
#lda.full.pred = prediction(predict(lda.full, lda.test, type = "response"), lda.test$y)
#lda.full.auc = round(as.numeric(performance(lda.full.pred, measure = "auc")@y.values), 3)
#lda.full.auc
```



#FITTED LDA MODEL
Fit an LDA model using the same predictors as the final logistic model
```{r message=FALSE, warning=FALSE}
lda.fit.data = df1
lda.fit.train = train1
lda.fit.test = test1
set.seed(2021)

#train model
lda.fit = lda(y~emp.var.rate + age + contact + month + poutcome, data=lda.fit.train) #add back contact,
month, poutcome

#make predictions
predictions.lda.fit = predict(lda.fit, newdata=lda.fit.test)
summary(predictions.lda.fit$class)

#confusion matrix
caret::confusionMatrix(predictions.lda.fit$class, lda.fit.test$y)

#AUC
#lda.fit.pred = prediction(predict(lda.fit, lda.fit.test, type = "response"), lda.fit.test$y)
#lda.fit.auc = round(as.numeric(performance(lda.fit.pred, measure = "auc")@y.values), 3)
#lda.fit.auc
```
````
````