EBA5002: Practice Module for Business Analytics Practice

Repeat Bias Prediction in the Leather Crafts Industry

Final Report
24 October 2021

Team Name
Talesmith

Team Members
Deng Jing
Ho Bao Yuan, Seth
Orson Teng

Table of Contents

1. Introduction

1.1 Project Sponsor - Gnome & Bow

Founded in 2013, Gnome & Bow (GNB) is a retailer of premium and personalized leather goods, ranging from briefcases, handbags, wallets and more. Their products are sold both online and in their brick-and-mortar store, as well as through stockists such as Zalora and KrisShop (SIA's sales arm). Their webpage can be reached at https://gnomenbow.com.

Picture from: Gnome & Bow webpage

1.2 Business Problem

Being in a niche sector is not without its problems. Owing to the premium nature of their branding and products, the company has limited options when it comes to generating revenue. They cannot lower their price willy-nilly, or give out overly-generous discounts, because the relatively high price point is a part of their brand value, which they must maintain through upholding price benchmarks. Aside from marketing campaigns to raise brand awareness, there is another way to generate revenue - repeat customers.

In almost every industry, having repeat customers can contribute significantly to the company's margins - GNB not being an exception. Repeat customers make multiple purchases over time, can be swayed to new product offerings, and cost less to retain relative to attracting new customers. Although the same person is not likely to buy two of the same item, GNB has a product range that is relatively wide, where they can encourage customers to buy products of different types. This is a good source of additional revenue, builds brand loyalty and has the potential to be indirect marketing, when customers flaunt their purchases online or when they go about their daily routine.

1.3 Project Objective

Our group's objective is to help GNB build a predictive model that predicts whether a first-time customer is likely to become a repeat customer based on their demographics and the characteristics of their first purchase. This would allow GNB to conduct further business actions like targeted marketing emails or subsequent purchase discounts.

In the report, we will refer to customers who has made multiple orders with GNB as 'repurchasers' and customers who only has made a single lifetime order with GNB as 'non-repurchasers'.

2. Data Cleansing & Preparation

2.1 Raw Data

After agreeing on the objective and scope of the project, we received three files which contained the initial data we would be using. Due to the large number of immaterial columns, we will not be going over every field here, and only items which were materially of interest will be discussed.

i) Membership data

This represents the list of members who signed up as members on the GNB webpage. The key fields are as follows:

Field Name	**Data Type**	**Comments**
Email Address	String	Unique identifier for membership
First and Last Name	String	Name of Customer
Gender	Character	Only 43% populated, with non-standardized values for gender
Birthday	Date	Only 43% populated
Country	String	Only 35% populated
Address	String	Only 3% populated
Member Rating	Integer	Ordinal ranking from 1-5 indicating email engagement level, 5 is highest engagement

ii) Product Catalogue

This is a list of products of both past and present products available for purchase, along with their product group and retail price. The key fields are as follows:

Field Name	**Data Type**	**Comments**
Product Name	String	Unique name of the product e.g. "Balsa Travel Duffel"
Product Group	String	3 groups: Bags, Wallets or Accessories
Product Type	String	14 product types e.g Backpack, Bifold wallet or Luggage Tag
Price	Numeric	Retail price of item
Description	String	Text description of the item on the web page

iii) Order List

This is our largest dataset, containing 7 years of transactions data. Initially having 6,600 rows, each row represents each item purchased, where a single order can span multiple rows if more than 1 item was purchased. Orders for Personalization (e.g. name engraving services), Refunds and Courier payments sometimes appear as unique rows as well. The key fields are as follows:

Field Name	**Data Type**	**Comments**
Name	String	Proxy for Order number, e.g. "#1234"
Email	String	Email address if available. Only 50% populated
Financial Status	String	Indicates whether an item was Paid or Refunded
Total	Numeric	Total numeric value of the order
Discount	Numeric	Discount in dollars applied to that order
Lineitem name	String	Product name

2.2 Identified Problems and Missing Data

An initial glance at the datasets looked promising, because there were a large number of fields with promising names. However, upon further scrutiny, we noticed that quite a large number of fields were not fully populated. More than half of the cells in both datasets were blank. Missing data was mostly found under the Membership list, where critical data about the customers' profile was missing. For example, Gender and Birthdate have more than 65% missing data, which made the raw data unusable and we are unable to make assumptions about the population in its current form.

In the Order List, the biggest issue was that order data was separated by row, meaning that a single order of multiple items would have multiple rows in the Order List. A customer who buys a Wallet, Backpack, and gets his or her name engraved on either item would occupy at least 3 rows on the order list; one for the wallet, one for the backpack, and one for the engraving service. If the item needed courier delivery or was refunded, more lines would be used. This was problematic because we needed to analyze orders as a whole order, and not as individual components of each order.

2.3 Data Cleansing and Preparation of Order List

The raw data alone had 75 columns, and because of this we opted to use Excel and VBA to clean the data, since the data could be visualized better in spreadsheet format, and the appropriate actions done well with a small amount of VBA code. In sequence, the actions taken to clean the data were:

i) Added OriginalRowId, attaching to each row its original numeric row number in the original data set, so that we could trace back processed data back to the raw data whenever required.

ii) Deleted Refunded and Voided rows, which were rows where the order was not fulfilled, but was instead refunded or voided in the system. These were not legitimate orders, hence were excluded.

iii) Deleted orders that were not associated with an Email, because these orders could not be identified to a unique customer nor can the company send targeted emails to them if identified by the model.

iv) Copied down data for single orders with multiple items that pertain to the entire order, where only the top row is populated, but bottom rows were only partially populated, although the data would have been the same as the top row of that order.

v) Trimming the text of the product name, removing colour and "limited edition" characters of the string, so as to standardise the product names.

vi) Deleted rows that corresponded to sale of E-gift cards, purchase of sample pieces, couriers services, service of repairs among others, to further sieve out items that were not legitimate orders compared to typical orders.

vii) Combined multiple rows of same order ID into a single order (VBA script in Appendix 1):

a) Wrote a loop that scanned if any of the subsequent rows had the same Order Number as the original row. If the subsequent row had the same Order Number as the first, meaning that they were part of the same order, script would combine the orders. The number of items would increase by 1, from a starting value of 1.

b) The category of Product (Bag, Wallet etc) of each row would be one-hot encoded. The first row would always encode itself, and if the second row is of the same Order Number, the product would be encoded into the products of the first row. The second row would then be cleared of its contents

2.4 Joining of Data Sets

After the bulk of the data cleaning, we proceeded to join the data sets. Our objective was to have a large transactional Order List, which contained details of each order such as the product details and customer details. These were to be taken from the other datasets.

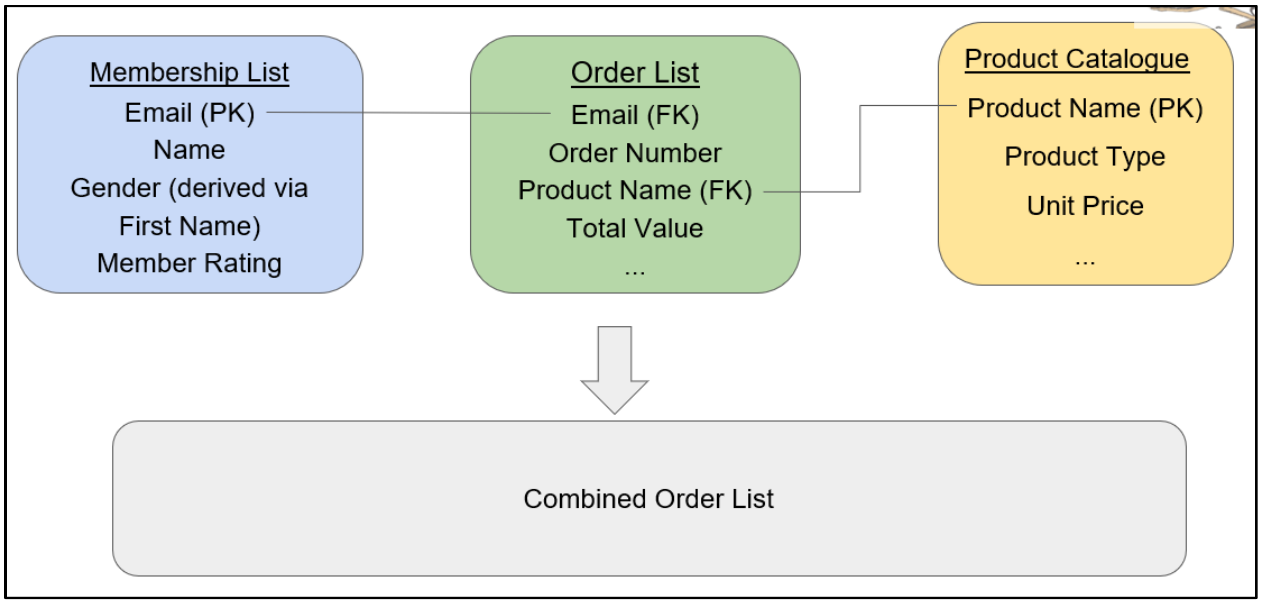


Fig 2.4 - Joining of data sets into a Combined Order List

Between the Order List and Membership list, the customers' unique Email was used as the joining key. Between the Order List and Product Catalogue, the Product Name was used as the joining key.

2.5 Derived Variables

We were able to derive additional variables from the data we had on hand, together with further data processing. This will also help to reduce the total number of variables by combining the information across multiple columns into one.

2.5.1 RepeatStatus

RepeatStatus refers to whether the order resulted in a subsequent purchase. In order to derive this, every order first starts with a value of 0. Starting from the oldest order to the newest, if an email detected a repeat of itself, and the repeat has a different order number, the older row would have its value changed from 0 to 1, and the newer row would have a value of 2, signifying that the first order was the first of many, and the second order was the second order for that particular email. Further repeats would be labelled 3,4,5 and so on. Hence, orders with values staying as 0 were those that did not result in a subsequent purchase.

Orders with RepeatStatus values of 2 and above were deleted, meaning we deleted the second and subsequent purchases of the same email account. This is because we only want the purchase details of first purchases (VBA script in Appendix 2).

2.5.2 Gender

We used R's Gender package to impute missing genders in our dataset. (Appendix 3) The steps were as follows:

1. Picking out first names
2. Replacing missing names with email names
3. Gender package used to assign gender

Due to the limitations in the Gender package (using SSA which might be outdated, some names are unable to generate a gender), a few fields are still missing, we then randomly assign the remaining missing genders.

2.5.3 Percentage Discount

This variable was derived from total and total discount, where:

$$Percentage\ Discount\ =\frac{total\ discount}{total\ +\ total\ discount}$$

2.5.4 Member Rating

Member rating refers to the rating assigned by Mailchimp to each individual member based on the degree of email engagement. Mailchimp is the email marketing platform which the sponsor uses. This can range from a member rating of 1 to 5. However, as our order data includes non-members as well since we consider purchases from non-members, we allocated an additional rating of 0 for non-members. This would allow us to also remove the member column, which indicates whether the order is made by a member, to be holistically represented with member rating instead.

2.5.5 Installment

We created a variable called Installment, which indicated whether a customer purchase was on full-cash terms or if it was purchased on an installment plan through an installment service provider such as Hoolah or Atome-Pay. If payment method contained “Hoolah” or “Atome”, Installment variable would be allocated a value of 1, otherwise 0.

2.5.6 Personalization Done

If Personalization was done, meaning orders where the product name was listed as “Engraving”, “Personalization” or similar words, the order would have its PersonalizationDone variable set to 1, from a default of 0. The Personalization row would then be cleared of its contents

2.5.7 Festive Purchase

Based on information provided by the sponsor, their sales usually peaks around the festive seasons of Christmas and Valentine’s, due to the gifting aspect commonly associated. As the team believes that purchases made during this period are more likely to have a different intent, this new variable was created. If the purchase was made between mid-November to mid-February, the value would be set to 1, otherwise 0.

2.5.8 Variables from Text Analytics Processing

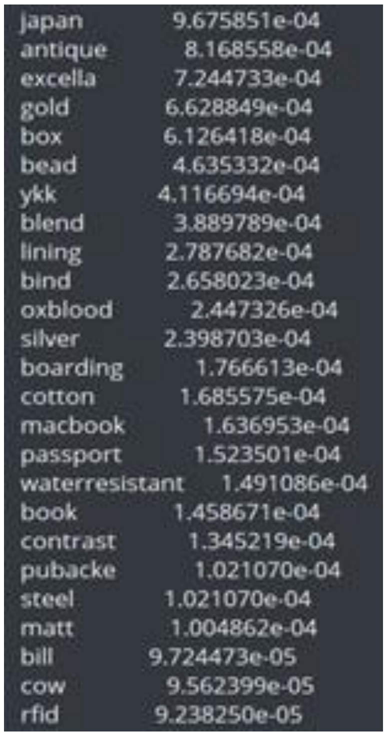
To further augment the dataset used for training and testing the models, the team used text analytics processing techniques to convert product description (available from the product catalogue dataset) into additional input variables. This was implemented with the packages Spacy and Scikit-learn via Python, together with Excel.

Firstly, an initial tokenizer was used to do the following to product description data:

1) Remove next line (\n) characters
2) Lowercase all characters
3) Remove punctuations
4) Lemmatize words into their root form for standardization
5) Remove words if they are part of Spacy’s default stopword list

As a result, 218 distinct words were left which had some issues that required further intervention with an additional customized stopword list. Words such as ‘1315’ and ‘b7’ that were uninterpretable (due to web-scraping), and words that were universal to almost all major products like ‘leather’ were added to this list. Words with repeated meanings such as ‘rfid’ and ‘rfidprotection’ also had duplicates added to this list.

The final tokenizer (Appendix 4) used this list to remove the identified stopwords, resulting in 115 distinct words. The processed text data and a list of the distinct words were then used by Excel to convert into 115 input variables, each variable indicating whether the product(s) purchased had a description which contained the words.



japan	9.675851e-04
antique	8.168558e-04
excella	7.244733e-04
gold	6.628849e-04
box	6.126418e-04
bead	4.635332e-04
ykk	4.116694e-04
blend	3.889789e-04
lining	2.787682e-04
bind	2.658023e-04
oxblood	2.447326e-04
silver	2.398703e-04
boarding	1.766613e-04
cotton	1.685575e-04
macbook	1.636953e-04
passport	1.523501e-04
waterresistant	1.491086e-04
book	1.458671e-04
contrast	1.345219e-04
pubacke	1.021070e-04
steel	1.021070e-04
matt	1.004862e-04
bill	9.724473e-05
cow	9.562399e-05
rfid	9.238250e-05

However, the team felt that 115 input variables could be further reduced. Using Rattle implemented via R, the 115 input variables were put through a Random Forest to calculate and rank their importance towards the target variable (RepeatStatus). The top 22 words with the highest importance values were selected with the cut-off value of 10^-5, as seen in the picture above. These were added to the dataset used for the building of models.

3. Final dataset

3.1 Data Dictionary

The final dataframe which we used to pass into the models consists of 46 training variables and 1 target variable “RepeatStatus”. The data dictionary in the table below includes a brief description on each of the variables.

Field	**Description**	**Values**
RepeatStatus	Whether this particular order has a future purchase from the same customer	0: no 1: yes, return customer
Gender	Gender of customer	F: female M: male
MEMBER_RATING	Email engagement level from mailchimp	0, 1, 2, 3, 4, 5 (lowest engagement to highest)
Accepts Marketing	Whether customer accepts marketing	Yes, no
Discount_Percentage	Percentage of discount for the order	number
Total	Total amount paid for the purchase	number
Installment	Paid by installment or in full	Cash, Installment
Source	Online or in store	Web: online Pos: store
Personalisation	If personalisation is involved	0: no 1: yes
NumberItemsTotal	Total number of items purchase	number
Product types x14	Whether the order includes this product type	0: no 1: yes
Description text x22	Whether the order includes products with this text in its description	0: no 1: yes
Festive_Purchase	Whether this order was made during festive season	0: no 1: yes

3.2 Data Profiling

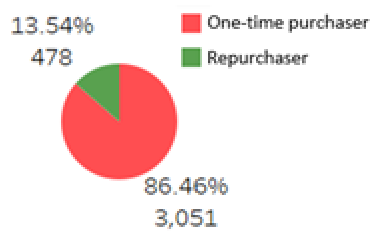


Fig 3.2 - Pie chart of target variable

After completing numerous data processing steps to process the dataset into useful data for model building, here are the results of profiling it. With a total of 3529 records, the majority of orders were done by non-repurchasers at 86.46%, which can be seen in the pie chart.

A large proportion of the input variables were categorical in nature, with 22 categorical variables derived from the text analytics processing and 21 variables from the other data processing steps. Only 3 input variables are numerical in nature.

3.2.1 Categorical Variables

Using stacked bar charts to compare the differences in proportions relative to the target variable, we can have a quick overview of potentially useful categorical variables. The variables highlighted here are those with more than 2% difference in proportion. Remaining variables can be found in Appendix 5.

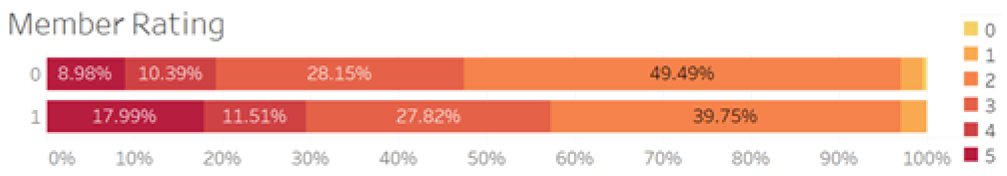


Fig 3.2.1 - Stacked bar chart for MemberRating

For member rating, one can see that there is generally a larger proportion of higher member ratings amongst orders related to repurchasers, compared to non-repurchasers. This difference is especially significant for those at the highest member rating value of 5.

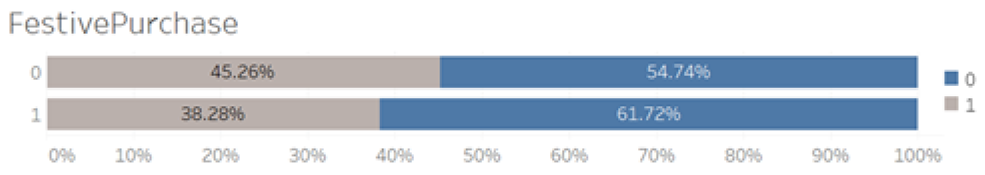


Fig 3.2.2 - Stacked bar chart for FestivePurchase

A larger proportion of purchases done (45.26%) during a festive period were by non-repurchasers, compared to those by repurchasers (38.28%). One possible perspective is that purchases meant for gifting to others (assuming that the product was positively received) resulted in a smaller product experience/weaker positive feedback compared to non-gifting purchases. Buyers spending on themselves could view the purchases more positively due to being the ones that forked out the money.

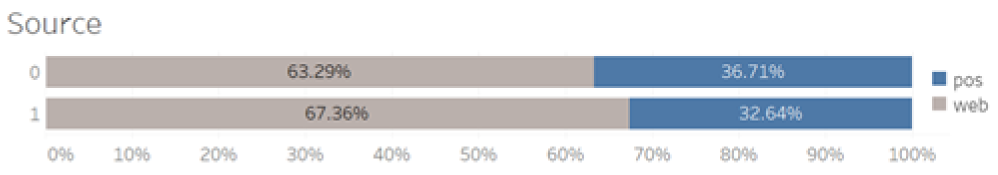


Fig 3.2.3 - Stacked bar chart for Source

Web purchases could also be a decent indicator to identify repurchasers, with 67.36% of purchases done by repurchasers coming from the web, compared to 63.29% that were made by non-repurchasers from the web.

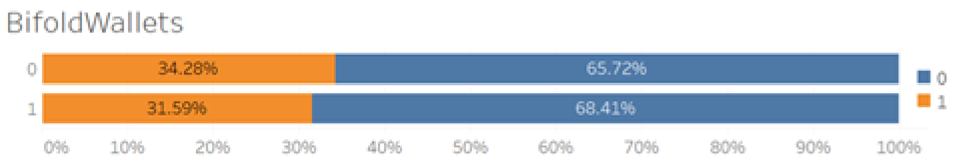


Fig 3.2.4 - Stacked bar chart for BifoldWallets

Interestingly, the only variable derived from the product category purchased that had more than 2% difference (between being made by repurchasers and non-repurchasers) was whether a Bifold Wallet was purchased. Note that this product category also contributed to the largest proportion of purchases in general, the rest being less than 20%. Based on the stacked bar chart, a Bifold Wallet being made in a first purchase could be an indicator to non-repurchasers.

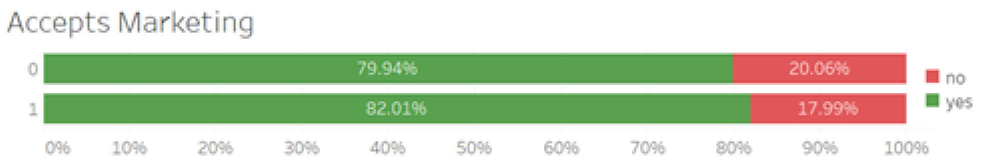


Fig 3.2.5 - Stacked bar chart for AcceptsMarketing

Lastly, if the purchaser has accepted marketing, they could be more likely to come back and make a subsequent purchase. A larger proportion of repurchasers accepted marketing (82.01%) than non-repurchasers (79.94%)

The remaining categorical variables will not be covered due the smaller differences in proportion relative to the ones above, and they can be found in the annex. Additionally, as the categorical variables arising from text analytics processing has been discussed earlier with regards to their variable importance, they will also not be covered.

3.2.2 Numerical Variables

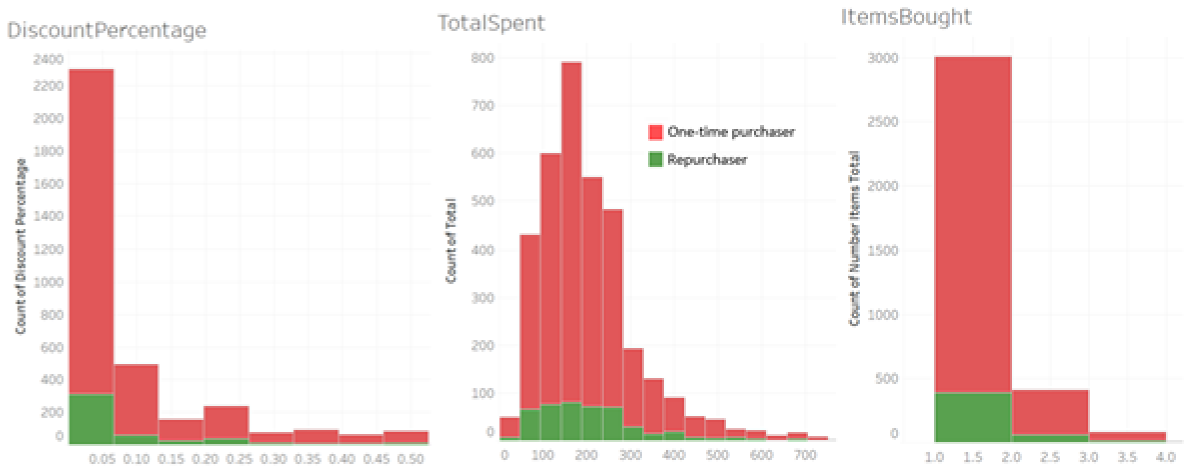


Fig 3.3 - Stacked histograms of numerical variables

The 3 stacked histograms above are plotted based on discount percentage, total spent and number of items bought. The majority of orders are made with less than 25% discount, between $50 to $300 spent and less than 3 items purchased. While no obvious insights can be derived visually when comparing the repurchasers against the one-time purchasers from the histograms, we hope that they can help the models differentiate between the repeat purchases.

4. Performance Metrics

We looked at the various performance metrics and evaluated their relevance to the business needs.

Accuracy	This metric is relevant for judging the models holistically.
Recall	This metric would be used as the main performance metric to optimise on to ensure the model aims to return a high rate of true positives. This translates to correctly identifying and sending emails to repeat customers.
Precision	This value would be low if there are high false positives. However, considering that wrongly identifying non-repurchasers might be of lower cost compared to missing out on the actual repurchasers, this metric will not be a priority.
Specificity	True negatives, will not be a priority.

Note that while accuracy is a relevant metric to reduce the odds of spamming non-repurchasers with emails, the potential revenue from reaching out to more repurchasers make recall a higher priority.

5. Modelling Process

5.1 Initial Models

The team decided to use 3 models that allowed us to set thresholds based on probability to alter the classification results, which are namely Random Forest, XGBClassifier and Logistic Regression. Initial recall of 0.6 was used as a minimum benchmark for tuning of threshold and a training-test split of 80-20 was standardized across the different models.

5.1.1 Random Forest

Random forest is a classification algorithm which consists of many decision trees. This makes it suitable for our problem, where we have many training variables, and we aim to classify the entries into 1 (repurchasers) and 0 (non-repurchasers). This algorithm was implemented using Python's Scikit-learn package (Appendix 6).

We used Pandas library's get dummies function to one-hot encode categorical variables. Next, RandomForestClassifier from Scikit-learn was used. A recall of 0.62 was achieved with a threshold of 0.09. The confusion matrix and the other performance metrics based on this threshold can be seen below in Figure 5.1.1.

	Predict : 0	Predict : 1	Recall	Accuracy	Precision	Specificity
Actual : 0	238	383	0.62	0.41	0.12	0.38
Actual : 1	32	58				

Fig 5.1.1 - Model results of Random Forest

5.1.2 XGBClassifier

Another model that the team attempted was the XGBClassifier, which was implemented using the XGBoost and Scikit-learn packages via Python. Due to the nature of categorical variables, they were first transformed using one-hot encoding techniques. Next, the data was split 80/20, and then trained with the help of GridSearchCV function to iteratively find the best combination of model parameters based on recall.

Using the optimum parameters (Appendix 7), the model's predicted probabilities were taken and converted into predicted target values based on a threshold value. In order to reach the team's benchmark of 60% recall, a threshold value of 0.07 was used. The confusion matrix and the varying scores can be seen below in figure 5.1.2.

	Predict : 0	Predict : 1	Recall	Accuracy	Precision	Specificity
Actual : 0	253	357	0.60	0.44	0.14	0.41
Actual : 1	38	58				

Fig 5.1.2 - Model results of XGBClassifier

5.1.3 Logistics Regression

We next ran the same dataframe in R, building a logistics regression model. We encoded most of the variables as factors, except the Discount and Order Total amounts, which we left as numeric. After splitting our Train set and Test set, we ran a logistics regression model, using the GLM formula in R with Binomial family structure, and a backwards stepwise direction, with RepeatStatus as the target, and all other variables (except Order ID, our order identifier) as potential predictors.

Initially, the result was bad, and the model predicted the entire Test set to be positive. This was with the default threshold of 0.5, the probability cut-off where values above this threshold would lead to that prediction being classified as a Positive. Slowly lowering the threshold, we found that most observations had a prediction score within the range of 0.07 to 0.15 , so we adjusted our threshold to this approximate range. As Recall was one of our higher priorities, we first aimed for a recall of 0.6, where we found that a threshold of 0.09 achieved this. The confusion matrix and the varying scores can be seen below in figure 5.1.3.

	Predict : 0	Predict : 1	Recall	Accuracy	Precision	Specificity
Actual : 0	305	305	0.60	0.51	0.16	0.50
Actual : 1	38	58				

Fig 5.1.3 - Model results of Logistics Regression

5.1.4 Comparison between models

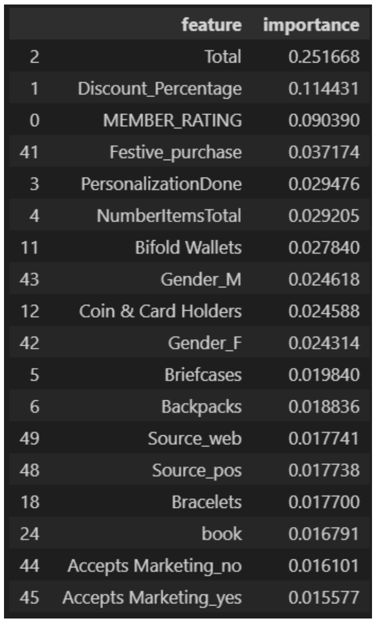
An initial benchmark of 0.6 recall was used when tuning the three models above. The recall rate was achieved at various threshold levels as shown in the diagram below in figure 5.1.4..

	Random Forest	XGBoost	Logistic Regression
Threshold	0.09	0.07	0.11
Recall	**0.62**	**0.60**	**0.60**
Accuracy	0.41	0.44	0.51
Precision	0.12	0.14	0.16
Specificity	0.38	0.41	0.50

Fig 5.1.4 - Side-by-side comparison of model results

5.1.5 Interesting Findings

Using RandomForestClassifier's feature importance function (Appendix 8), an initial dive into looking at the feature importance was surprising, only 1 word appeared in the top most important words as seen in the picture below.



	feature	importance
2	Total	0.251668
1	Discount_Percentage	0.114431
0	MEMBER_RATING	0.090390
41	Festive_purchase	0.037174
3	PersonalizationDone	0.029476
4	NumberItemsTotal	0.029205
11	Bifold Wallets	0.027840
43	Gender_M	0.024618
12	Coin & Card Holders	0.024588
42	Gender_F	0.024314
5	Briefcases	0.019840
6	Backpacks	0.018836
49	Source_web	0.017741
48	Source_pos	0.017738
18	Bracelets	0.017700
24	book	0.016791
44	Accepts Marketing_no	0.016101
45	Accepts Marketing_yes	0.015577

This result was interesting, which prompted us to test our models again without the variables generated from text analytics processing. Considering how the model eventually has to be implemented for the business to use, reducing variables would greatly reduce the complexity of the model as well.

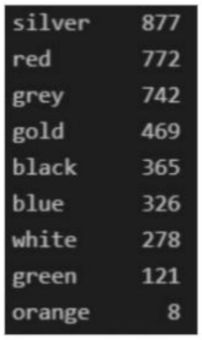
	Random Forest		**XGBoost**		**Logistic Regression**	
Threshold	0.09		0.07		0.11	
Recall	0.62	**0.64**	0.60	**0.69**	0.60	**0.70**
Accuracy	0.44	**0.46**	0.44	**0.43**	0.51	**0.41**
Precision	0.13	0.13	0.14	**0.15**	0.16	**0.15**
Specificity	0.42	**0.44**	0.41	**0.39**	0.50	**0.37**

Fig 5.1.5 - Model results before (left) and after (right) removing text variables

Upon removal, all three models performed better without the text variables in terms of recall. Based on Figure 5.1.5, the changes in the various performance metrics upon removal of text variables can be seen. This helped the team to conclude that the variables resulting from text analytics processing did not help with identifying repurchasers, and hence were subsequently removed.

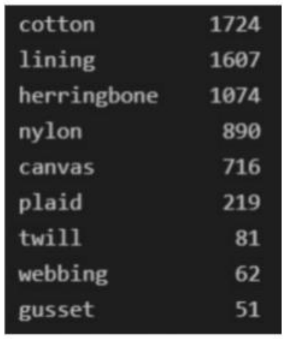
Despite removing the text variables from our model, we still feel that there are additional insights which we can derive from these data. Therefore, we explored wordnet (Appendix 9) to identify different categories which may value-add to the business. For a start, we explored "colors", "materials" and "decoration". Since the text tokens were already one-hot encoded, their popularity can be ranked after a summation of the times they appear. This translates into how popular a particular feature is.

Colors



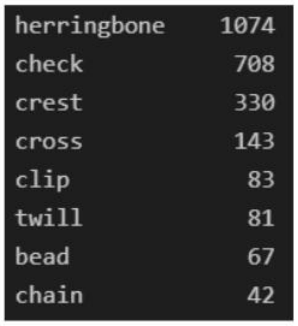
```
silver    877
red       772
grey      742
gold      469
black     365
blue      326
white     278
green     121
orange      8
```

Materials



```
cotton         1724
lining         1607
herringbone    1074
nylon           890
canvas          716
plaid           219
twill            81
webbing          62
gusset           51
```

Decoration



```
herringbone    1074
check           708
crest           330
cross           143
clip             83
twill            81
bead             67
chain            42
```

It might be interesting to note that words like cotton appeared a lot because it is the inner-lining material of most of the bags. This may not be so useful. Business has to make use of these data based on their own discretion, considering their understanding of the products.

5.2 Final Model Selection

5.2.1 New metric - Email reduction

Only using recall is insufficient for comparison between the models, because all models can achieve higher recall by simply lowering the threshold. This would introduce more false positives as we result. However, the models would be pointless if recall is 100% and if this is from the models labelling everyone as repurchasers. Therefore, we looked into the business objective once again and decided to derive an additional performance metric - email reduction, which tells us how much marketing efforts we can save and the degree to which the emails are targeted. The calculation is as follows:

$$Email\ reduction = \frac{True\ Negatives\ +\ False\ Negatives}{Total\ orders}$$

This indicator would be used in conjunction with recall to compare the models, we want both metrics to be as high as possible.

5.2.2 Comparison between models

A graph of email reduction against recall values (generated by varying the threshold levels) is plotted for all the three models. Note that this was done after removing the variables generated from text analytics processing. Using this graph, the model that is closest to the top right will be the best model, because it means that it can achieve higher recall with similar levels of email reduction.

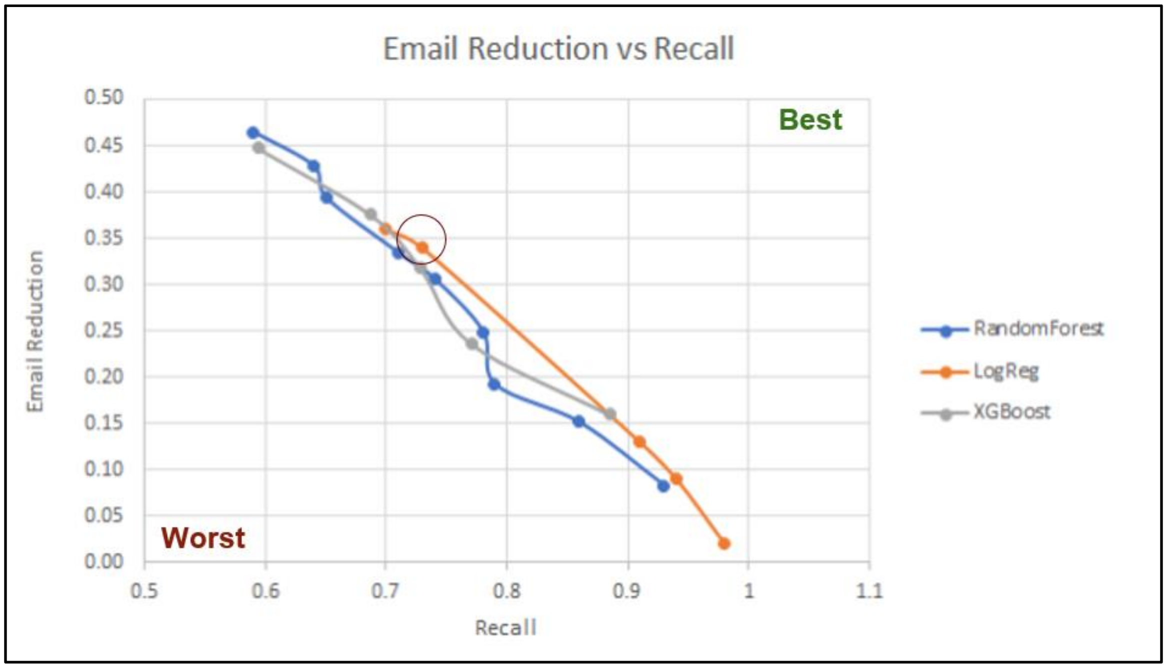
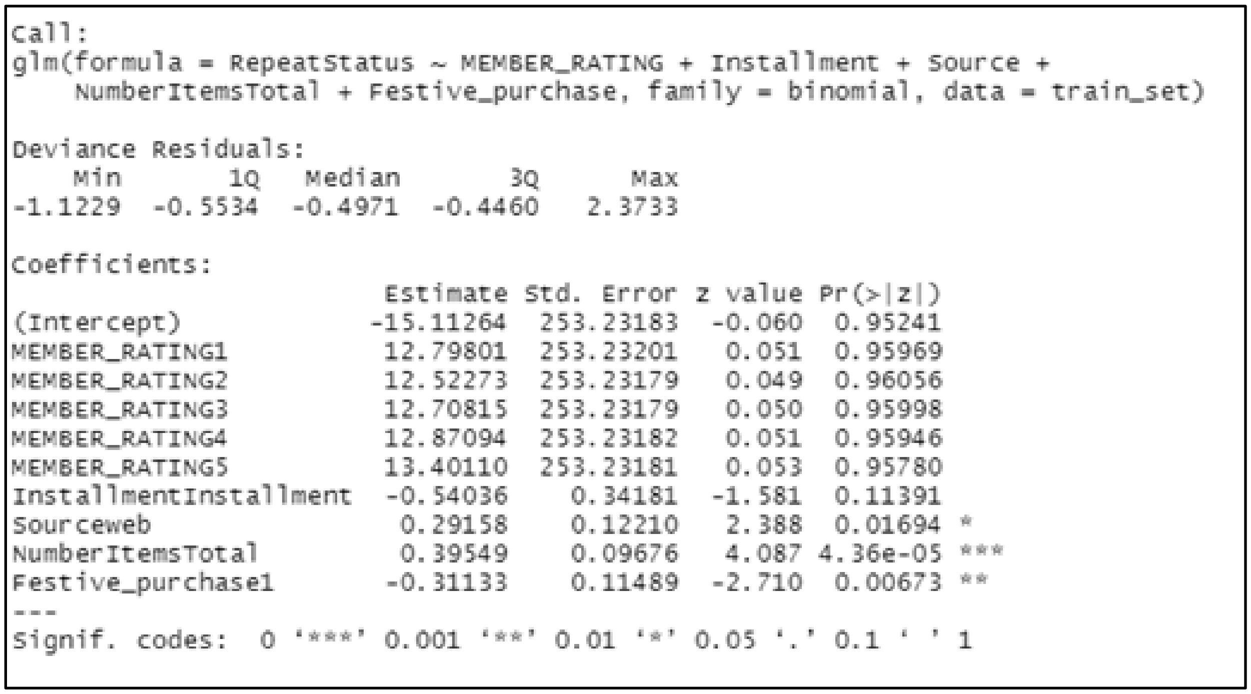


Fig 5.2.2 - Plot of Email Reduction versus Recall, for all 3 models

We can see that logistic regression slightly outperforms the other two models. Hence, it was selected as the final model. The second point in the logistic regression represents a slight kink in the graph, which we identify as an optimal point with a relatively high recall of >0.7 and an email reduction of close to 0.35.

6. Final Model

6.1 Evaluation



```
Call:
glm(formula = RepeatStatus ~ MEMBER_RATING + Installment + Source +
    NumberItemsTotal + Festive_purchase, family = binomial, data = train_set)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1229  -0.5534  -0.4971  -0.4460   2.3733

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            -15.11264  253.23183  -0.060  0.95241
MEMBER_RATING1          12.79801  253.23201   0.051  0.95969
MEMBER_RATING2          12.52273  253.23179   0.049  0.96056
MEMBER_RATING3          12.70815  253.23179   0.050  0.95998
MEMBER_RATING4          12.87094  253.23182   0.051  0.95946
MEMBER_RATING5          13.40110  253.23181   0.053  0.95780
InstallmentInstallment  -0.54036    0.34181  -1.581  0.11391
Sourceweb                0.29158    0.12210   2.388  0.01694 *
NumberItemsTotal         0.39549    0.09676   4.087 4.36e-05 ***
Festive_purchase1       -0.31133    0.11489  -2.710  0.00673 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig 6.1.1 - Logistics Regression Output of final model, using R

Based on the results of the logistic regression model as seen above, only 5 variables were selected. It can be seen that in general, orders done by those with member ratings of 1-5 increase the probability of being a repurchaser significantly over 0 (base class). Amongst these values, those with a rating of 4 and 5 contribute the most to repurchase behaviour, as opposed to 1-3. Probability of identifying the customer as a repurchaser also increases if an order is fully paid instead of via installments, purchased over the web, included a larger number of items and is done outside of the festive period.

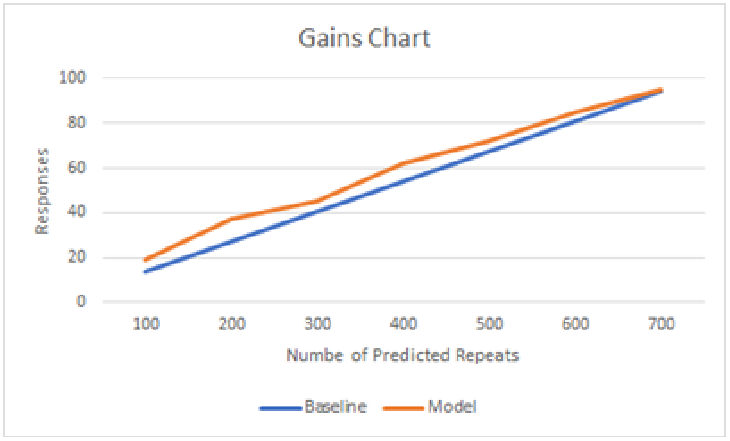


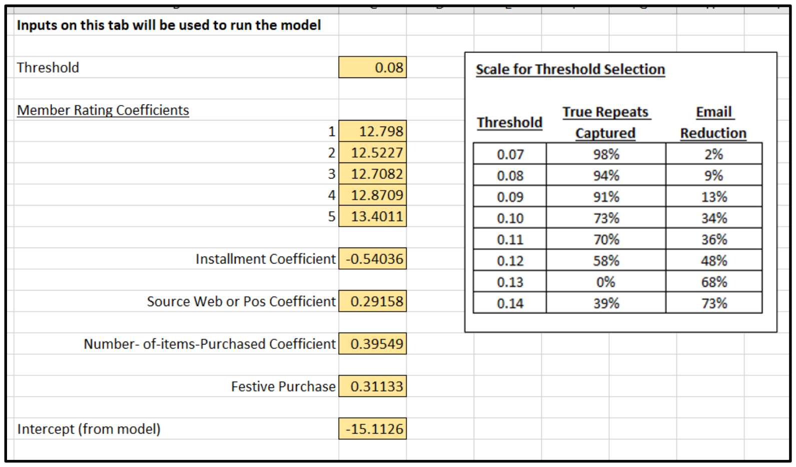
Fig 6.1.2 - Gains chart of Final Model

However, the group admits that the final model was not as good as we had hoped. From the gains chart above, comparing against the baseline (with 13.5% chance of repurchase), the gains of the selected model were not significantly larger. On the other hand, as the main form of email marketing was assumed to be low cost in nature, the team feels that higher false positive rates are still a good trade-off for a higher recall given the gains in reaching out to a customer which would result in repurchase behavior.

6.2 Implementation

Thinking of ways we apply this model for the business, we developed an excel based calculator for short term implementation. This Excel Calculator applies our logistics regression model to data that a user manually enters. The Calculator has a model inputs page, where a user could adjust the variables used in the calculation. There is also an Order input page, for a user to input order details, using which the model would predict the RepeatStatus based on the input, with a click of the macro-linked button.

Model Input page



Inputs on this tab will be used to run the model

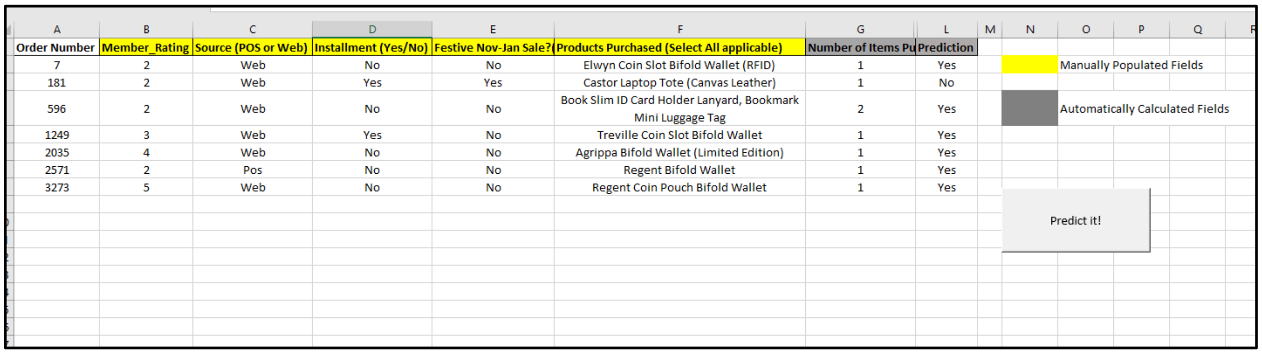
Threshold	0.08
Member Rating Coefficients	
1	12.798
2	12.5227
3	12.7082
4	12.8709
5	13.4011
Installment Coefficient	-0.54036
Source Web or Pos Coefficient	0.29158
Number- of-items-Purchased Coefficient	0.39549
Festive Purchase	0.31133
Intercept (from model)	-15.1126

Scale for Threshold Selection

Threshold	True Repeats Captured	Email Reduction
0.07	98%	2%
0.08	94%	9%
0.09	91%	13%
0.10	73%	34%
0.11	70%	36%
0.12	58%	48%
0.13	0%	68%
0.14	39%	73%

Fig 6.2.1 - Excel Calculator's Model Input Tab, for users to change coefficients

Order Input Page



Order Number	Member_Rating	Source (POS or Web)	Installment (Yes/No)	Festive Nov-Jan Sale?(	Products Purchased (Select All applicable)	Number of Items Pu	Prediction
7	2	Web	No	No	Elwyn Coin Slot Bifold Wallet (RFID)	1	Yes
181	2	Web	Yes	Yes	Castor Laptop Tote (Canvas Leather)	1	No
596	2	Web	No	No	Book Slim ID Card Holder Lanyard, Bookmark Mini Luggage Tag	2	Yes
1249	3	Web	Yes	No	Treville Coin Slot Bifold Wallet	1	Yes
2035	4	Web	No	No	Agrippa Bifold Wallet (Limited Edition)	1	Yes
2571	2	Pos	No	No	Regent Bifold Wallet	1	Yes
3273	5	Web	No	No	Regent Coin Pouch Bifold Wallet	1	Yes

🟨	Manually Populated Fields
▨	Automatically Calculated Fields

Fig 6.2.2 - Excel Calculator's Order Input tab, for a user to manually input order details and obtain a prediction

7. Closing Thoughts and Conclusion

The current data set could be overrepresented by non-repurchasers, because there are likely to be some recent purchasers, who have not yet made a repurchase, but might in future. Our model is also calibrated based on historical purchases, together with the historical product offerings, some of which are no longer available. It is difficult to predict human behaviours, especially impulse buying, and effects of concurrent marketing tactics. We found it hard to fully capture behaviour with the data we had available.

Covid19 has impacted all our lives greatly, and has definitely affected spending patterns and behaviour of consumers over the last two years. It is likely that this spending pattern and behaviour will change Post Covid-19.

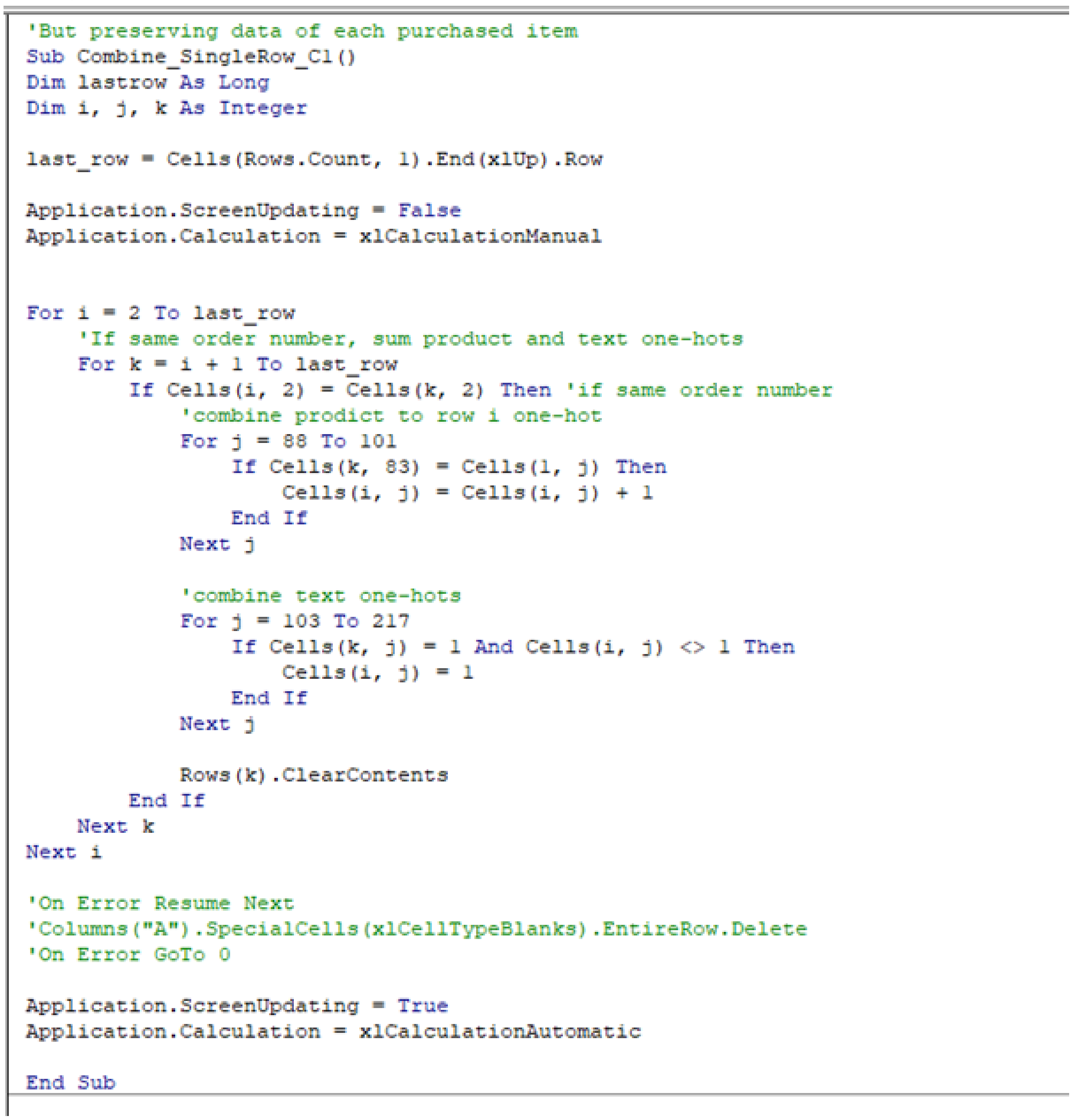
However, we hope that this initial step towards data analytics-driven decision making (using the excel calculator) can display some short-term benefits for GNB, and that they will see value in the insights which we provide. In the future, when more new products are added to the lineup and old ones are phased out, GNB can consider enhancing the current model by investing in data analytics software, and/or hire a data analyst who could bring their analytics game forward. For data collection in the future, GNB can consider compulsory inputs of Birthday and Gender when signing up for membership and this will help to derive more potentially useful fields such as age.

With better data collection and investment into analytics, we believe that there is potential for GNB to better utilise the data that they already have.

(Finalized scripts and dataset are available for download in https://drive.google.com/drive/folders/1qrecsiwkJT05PkD_B33Unry3JFiwoPCW?usp=sharing)

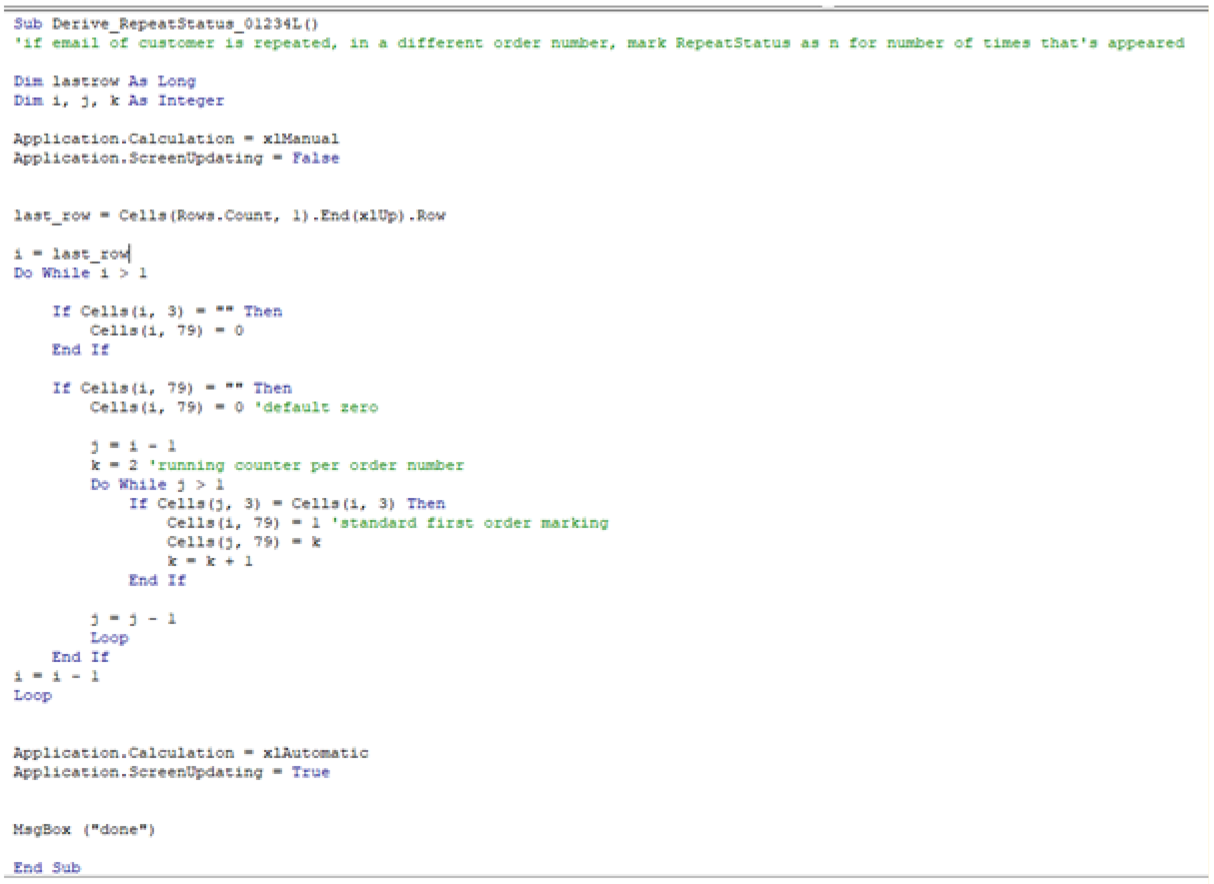
8. Appendix

Appendix 1: Excel VBA code for Combining multiple orders into a single row



```
'But preserving data of each purchased item
Sub Combine_SingleRow_C1()
Dim lastrow As Long
Dim i, j, k As Integer

last_row = Cells(Rows.Count, 1).End(xlUp).Row

Application.ScreenUpdating = False
Application.Calculation = xlCalculationManual


For i = 2 To last_row
    'If same order number, sum product and text one-hots
    For k = i + 1 To last_row
        If Cells(i, 2) = Cells(k, 2) Then 'if same order number
            'combine prodict to row i one-hot
            For j = 88 To 101
                If Cells(k, 83) = Cells(1, j) Then
                    Cells(i, j) = Cells(i, j) + 1
                End If
            Next j

            'combine text one-hots
            For j = 103 To 217
                If Cells(k, j) = 1 And Cells(i, j) <> 1 Then
                    Cells(i, j) = 1
                End If
            Next j

            Rows(k).ClearContents
        End If
    Next k
Next i

'On Error Resume Next
'Columns("A").SpecialCells(xlCellTypeBlanks).EntireRow.Delete
'On Error GoTo 0

Application.ScreenUpdating = True
Application.Calculation = xlCalculationAutomatic

End Sub
```

Appendix 2: Excel VBA Code for deriving RepeatStatus

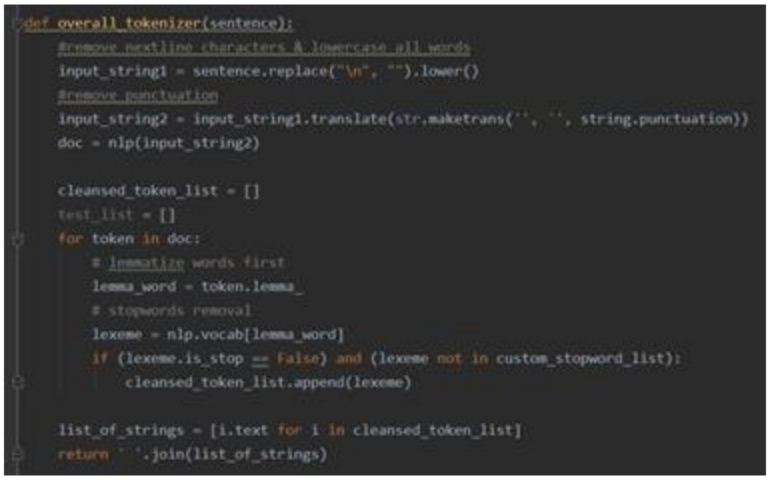


```
Sub Derive_RepeatStatus_01234L()
'if email of customer is repeated, in a different order number, mark RepeatStatus as n for number of times that's appeared

Dim lastrow As Long
Dim i, j, k As Integer

Application.Calculation = xlManual
Application.ScreenUpdating = False


last_row = Cells(Rows.Count, 1).End(xlUp).Row

i = last_row
Do While i > 1

    If Cells(i, 3) = "" Then
        Cells(i, 79) = 0
    End If

    If Cells(i, 79) = "" Then
        Cells(i, 79) = 0 'default zero

        j = i - 1
        k = 2 'running counter per order number
        Do While j > 1
            If Cells(j, 3) = Cells(i, 3) Then
                Cells(i, 79) = 1 'standard first order marking
                Cells(j, 79) = k
                k = k + 1
            End If

        j = j - 1
        Loop
    End If
i = i - 1
Loop


Application.Calculation = xlAutomatic
Application.ScreenUpdating = True


MsgBox ("done")

End Sub
```

Appendix 3: Impute Gender via R

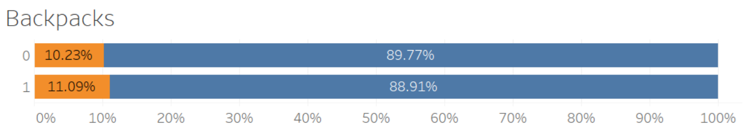
```
#remove rows where both email and names are missing
df <- df[!(nchar(df$Email)<2 & nchar(df$Billing.Name)<2), ]

#take first name
df$Billing.Name <- word(df$Billing.Name, 1)

#remove numbers, replace firstname with email name if firstname is less than 2 characters / empty
df$Billing.Name <- gsub('[0-9]+', '', df$Billing.Name)
df$Billing.Name <- ifelse(nchar(df$Billing.Name) < 1, gsub("@.*$", "", df$Email),df$Billing.Name)

df$Gender <- ""

for ( i in 1:length(df$Billing.Name)) {
  try(df$Gender[i] <- gender(df$Billing.Name[i],method="ssa")$gender)
}

#imputation: for gender = undefined, take random value
df$Gender[df$Gender==""]<-sample(c("male", "female"))
```

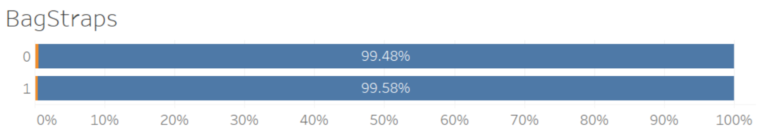
Appendix 4: Text Analytics Processing - Final Tokenizer

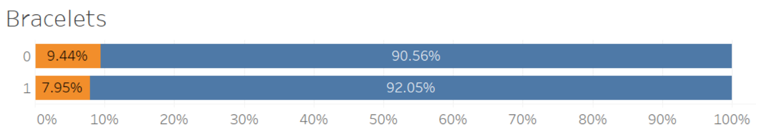


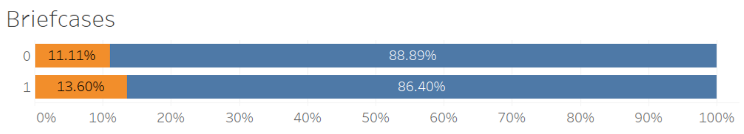
```python
def overall_tokenizer(sentence):
    #remove nextline characters & lowercase all words
    input_string1 = sentence.replace("\n", "").lower()
    #remove punctuation
    input_string2 = input_string1.translate(str.maketrans('', '', string.punctuation))
    doc = nlp(input_string2)

    cleansed_token_list = []
    test_list = []
    for token in doc:
        # lemmatize words first
        lemma_word = token.lemma_
        # stopwords removal
        lexeme = nlp.vocab[lemma_word]
        if (lexeme.is_stop == False) and (lexeme not in custom_stopword_list):
            cleansed_token_list.append(lexeme)

    list_of_strings = [i.text for i in cleansed_token_list]
    return ' '.join(list_of_strings)
```

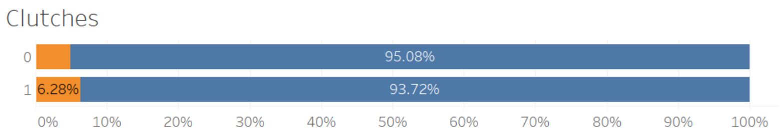
Appendix 5: Data profiling - Additional categorical variables

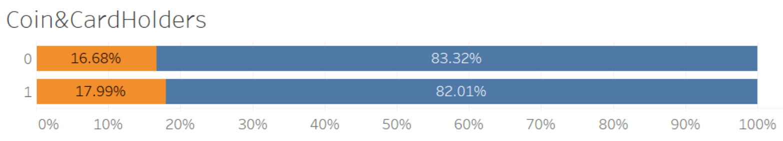


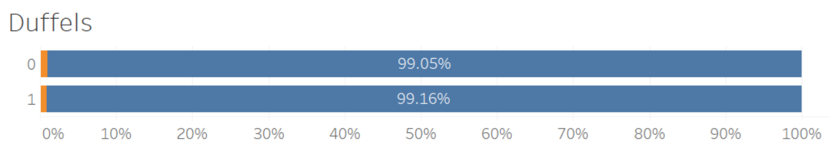


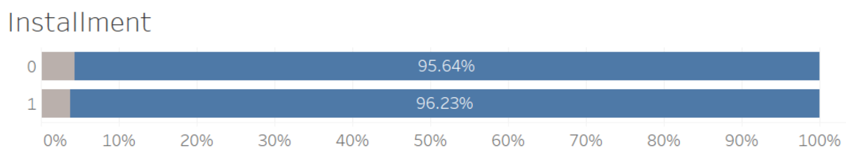


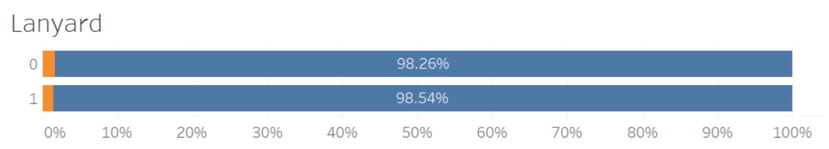


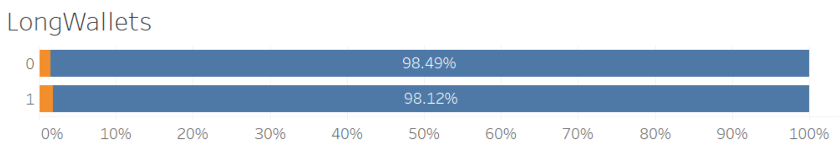


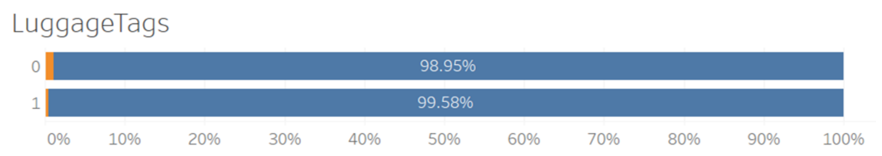


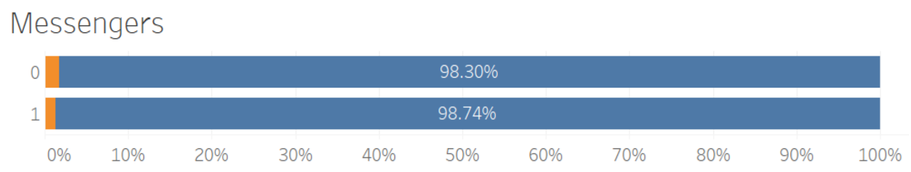

Duffels
0
99.05%
1
99.16%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

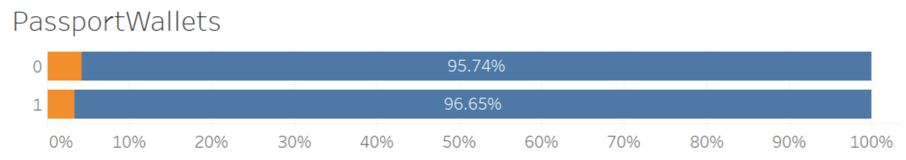

Installment
0
95.64%
1
96.23%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

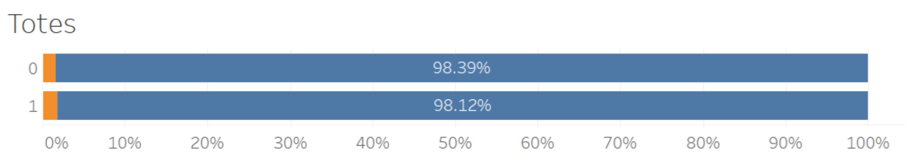

Lanyard
0
98.26%
1
98.54%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

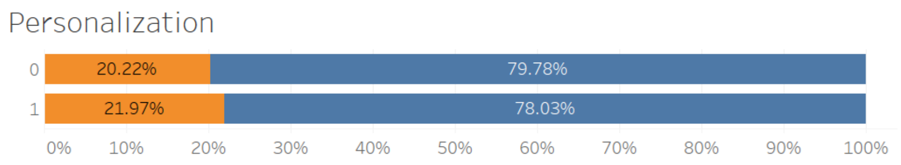

LongWallets
0
98.49%
1
98.12%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%


LuggageTags
0
98.95%
1
99.58%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

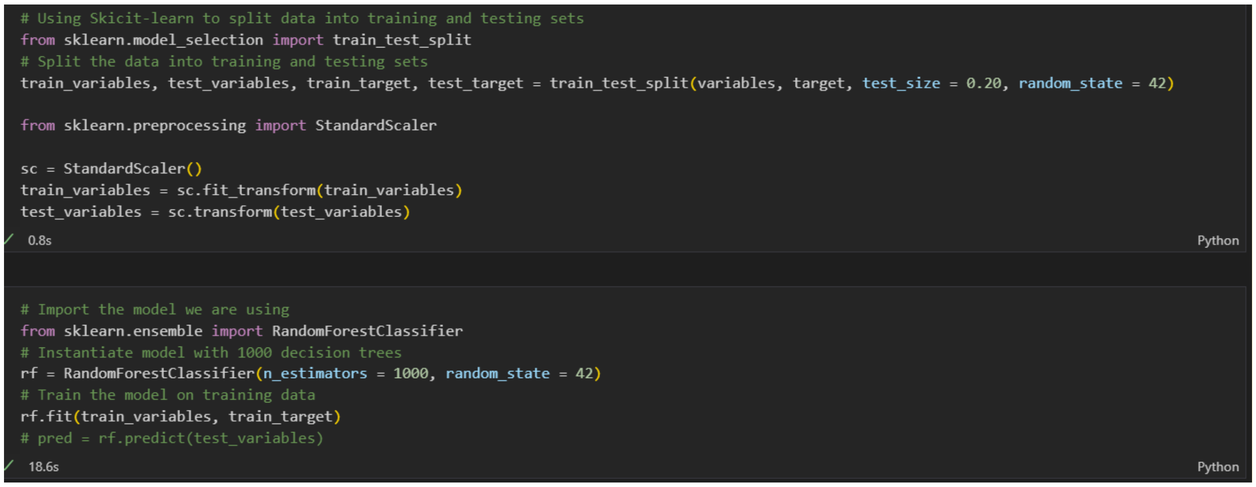

Messengers
0
98.30%
1
98.74%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%


PassportWallets
0
95.74%
1
96.65%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%


Totes
0
98.39%
1
98.12%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%


Personalization
0
20.22%
79.78%
1
21.97%
78.03%
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

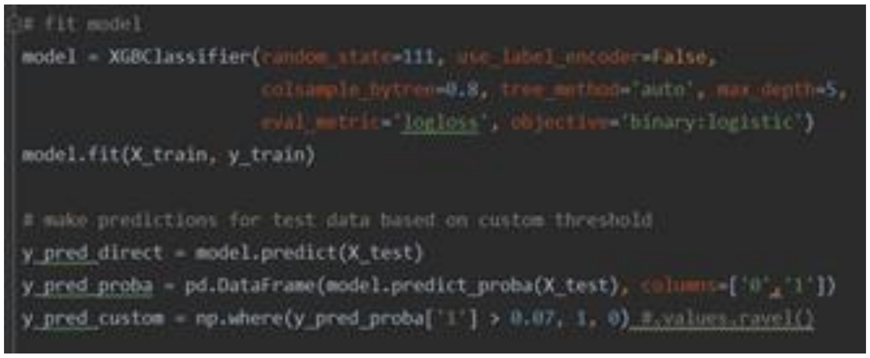
Appendix 6: Random Forest script in Python



```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
train_variables, test_variables, train_target, test_target = train_test_split(variables, target, test_size = 0.20, random_state = 42)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
train_variables = sc.fit_transform(train_variables)
test_variables = sc.transform(test_variables)
```
✓ 0.8s Python

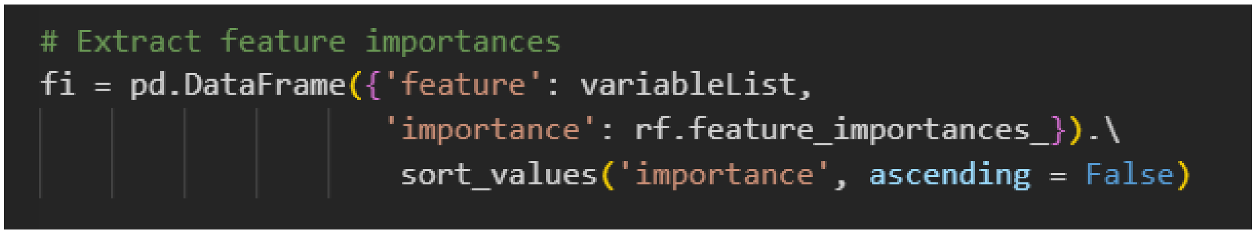
```
# Import the model we are using
from sklearn.ensemble import RandomForestClassifier
# Instantiate model with 1000 decision trees
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42)
# Train the model on training data
rf.fit(train_variables, train_target)
# pred = rf.predict(test_variables)
```
✓ 18.6s Python

Appendix 7: XGBClassifier



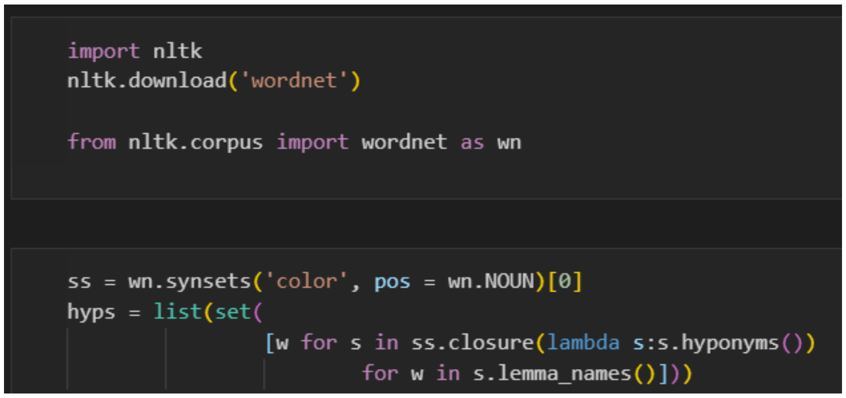
```
# fit model
model = XGBClassifier(random_state=111, use_label_encoder=False,
                      colsample_bytree=0.8, tree_method='auto', max_depth=5,
                      eval_metric='logloss', objective='binary:logistic')
model.fit(X_train, y_train)

# make predictions for test data based on custom threshold
y_pred_direct = model.predict(X_test)
y_pred_proba = pd.DataFrame(model.predict_proba(X_test), columns=['0','1'])
y_pred_custom = np.where(y_pred_proba['1'] > 0.07, 1, 0) #.values.ravel()
```

Appendix 8: Feature Importance



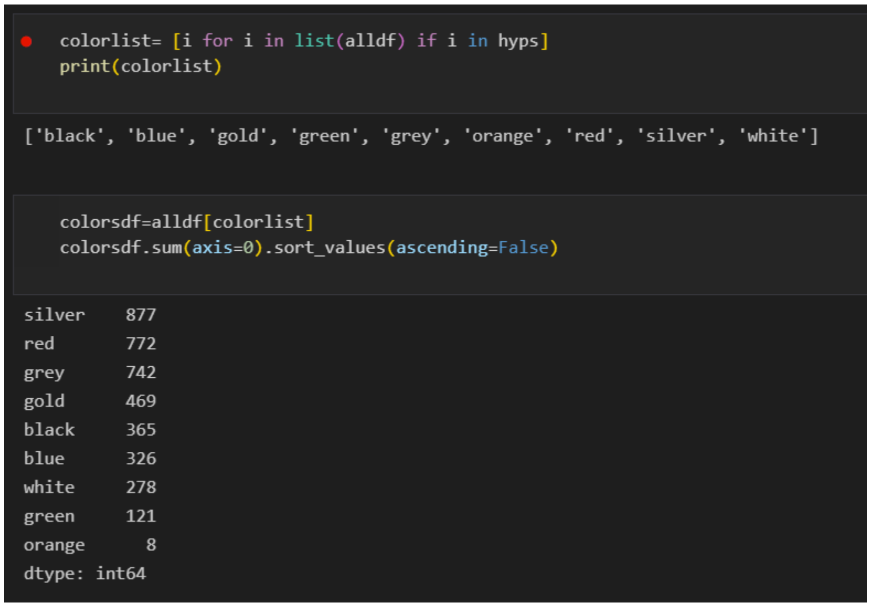
```
# Extract feature importances
fi = pd.DataFrame({'feature': variableList,
                   'importance': rf.feature_importances_}).\
                    sort_values('importance', ascending = False)
```

Appendix 9: Using Wordnet to identify popular features



```
import nltk
nltk.download('wordnet')

from nltk.corpus import wordnet as wn
```

```
ss = wn.synsets('color', pos = wn.NOUN)[0]
hyps = list(set(
                [w for s in ss.closure(lambda s:s.hyponyms())
                        for w in s.lemma_names()]))
```



```
colorlist= [i for i in list(alldf) if i in hyps]
print(colorlist)
```

```
['black', 'blue', 'gold', 'green', 'grey', 'orange', 'red', 'silver', 'white']
```

```
colorsdf=alldf[colorlist]
colorsdf.sum(axis=0).sort_values(ascending=False)
```

```
silver    877
red       772
grey      742
gold      469
black     365
blue      326
white     278
green     121
orange      8
dtype: int64
```