Bookbinders Case Study

Executive Summary

In our case study, we aim to predict whether the members of the Bookbinders Book Club (BBBC) will purchase *The Art History of Florence*. The aim for this case study is two things: to better understand what factors influence a customer to purchase the book, and to develop models that will accurately predict the outcome.

Our focus is on linear regression, logistic regression, and support vector machine models (SVM) for classification. Logistic regression provides probabilities, while SVM finds the best separation between purchase and non-purchase groups to output a prediction label. We reviewed relevant literature highlighting the suitability of logistic regression and SVM. Methodologically, we cleaned and balanced BBBC data for better model performance. Balancing data significantly improved sensitivity and profitability, emphasizing the importance of data quality. We recommend focusing on models trained with balanced data for future marketing strategies.

Additionally, we propose automating future modeling efforts, integrating streamlined processes for data cleaning, feature selection, model training, and monitoring. This forward-thinking approach ensures ongoing relevance, efficiency, and effectiveness of predictive models, supporting informed marketing strategies and driving profitability for the company. Further details on methodology, results, conclusions, and recommendations are provided in the subsequent sections.

Problem

The task we were given for this case study was to predict if a member of the Bookbinders Book Club (BBBC) would purchase the book *The Art History of Florence*. Because the task is classification oriented, we will mainly explore logistic models and support vector machine models. We will ultimately choose a model that includes factors that influence customers to buy the book and is highly accurate. We will also examine a linear regression model and explain why this method is not effective for classification. Our problem encompasses two main aspects: first, to understand the factors that influence a member's decision to buy the book, and second, to develop robust predictive models that accurately forecast this outcome. The outcome variable, *Choice*, serves as the focal point for prediction, while a set of predictor variables offer insights into the client's purchasing habits and gender.

Ultimately, the successful development and validation of predictive models can equip BBBC with valuable insights to increase profits through more targeted mailing campaigns with an improved response rate. These insights will improve the efficacy of their direct mail program, and in turn help inform their company's strategy as they plan for the future. The remainder of this report will include a review of related literature, insights into methodology used, details regarding data preparation and manipulation, and an analysis of our models and findings.

Review of Related Literature

In a 2017 study by Jincheng Cao of Stanford University[1], various predictive models (referred to as response models) were utilized to rank the order of consumers likelihood to respond to a direct mail campaign for a lending institution. The response models accessed in the study includes logistic regression and support vector machines (SVM), which will be described further in *Methodology*. Additionally, Cao employed gradient boosting trees, which is a binary classifier that groups numerous shallow decision trees with the same depth, trains using an optimization algorithm, and outputs a probability (Cao, 2017, p. 2). Finally, the study tested a neural network, a deep learning technique that processes information through layers of neurons (Cao, 2017, p. 2). Cao built a two layered neural network with a sigmoid activation function and trained it using "batch gradient descent algorithm through forward- and backward-propagation" (Cao, 2017, p. 3).

The study found that the response model that is least applicable to the lending industry is the SVM because it outputs 0s and 1s rather than a probability. Opposite, the three other methods provide probabilistic outputs appropriate to the industry need, and they were evaluated on log-loss, "a measure of distance between predicted class versus actual class of an instance" (Cao, 2017, p. 3), and on AUC, "the area under the ROC curve, which plots the true positive rate against false positive rate for varying thresholds" (Cao, 2017, p. 3). The study concluded that the preferred method depends on the circumstances. When training samples are limited, the Gradient Boosting Tree yields the best performance, granted that its parameters are carefully tuned to prevent overfitting. When a simpler and more interpretable model is desired, logistic regression serves as a good choice. Lastly, Cao suggested the neural network when the training set is large (Cao, 2017, p. 5).

[1] https://cs229.stanford.edu/proj2017/final-reports/5242050.pdf. Accessed Feb 24, 2024.

Methodology

Our analysis will access three modeling methods on the request of BBBC: Linear regression, logistic regression, and support vector machines (SVM). As described in the *Problem* section above, the company wants to increase their response rate by selecting which customers to target based on a prediction whether the customer will make a purchase (1) or not (0). In other words, the predictor variable, *Choice*, is binary, making the problem a classification task. While logistic regression and SVM are methods commonly used for classification tasks, fitting a linear regression on a binary predictor does not result in a feasible output. This is because linear regression assumes a continuous response variable, and consequently, it may predict values less than 0 and greater than 1.

In contrast, logistic regression returns a *probability* of an event occurring, making the output between 0 and 1. In this case, an event is when *Choice* is 1, indicating that the customer will make a purchase. Like linear regression, logistic regression is highly interpretable compared to other machine learning methods, as they provide beta coefficients that can explain each predictor's effect on the response variable. Furthermore, we can apply stepwise selection to isolate the predictors that are the most significant predictors of *Choice*. In this analysis, we will use the AIC criterion to select predictors, and we will compare results to the logit model using all predictors. Some of the limitations of logistic regression results from its numerous assumptions about the data, such as 1) linearity in the logit function for continuous variables, 2) no multicollinearity among the variables, and 3) no highly influential outliers.

Conversely, SVM is a non-parametric method that does not entail any assumption about the distribution of the data. A support vector machine seeks to find the hyperplane that best separates the two classes from each other. While SVM in theory assumes linear separability, using kernel tricks to transform the data into a higher dimensional space allows the SVM to handle nonlinearly separable points. In our analysis, we will experiment with a linear kernel as well as RBF, polynomial, and sigmoid. Support vector machines contain several parameters that can be tuned to achieve optimal performance. The parameters we will tune in this analysis are regularization and gamma. The regularization parameter, which we also will refer to as cost, controls "the trade-off between maximizing the margin and minimizing the training error term" (IBM, 2021).[2] According to IBM, the value should

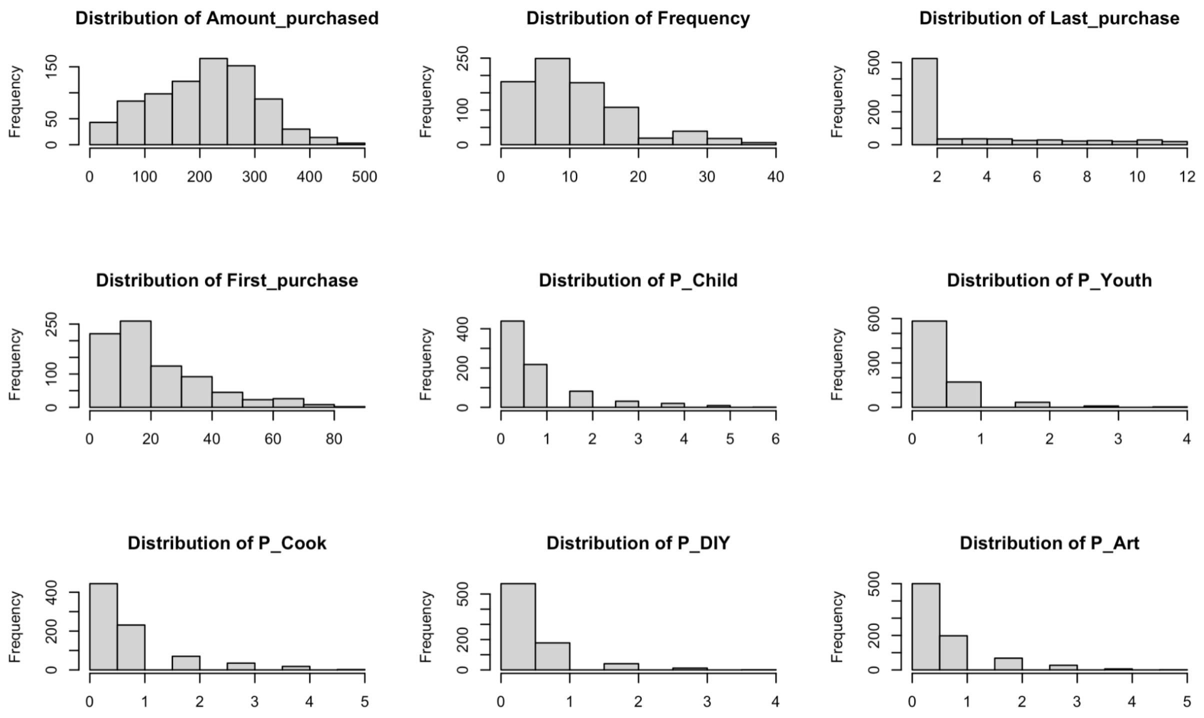
[2] https://www.ibm.com/docs/en/spss-modeler/18.2.2?topic=node-svm-expert-options. Accessed on Feb 25, 2024.

normally be between 1 and 10, so we will tune our SVM in this range and identify the optimal cost. A higher value will improve training accuracy; however, this can result in overfitting. The same is the case for the gamma parameter, which determines the extent to which a single training sample influences the hyperplane. As described in *Review of Related Literature*, the SVM outputs 0 and 1 labels rather than a probability between 0 and 1. This can be a limitation in cases where a probability is the desired output.

We will assess how each model performs on accuracy, sensitivity, and specificity. However, we will primarily compare our models on profit, while taking into account the simplicity and interpretability of the models.

Data

The dataset provided for this project contains member information for BBBC. It was provided to us already split into train and test sets. Initially, both sets had 12 variables. The train dataset contained 1600 observations and the test dataset contained 2300 observations. The datasets were already quite clean, as there were no missing values that needed to be removed. All variables (*Observation*, *Choice*, *Gender*, *Amount_purchased*, *Frequency*, *Last_purchase*, *First_purchase*, *P_Child*, *P_Youth*, *P_Cook*, *P_DIY*, *P_Art*) were coded as numeric, and we converted the response variable, *Choice*, and the *Gender* variable to factors. In reviewing the other variables, we identified *Observation* as extraneous to our modeling needs, and removed it from our train and test datasets. After initially trying models that yielded results with low sensitivity, we knew we had to balance our dataset. After balancing our data, our new train and test datasets had 800 and 408 observations respectively. The relatively small number of observations is a limitation of the dataset, as more observations provide a better likelihood of models identifying patterns and relationships in the data. However, we were still able to create models to increase profits from direct mail campaigns despite decreasing the number of observations in the train and test datasets. Next we generated histograms for each variable to evaluate whether there is skew in our data. We found that *Frequency*, *Last_purchase*, *First_purchase*, *P_Child*, *P_Youth*, *P_Cook*, *P_DIY*, and *P_Art* are all skewed to the right.



This could potentially distort the decision boundary in our logistic regression models, leading to misclassification of observations. Despite the skewed nature of the variables, we did not find that the accuracy of our logistic regression models were negatively impacted by the skewed data.

Results

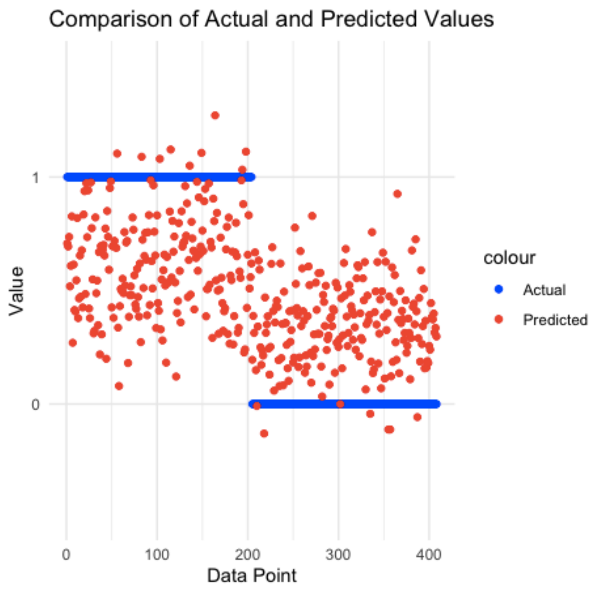
Balancing the data showed to significantly improve model sensitivity by decreasing the *proportion* of false positives and increasing the *proportion* of true positives. To highlight the magnitude of difference in performance, below are the confusion matrices, performance metrics, and profits of the logistic regression using selected predictors on unbalanced and balanced data respectively:

	log.sel.unb **(unbalanced data)**			***log.sel*** **(balanced data)**		
Confusion Matrix		Ref 0	Ref 1		Ref 0	Ref 1
	Pred 0	1582	514	Pred 0	162	42
	Pred 1	66	138	Pred 1	66	138
Accuracy	74.78%			73.53%		
Sensitivity	21.17%			76.67%		

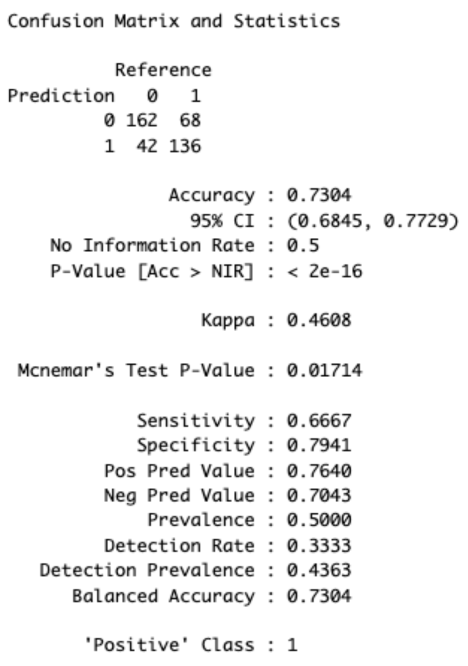
Specificity	96.00%	71.05%
Profit	**$27,717.39**	**$156,250**

Using balanced data resulted in significantly better sensitivity and greater profit across all models accessed. For this reason, we will only focus on the models using balanced data going forward.

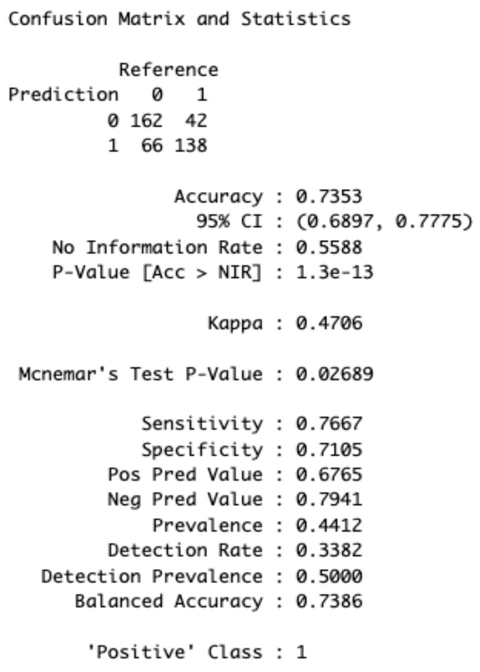
The first method we assessed was linear regression. This method is not suitable for classification tasks, which is illustrated by the scatter plot below of actual vs. predicted values. Since the task is a classification problem, we only have two outcomes: purchase (1) or no purchase (0). In the plot below, the blue dots represent the actual outcome of 0 and 1, while the predicted values represented in the red dots are not binary values or probabilities. It is clear that linear regression assumes a continuous response variable and therefore it's not ideal to use for a classification problem.



We proceeded with logistic regression, starting with all variables except *Last_purchase* after detecting multicollinearity. This left our predictors for the first logit model (*log.all*) to be: *Gender1*, *Amount_purchased*, *Frequency*, *First_purchase*, *P_Child*, *P_Youth*, *P_Cook*, *P_DIY*, *P_Art*. We also wanted to explore a logistic regression model with selected predictors (*log.sel*), so a stepwise selection using the AIC criterion was performed. According to the result, the best predictors, which will be used in *log.sel*, are: *P_Art*, *Frequency*, *Gender1*, *P_Child*, *P_DIY*, *First_purchase*, *P_Cook*, *Amount_purchased*. On the following page, confusion matrices and performance metrics are shown for both models.



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 162  68
         1  42 136

               Accuracy : 0.7304
                 95% CI : (0.6845, 0.7729)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4608

 Mcnemar's Test P-Value : 0.01714

            Sensitivity : 0.6667
            Specificity : 0.7941
         Pos Pred Value : 0.7640
         Neg Pred Value : 0.7043
             Prevalence : 0.5000
         Detection Rate : 0.3333
   Detection Prevalence : 0.4363
      Balanced Accuracy : 0.7304

       'Positive' Class : 1
```

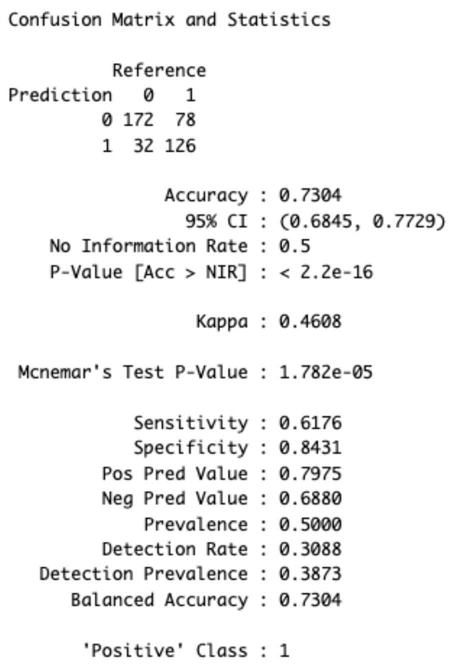


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 162  42
         1  66 138

               Accuracy : 0.7353
                 95% CI : (0.6897, 0.7775)
    No Information Rate : 0.5588
    P-Value [Acc > NIR] : 1.3e-13

                  Kappa : 0.4706

 Mcnemar's Test P-Value : 0.02689

            Sensitivity : 0.7667
            Specificity : 0.7105
         Pos Pred Value : 0.6765
         Neg Pred Value : 0.7941
             Prevalence : 0.4412
         Detection Rate : 0.3382
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.7386

       'Positive' Class : 1
```

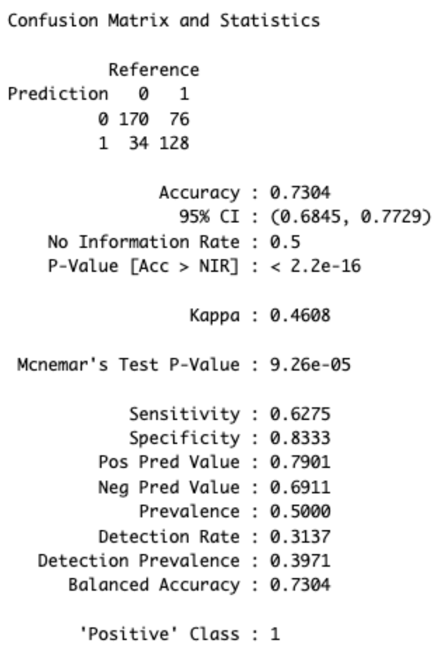
(*log.all* results vs. *log.sel* results)

According to the results, both models have similar overall accuracy. However, the first model (*log.all*) has higher specificity and the second model (*log.sel*) has higher sensitivity score, meaning the first model is better at predicting negative cases and the second model is better at predicting positive cases. Even though they share many similarities, *log.sel* outperforms *log.all* due to its balanced accuracy, sensitivity, and specificity scores.

Next, we wanted to explore SVM models and tested with 4 different kernel types: RBF, Linear, Polynomial, and Sigmoid. All variables were used as predictors in these models.

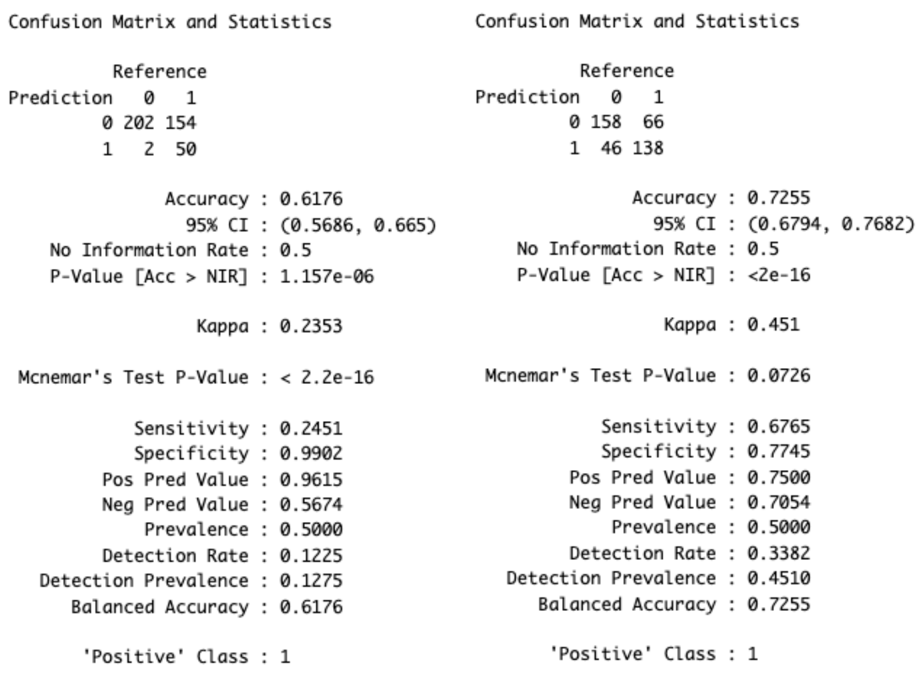


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 172  78
         1  32 126

               Accuracy : 0.7304
                 95% CI : (0.6845, 0.7729)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4608

 Mcnemar's Test P-Value : 1.782e-05

            Sensitivity : 0.6176
            Specificity : 0.8431
         Pos Pred Value : 0.7975
         Neg Pred Value : 0.6880
             Prevalence : 0.5000
         Detection Rate : 0.3088
   Detection Prevalence : 0.3873
      Balanced Accuracy : 0.7304

       'Positive' Class : 1
```



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 170  76
         1  34 128

               Accuracy : 0.7304
                 95% CI : (0.6845, 0.7729)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4608

 Mcnemar's Test P-Value : 9.26e-05

            Sensitivity : 0.6275
            Specificity : 0.8333
         Pos Pred Value : 0.7901
         Neg Pred Value : 0.6911
             Prevalence : 0.5000
         Detection Rate : 0.3137
   Detection Prevalence : 0.3971
      Balanced Accuracy : 0.7304

       'Positive' Class : 1
```

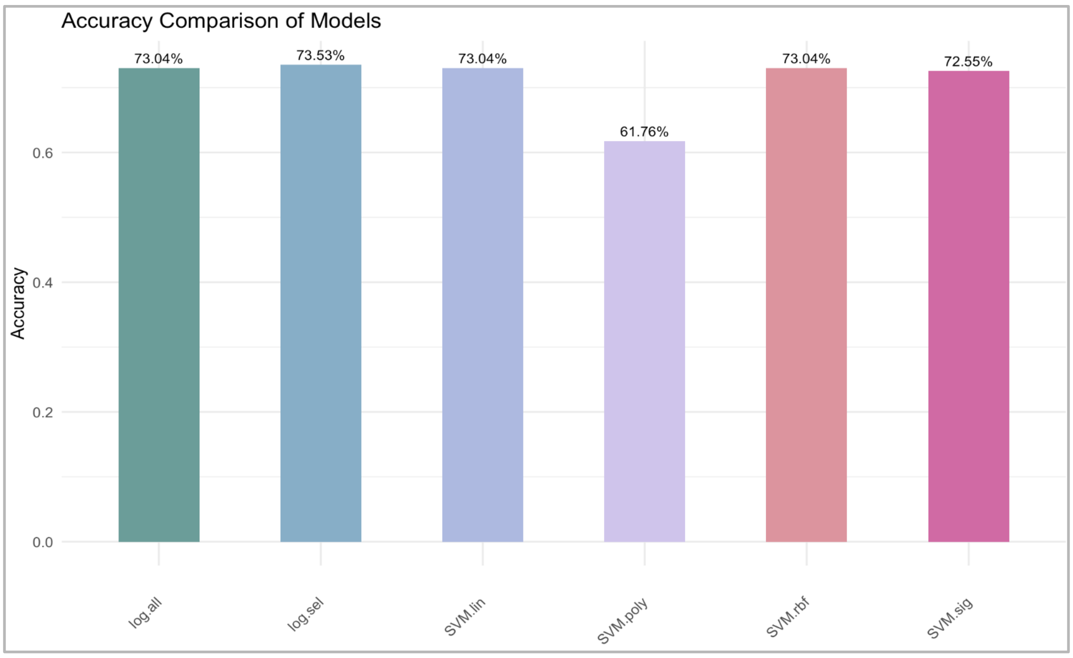
(*SVM.rbf* results vs. *SVM.lin* results)

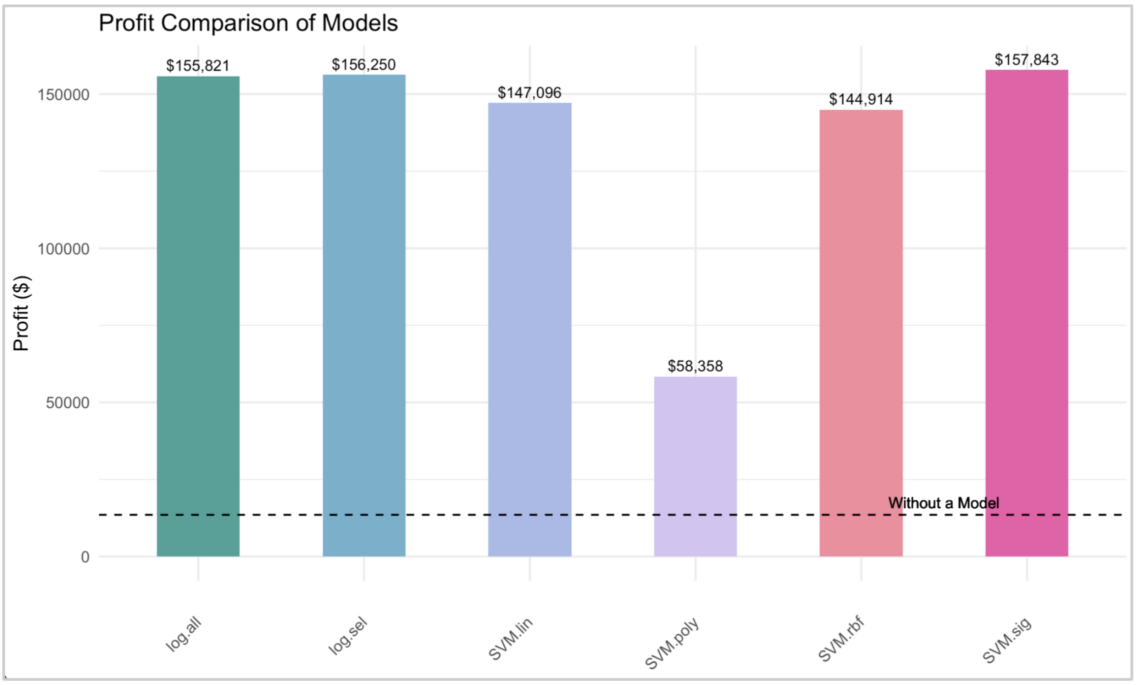


```
Confusion Matrix and Statistics              Confusion Matrix and Statistics

          Reference                                    Reference
Prediction   0   1                           Prediction   0   1
         0 202 154                                    0 158  66
         1   2  50                                    1  46 138

               Accuracy : 0.6176                            Accuracy : 0.7255
                 95% CI : (0.5686, 0.665)                     95% CI : (0.6794, 0.7682)
    No Information Rate : 0.5                    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.157e-06              P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.2353                               Kappa : 0.451

 Mcnemar's Test P-Value : < 2.2e-16           Mcnemar's Test P-Value : 0.0726

            Sensitivity : 0.2451                         Sensitivity : 0.6765
            Specificity : 0.9902                         Specificity : 0.7745
         Pos Pred Value : 0.9615                      Pos Pred Value : 0.7500
         Neg Pred Value : 0.5674                      Neg Pred Value : 0.7054
             Prevalence : 0.5000                          Prevalence : 0.5000
         Detection Rate : 0.1225                      Detection Rate : 0.3382
   Detection Prevalence : 0.1275                Detection Prevalence : 0.4510
      Balanced Accuracy : 0.6176                   Balanced Accuracy : 0.7255

       'Positive' Class : 1                         'Positive' Class : 1
```

(*SVM.poly* results vs. *SVM.sig* results)

From the results above, we can see that RBF and linear kernels show similar performance in terms of accuracy, sensitivity, and specificity. Polynomial kernel performs poorly compared to the other SVMs, with significantly lower sensitivity and comparable accuracy. Sigmoid kernel performs reasonably well with a balanced accuracy, with a slightly lower sensitivity compared to the RBF and linear kernels. Therefore, among the SVMs, RBF and linear kernels seem to be more suitable for predicting *Choice* based on the provided results.





To better compare all of our models, two bar graphs were created to compare the overall accuracy along with predicted profits as seen above. When looking at all of the models, we can see that both methods, logistic regression and SVM, created great outcomes with their models. Even the model that exhibited the worst performance (*SVM.poly*) resulted in a profit far greater than using no model. We notice from the plots that overall accuracy is not directly reflected in profits in terms of their relative rank. Using the ranking table below, we can see that even though *SVM.sig* is ranked #5 for overall accuracy, the model outperformed all other models when predicting profits. This is because profits are more affected by the proportion of true positives and false positives to total predictions, and accuracy can be a misleading metric in this context.

Rank (best to worst)	Accuracy	Profits
1	log.sel	SVM.sig
2	log.all	log.sel
3	SVM.in	log.all
4	SVM.rbf	SVM.lin
5	SVM.sig	SVM.rbf
6	SVM.poly	SVM.poly

To summarize, the best model depends on the stakeholders' preference. The logit model with selected predictors (*log.sel*) would be most ideal for ensuring accuracy, while the SVM with a sigmoid kernel model would be best for maximizing profits. However, it is important to note that the difference in accuracy (and profits) between the top models is minimal. Additionally, other factors can be considered, such as interpretability and simplicity.

Conclusion & Recommendations

In conclusion, our case study demonstrates the effectiveness of logistic regression and SVM models in predicting member purchases for the Bookbinders Book Club. By leveraging these models, BBBC can enhance their direct mail campaigns, targeting individuals more likely to purchase *The Art History of Florence*, therefore increasing overall profitability.

The three models we looked at were linear regression, logistic regression, and support vector machine models. While the SVM using a sigmoid kernel (*SVM.sig*) resulted in a marginally greater profit, the logit model with selected predictors (*log.sel*) performed almost equally well. Additionally, this model isolates the factors that most influenced Choice and allows for great interpretability of each predictors specific effect on *Choice*. These advantages outweigh the slightly greater profit by *SVM.sig*, which is why we will recommend BBBC to develop an expertise in this method to evaluate its future direct mail campaigns.

To simplify and automate the recommended methods for future modeling efforts at the company, we propose implementing streamlined processes. This includes automating data cleaning and preprocessing steps, utilizing automated feature selection techniques, and adopting automated machine learning platforms for model training and tuning. Integration with existing systems should be prioritized for seamless deployment and accessibility. Regular model updates and monitoring systems will ensure ongoing relevance and effectiveness. These measures will minimize manual effort, enhance efficiency, and maintain the accuracy of predictive models, ultimately supporting informed marketing strategies and driving profitability for the company.

Through data processing and balancing, we improved model sensitivity and accuracy, underscoring the importance of data quality in predictive modeling. Our findings emphasize the value of using balanced data sets for more reliable predictions, leading to better-informed marketing strategies and higher profits. Moving forward, BBBC should continue to refine and update their predictive models, incorporating new data and adjusting parameters as necessary. By doing so, they can stay ahead of market trends and maintain a competitive edge in the book club industry.

Appendix

Illustrating the effect of balancing data:

```
#### log.all.unb
log.all.unb = glm(Choice ~ .,data = bbbc_train, family = binomial)
summary(log.all.unb)

##
## Call:
## glm(formula = Choice ~ ., family = binomial, data = bbbc_train)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.3515281  0.2143839  -1.640   0.1011
## Gender1         -0.8632319  0.1374499  -6.280 3.38e-10 ***
## Amount_purchased  0.0018641  0.0007918   2.354   0.0186 *
## Frequency       -0.0755142  0.0165937  -4.551 5.35e-06 ***
## Last_purchase     0.6117713  0.0938127   6.521 6.97e-11 ***
## First_purchase  -0.0147792  0.0128027  -1.154   0.2483
## P_Child         -0.8112489  0.1167067  -6.951 3.62e-12 ***
## P_Youth      -0.6370422  0.1433778  -4.443 8.87e-06 ***
## P_Cook       -0.9230066  0.1194814  -7.725 1.12e-14 ***
## P_DIY        -0.9058697  0.1437025  -6.304 2.90e-10 ***
## P_Art         0.6861124  0.1270176   5.402 6.60e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1392.2  on 1589  degrees of freedom
## AIC: 1414.2
##
## Number of Fisher Scoring iterations: 5
```

```
vif(log.all.unb)
```

```
##          Gender Amount_purchased       Frequency    Last_purchase
##        1.023359         1.232172        2.490447        17.706670
##  First_purchase          P_Child         P_Youth           P_Cook
##        9.247748         2.992269        1.761546         3.229097
##           P_DIY            P_Art
##        1.992698         1.938089
```

```
# Removing Last_purchase
log.all.unb = glm(formula = Choice ~ . -Last_purchase, data = bbbc_train, family = binomial)
summary(log.all.unb)
```

```
##
## Call:
## glm(formula = Choice ~ . - Last_purchase, family = binomial,
##     data = bbbc_train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.1489829  0.2095375  -0.711 0.477079
## Gender1          -0.8302649  0.1350384  -6.148 7.83e-10 ***
## Amount_purchased  0.0022691  0.0007747   2.929 0.003399 **
## Frequency    -0.1194992  0.0152620  -7.830 4.89e-15 ***
## First_purchase          0.0306235  0.0108454   2.824 0.004748 **
## P_Child      -0.3456948  0.0908420  -3.805 0.000142 ***
## P_Youth      -0.1789417  0.1226235  -1.459 0.144489
## P_Cook       -0.4578299  0.0950443  -4.817 1.46e-06 ***
## P_DIY        -0.4265209  0.1209960  -3.525 0.000423 ***
## P_Art         1.0778036  0.1144995   9.413  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1437.0  on 1590  degrees of freedom
## AIC: 1457
##
## Number of Fisher Scoring iterations: 5
```

```
vif(log.all.unb)
```

```
##         Gender Amount_purchased        Frequency   First_purchase
##       1.021977         1.220305         2.173240         6.886806
##        P_Child          P_Youth           P_Cook            P_DIY
##       1.904631         1.320305         2.060140         1.462770
##          P_Art
##       1.603865
```

```
##### log.sel.unb
null_model_unb = glm(Choice ~ 1, data = bbbc_train, family = binomial)
full_model_unb = log.all.unb

step.model.AIC.unb = step(null_model_unb, scope = list(upper = full_model_unb),
                   direction = "both", test = "Chisq", trace = F)
summary(step.model.AIC.unb)
```

```
##
## Call:
## glm(formula = Choice ~ P_Art + Frequency + Gender + P_Cook +
##     P_DIY + Amount_purchased + P_Child + First_purchase + P_Youth,
##     family = binomial, data = bbbc_train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.1489829  0.2095375  -0.711 0.477079
## P_Art             1.0778036  0.1144995   9.413  < 2e-16 ***
## Frequency        -0.1194992  0.0152620  -7.830 4.89e-15 ***
## Gender1          -0.8302649  0.1350384  -6.148 7.83e-10 ***
## P_Cook           -0.4578299  0.0950443  -4.817 1.46e-06 ***
## P_DIY            -0.4265209  0.1209960  -3.525 0.000423 ***
## Amount_purchased  0.0022691  0.0007747   2.929 0.003399 **
## P_Child          -0.3456948  0.0908420  -3.805 0.000142 ***
## First_purchase    0.0306235  0.0108454   2.824 0.004748 **
## P_Youth          -0.1789417  0.1226235  -1.459 0.144489
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1437.0  on 1590  degrees of freedom
## AIC: 1457
##
## Number of Fisher Scoring iterations: 5
```

```
# Best model based on stepwise
log.sel.unb <- glm(Choice ~ P_Art + Frequency + Gender + P_DIY + P_Cook + P_Child +
                   First_purchase + Amount_purchased + P_Youth, bbbc_train, family = binomial)
summary(log.sel)
```

```
##
## Call:
## glm(formula = Choice ~ P_Art + Frequency + Gender + P_DIY + P_Cook +
##     P_Child + First_purchase + Amount_purchased, family = binomial,
##     data = train_bal)
##
## Coefficients:
```

```
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.8797500  0.2560728   3.436 0.000591 ***
## P_Art          1.2069720  0.1545460   7.810 5.73e-15 ***
## Frequency     -0.1229230  0.0183054  -6.715 1.88e-11 ***
## Gender1       -0.7331796  0.1690540  -4.337 1.44e-05 ***
## P_DIY         -0.5341190  0.1588913  -3.362 0.000775 ***
## P_Cook        -0.3366476  0.1168964  -2.880 0.003978 **
## P_Child       -0.4351938  0.1116050  -3.899 9.64e-05 ***
## First_purchase         0.0357414  0.0128205   2.788 0.005306 **
## Amount_purchased  0.0015329  0.0009822   1.561 0.118596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##         Null deviance: 1109.04  on 799  degrees of freedom
## Residual deviance:  872.64  on 791  degrees of freedom
## AIC: 890.64
##
## Number of Fisher Scoring iterations: 5
```

```
# Predict the responses on the testing data
bbbc_test$PredProb = predict.glm(log.sel, newdata = bbbc_test, type = "response")
bbbc_test$PredY = ifelse(bbbc_test$PredProb >= 0.5, 1,0)

caret::confusionMatrix(as.factor(bbbc_test$Choice), as.factor(bbbc_test$PredY), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 903 297
##          1 123 277
##
##                   Accuracy : 0.7375
##                     95% CI : (0.7152, 0.7589)
##        No Information Rate : 0.6412
##        P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                    Kappa : 0.3886
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.4826
##         Specificity : 0.8801
##         Pos Pred Value : 0.6925
##         Neg Pred Value : 0.7525
##                    Prevalence : 0.3588
##         Detection Rate : 0.1731
##         Detection Prevalence : 0.2500
##         Balanced Accuracy : 0.6813
##
##         'Positive' Class : 1
##
```

Selection of model predictors for the first logistic regression model:

```
################ Logistic regression ################

log.all = glm(Choice ~ .,data = train_bal, family = binomial)
summary(log.all)
```

```
##
## Call:
## glm(formula = Choice ~ ., family = binomial, data = train_bal)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)     0.719827   0.269615   2.670 0.007589 **
## Gender1        -0.744311   0.171978  -4.328 1.51e-05 ***
## Amount_purchased  0.001123   0.001003   1.120 0.262674
## Frequency      -0.081057   0.021828  -3.713 0.000205 ***
## Last_purchase          0.614180   0.131459   4.672 2.98e-06 ***
## First_purchase   -0.007123   0.017481  -0.407 0.683665
## P_Child       -0.889430   0.152134  -5.846 5.02e-09 ***
## P_Youth       -0.624450   0.183136  -3.410 0.000650 ***
## P_Cook        -0.836064   0.159087  -5.255 1.48e-07 ***
## P_DIY         -0.979040   0.189289  -5.172 2.31e-07 ***
## P_Art          0.755409   0.180001   4.197 2.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1109.04  on 799  degrees of freedom
## Residual deviance:  846.61  on 789  degrees of freedom
## AIC: 868.61
##
## Number of Fisher Scoring iterations: 5
```

```
vif(log.all)
```

```
##          Gender Amount_purchased        Frequency    Last_purchase
##        1.025875         1.193503         3.195547        19.757532
##  First_purchase          P_Child          P_Youth           P_Cook
##       10.389191         3.503547         1.845452         3.347260
##           P_DIY            P_Art
##        2.039381         1.907348
```

```
# Removing Last_purchase
log.all = glm(formula = Choice ~ . -Last_purchase, data = train_bal, family = binomial)
summary(log.all)
```

```
##
## Call:
## glm(formula = Choice ~ . - Last_purchase, family = binomial,
##     data = train_bal)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.9208144  0.2601682   3.539 0.000401 ***
## Gender1       -0.7290414  0.1691635  -4.310 1.63e-05 ***
## Amount_purchased  0.0015558  0.0009837   1.582 0.113747
## Frequency    -0.1287053  0.0193036  -6.667 2.60e-11 ***
## First_purchase          0.0409473  0.0139291   2.940 0.003285 **
## P_Child      -0.4552718  0.1135639  -4.009 6.10e-05 ***
## P_Youth      -0.1441172  0.1483594  -0.971 0.331346
## P_Cook       -0.3602930  0.1197804  -3.008 0.002630 **
## P_DIY        -0.5439201  0.1595118  -3.410 0.000650 ***
## P_Art         1.1811966  0.1565024   7.547 4.44e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1109.04  on 799  degrees of freedom
## Residual deviance:  871.69  on 790  degrees of freedom
## AIC: 891.69
##
## Number of Fisher Scoring iterations: 5
```

```
vif(log.all)
```

```
##           Gender Amount_purchased        Frequency   First_purchase
##         1.026975         1.196322         2.576274         6.599038
##          P_Child          P_Youth           P_Cook            P_DIY
##         2.059394         1.240451         1.917399         1.503938
##            P_Art
##         1.454631
```

Stepwise selection predictors:

```
###### Stepwise Selection with AIC
null_model = glm(Choice ~ 1, data = train_bal, family = binomial)
full_model = log.all

step.model.AIC = step(null_model, scope = list(upper = full_model),
                      direction = "both", test = "Chisq", trace = F)
summary(step.model.AIC)
```

```
##
## Call:
## glm(formula = Choice ~ P_Art + Frequency + Gender + P_Child +
##     P_DIY + First_purchase + P_Cook + Amount_purchased, family = binomial,
##     data = train_bal)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.8797500  0.2560728   3.436 0.000591 ***
## P_Art            1.2069720  0.1545460   7.810 5.73e-15 ***
## Frequency       -0.1229230  0.0183054  -6.715 1.88e-11 ***
## Gender1         -0.7331796  0.1690540  -4.337 1.44e-05 ***
## P_Child         -0.4351938  0.1116050  -3.899 9.64e-05 ***
## P_DIY           -0.5341190  0.1588913  -3.362 0.000775 ***
## First_purchase   0.0357414  0.0128205   2.788 0.005306 **
## P_Cook          -0.3366476  0.1168964  -2.880 0.003978 **
## Amount_purchased  0.0015329  0.0009822   1.561 0.118596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1109.04  on 799  degrees of freedom
## Residual deviance:  872.64  on 791  degrees of freedom
## AIC: 890.64
##
## Number of Fisher Scoring iterations: 5
```

```
# Best model based on stepwise
log.sel <- glm(Choice ~ P_Art + Frequency + Gender + P_DIY + P_Cook + P_Child +
     First_purchase + Amount_purchased, train_bal, family = binomial)
summary(log.sel)
```

```
##
## Call:
## glm(formula = Choice ~ P_Art + Frequency + Gender + P_DIY + P_Cook +
##     P_Child + First_purchase + Amount_purchased, family = binomial,
##     data = train_bal)
```

```
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.8797500  0.2560728   3.436 0.000591 ***
## P_Art            1.2069720  0.1545460   7.810 5.73e-15 ***
## Frequency       -0.1229230  0.0183054  -6.715 1.88e-11 ***
## Gender1         -0.7331796  0.1690540  -4.337 1.44e-05 ***
## P_DIY           -0.5341190  0.1588913  -3.362 0.000775 ***
## P_Cook          -0.3366476  0.1168964  -2.880 0.003978 **
## P_Child         -0.4351938  0.1116050  -3.899 9.64e-05 ***
## First_purchase     0.0357414  0.0128205   2.788 0.005306 **
## Amount_purchased  0.0015329  0.0009822   1.561 0.118596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1109.04  on 799  degrees of freedom
## Residual deviance:  872.64  on 791  degrees of freedom
## AIC: 890.64
##
## Number of Fisher Scoring iterations: 5
```

Calculating profit using all models:

```
##### Sending to all:
cost_all = 50000 * 0.65
# 50,000 * Unit mailing cost

revenue_all = 0.0903 * 50000 * (31.95-15-6.75)
# Revenue: 50,000 * 9.03% * (Selling Price - Costs)

profit_all = (revenue_all - cost_all)
profit_all # 13,553

## [1] 13553
```

```
##### Using log.all to target customers:
cost_log.all = (42+136)/408 * 50000 * 0.65
#          (FP+TP)/408 * 50,000 * Unit mailing cost
#                      ^
#          Proportion of 1's in model

revenue_log.all = 136/408 * 50000 * (31.95-15-6.75)
# Number of TP/Total test observations * 50,000 * (Selling Price - Book costs)

profit_log.all = (revenue_log.all - cost_log.all)
profit_log.all # $155,821.1

## [1] 155821.1

##### Using log.sel to target customers:
cost_log.sel = (66+138)/408 * 50000 * 0.65
#          (FP+TP)/408 * 50,000 * Unit mailing cost
#                      ^
#          Proportion of 1's in model

revenue_log.sel = 138/408 * 50000 * (31.95-15-6.75)
# Number of TP/Total test observations * 50,000 * (Selling Price - Costs)

profit_log.sel = (revenue_log.sel - cost_log.sel)
profit_log.sel # $156,250

## [1] 156250

##### Using svm.rbf to target customers:
cost_svm.rbf = (32+126)/408 * 50000 * 0.65
#          (FP+TP)/408 * 50,000 * Unit mailing cost
#                      ^
#          Proportion of 1's in model

revenue_svm.rbf = 126/408 * 50000 * (31.95-15-6.75)
# Number of TP/Total test observations * 50,000 * (Selling Price - Costs)
```

```
profit_svm.rbf = (revenue_svm.rbf - cost_svm.rbf)
profit_svm.rbf # $144,914.2

## [1] 144914.2

##### Using svm.lin to target customers:
cost_svm.lin = (34+128)/408 * 50000 * 0.65
#         (FP+TP)/408 * 50,000 * Unit mailing cost
#                    ^
#         Proportion of 1's in model

revenue_svm.lin = 128/408 * 50000 * (31.95-15-6.75)
# Number of TP/Total test observations * 50,000 * (Selling Price - Costs)

profit_svm.lin = (revenue_svm.lin - cost_svm.lin)
profit_svm.lin # $147,095.6

## [1] 147095.6

##### Using svm.poly to target customers:
cost_svm.poly = (2+50)/408 * 50000 * 0.65
#         (FP+TP)/408 * 50,000 * Unit mailing cost
#                    ^
#         Proportion of 1's in model

revenue_svm.poly = 50/408 * 50000 * (31.95-15-6.75)
# Number of TP/Total test observations * 50,000 * (Selling Price - Costs)

profit_svm.poly = (revenue_svm.poly - cost_svm.poly)
profit_svm.poly # $58,357.84

## [1] 58357.84

##### Using svm.sig to target customers:
cost_svm.sig = (46+138)/408 * 50000 * 0.65
#         (FP+TP)/408 * 50,000 * Unit mailing cost
#                    ^
#         Proportion of 1's in model
```

```
revenue_svm.sig = 138/408 * 50000 * (31.95-15-6.75)
# Number of TP/Total test observations * 50,000 * (Selling Price - Costs)


profit_svm.sig = (revenue_svm.sig - cost_svm.sig)
profit_svm.sig # $157,843.1

## [1] 157843.1
```