capstone_project

becker

2018-08-03

What is Wishpond?

Wishpond is an All-in-one Marketing Suite for Retailers and Brand. The suite includes social contest and promotions app that permits you to run unlimited vote contests, sweepstakes, and campaigns. It additionally supplies a number of alternative tools such as landing page creators, email marketing and marketing automation.

The Problem?

Wishpond has thousands of customers that use the platform regularly. However, not much has been done to understand its customers and their actions in the platform. Therefore, this capstone project through data analysis, will investigate the different categories customers belong to and their main actions. In addition, it will use machine learning to indentify where customers will fall on and provide Wishpond with substantial data to focus on more specifics subjects that may generate more revenue.

The Data Set

The dataset for this project was acquired from Woopra, a customer service analytics that was installed in the Wishpond platform to track its customers' actions. The database contains a total of 14,417 observations and 21 variables. For better understanding of the database, we will go over 8 of the most important variables to understand Wishpond's customer.

Plan Name – (plan_name)

Plan name identifies which plan the customer currently belongs to. Customers on Wishpond can belong to 4 different plans, they are: the starter plan, which is the plan for people that are just testing the platform and for those who have cancelled their plan. Basic plan, is the cheapest plan with a maximum amount of 1000 leads. Pro, is the intermediate plan, and it offers a maximum of 2500 leads. Growth plan offers unlimited leads.

Business Industry – (business_industry)

Business industry identifies which industry customers belong to such as e-commerce, B2B, spa, fitness and others.

Affiliated – (affiliated)

Affiliated variable shows us if the customer has signed-up on Wishpond via an affiliated link or not.

Sales – (sales)

Sales variable shows us if the customer has signed-up on Wishpond after a demo with one of the sales agent at Wishpond.

Read the blog – (read_blog)

The read the blog variable identify those who have visited the Wishpond blog at one point in time. Wishpond blog is mostly about case-studies and content on lead generation techniques.

Total Leads – (total_leads)

Total leads represent the total amount of leads that the customer has acquired through Wishpond landing page or contest.

Total tools – (total_live_tools)

Total tools represent the total amount of live tools, such as contests, pop-ups, workflows, forms and landing pages, that customers have under their account.

Unsubscribed – (Unsubscribed)

Unsubscribed identifies if a customer has unsubscribed from Wishpond or not.

Data Limitation

The limitation of the database is the large amount of missing data. The plan name variable is missing a third of its data, and other variables have an even larger number of missing data. In addition, a large amount of data from customers that have unsubscribed is lost, as Wishpond reset to zero the total leads and total tools they had under their account. Therefore, making it harder to determine if tools creation and amount of leads plays a role on their subscription.

Data Wrangling

First Part

The main database is formed by three different tables. These tables are: wp_data, upgrade_1 and cust_can. The wp_data and upgrade_1 tables have the same variables, the difference between them is that upgrade_1 has more information about customers that have upgraded to a paid account, while wp_data is more focused on clients that have assigned the type of industry they belong to. Lastly the cust_can table represents all the customer that have unsubscribed from Wishpond. Below shows the code of how these tables were combined into one. First we transformed the factor variables from the three tables into characters in order to not lose any information when comparing the variables from different tables.

```
#Transforming variables into characters to be able to use join function
wp_data[c(1:6, 12:16, 19)] <- lapply(wp_data[c(1:6, 12:16, 19)], as.character)
upgraded_1[c(1:6, 12:16, 19)] <- lapply(upgraded_1[c(1:6, 12:16, 19)], as.character)
cust_can$pid <- as.character(cust_can$pid)
```

Second, we run the anti_join function to remove any duplicates from the upgrade_1 table and then we add the table to the wp_data with bind_rows function.

```
# Here, we used anti_join to avoid having duplicates.
upgraded_1 <- anti_join(upgraded_1, wp_data, by = "pid")

#Combining Rows from different tables
wp_data <- bind_rows(wp_data, upgraded_1)
```

Third, to know all the customers from the database that have unsubscribed, it was used the left_join function. Therefore, maintaining all the observations on wp_data while adding a new column, Unsubscribed.

```
# Using left_join to maintain all the current data from "wp_data" while adding a new column,
Unsubscribed, from "cust_can"
wp_data <- left_join(wp_data, cust_can, by = c("pid" = "pid"))
```

Lastly, during the join, empty columns were added. Thus, we get rid off with the code below.

```
# Get rid off unwanted columns
wp_data <- wp_data[, -c(22:34)]
```

Second Part

The database contained a lot of missing data, which made it difficult to read it. Therefore, empty values and NAs were replaced in order to create a more meaningful database.

For the missing data and NAs value on business industry, it was assigned the value "Unknown"

```
wp_data$business_industry <- gsub("^$", "Unknown", wp_data$business_industry)
wp_data <- wp_data %>%  replace_na(list(business_industry = "Unknown"))
```

For the missing data on selected_service, it was assigned the value "Unknown"

```
wp_data$selected_service <- gsub("^$", "Unknown", wp_data$selected_service)
```

For all the missing and null data on business_size, it was assigned the value "Unknown"

```
wp_data$business_size[wp_data$business_size == "null"] <- "Unknown"
wp_data$business_size <- gsub("^$", "Unknown", wp_data$business_size)
```

For the missing data on cancel_contact_support, it was assigned the value "No", as only if a customer that has contacted support received a value of "Yes"

```
wp_data$current_cancel_contact_support <- gsub("^$", "No",
wp_data$current_cancel_contact_support)
```

For the missing data on current_cancel_feeling, it was assigned the value "Unknown"

```
wp_data$current_cancel_feeling <- gsub("^$", "Unknown", wp_data$current_cancel_feeling)
```

For the missing data on has_seen_the_blog_overlay, it was assigned the value "FALSE" as all customers that have read the blog receveid the “TRUE” value

```
wp_data <- wp_data %>%  replace_na(list(has_seen_the_blog_overlay = "FALSE"))
```

For the missing data on support_agent_name, it was assigned the value “Unknown”

```
wp_data$support_agent_name <- gsub("^$", "Unknown", wp_data$support_agent_name)
```

For the missing data on locale, it was assigned the value “Unknown”

```
wp_data$locale <- gsub("^$", "Unknown", wp_data$locale)
```

For the missing data on plan_name, it was assigned the value “Unknown”

```
wp_data$plan_name <- gsub("^$", "Unknown", wp_data$plan_name)
```

For the missing data on sales_agent, it was assigned the value “Unknown”

```
wp_data$sales_agent <- gsub("^$", "Unknown", wp_data$sales_agent)
```

For the missing data on interest, it was assigned the value “Unknown”

```
wp_data$interest <- gsub("^$", "Unknown", wp_data$interest)
```

For the missing data on sales, it was assigned the value “FALSE” as all customers that have subscribed after a demo receveid the “TRUE” value

```
wp_data <- wp_data %>%  replace_na(list(sales = "FALSE"))
```

For the missing data on affiliated, it was assigned the value “FALSE” as all customers that have subscribed after coming thorugh an affiliated link receveid the “TRUE” value

```
wp_data <- wp_data %>%  replace_na(list(affiliated = "FALSE"))
```

For the missing data on Unsubscribed, it was assigned the value “FALSE” as all customers that have unsubscribed from Wishpond receveid the “TRUE” value

```
wp_data <- wp_data %>% replace_na(list(Unsubscribed = "FALSE"))
```

Third Part

Some variables had a lot of different levels that could be represented by a unique level. Thus, the levels of these variables were summarized by using the code below:

All the values containing a website will be replace with the “TRUE” value, while assigning “FALSE” to the empty rows

```
wp_data$website[wp_data$website != ""] <- "TRUE"
wp_data$website[wp_data$website != "TRUE"] <- "FALSE"
```

It will be assigned the “Self Service” value to all the rows containing “self” and “Fully Managed” to all the rows containing “fully”

```
wp_data$selected_service <- gsub(".*self.*", "Self Service", wp_data$selected_service)
wp_data$selected_service <- gsub(".*fully.*", "Fully Managed", wp_data$selected_service)
```

Some plans at Wishpond have a name added to the plan they upgraded to if a special discount was given. Therefore, in order to simplify the variable, it was assigned “Basic” to all the rows containing “Basic”, and “Growth Plan” to all the rows containing “Growth”

```
str(wp_data$plan_name)
##  chr [1:14147] "Basic" "Basic" "Unknown" "Starter" "Basic" ...
wp_data$plan_name <- as.factor(wp_data$plan_name)
levels(wp_data$plan_name)
## [1] "Basic"                      "Basic Annual Commitment"
## [3] "Basic Plan KO special $19"   "Basic Plan KO special Basic"
## [5] "Growth Annual 15k"           "Growth Plan"
## [7] "Pro"                        "Starter"
## [9] "Unknown"
wp_data$plan_name <- gsub(".*Basic.*", "Basic", wp_data$plan_name)
wp_data$plan_name <- gsub(".*Growth.*", "Growth Plan", wp_data$plan_name)
```

When customers subscribe to Wishpond, they get to write the why they decide to subscribe to the platform. Below is the code to simplify all the different values entered, for easier understanding of the variable.

```
# Code used to replace all the values that are related to the contest tool at Wishpond.
wp_data$interest <-
gsub(".*coupon.*|.*photo.*|.*instagram*|.*referral.*|.*sweepstake.*|.*text.*|.*video.*|.*vote.*|.*pi
nterest.*", "Contest", wp_data$interest)
```

```r
# Code used to replace all the values that are related to the pop-up tool
wp_data$interest <- gsub(".*forms.*|.*action*|.*pop.*", "Pop-Up", wp_data$interest)

# Code used to replace all the values that are related to the Landing Page tool
wp_data$interest <- gsub(".*landing.*", "Landing Page", wp_data$interest)
```

In order to evaluate the total amount of live tools by each customer, live pop-ups, live workflow, live contests and live landing pages was combined into a unique variable total live tools.

```r
# Sum of all the live tools on Wishpond to total_live_tools
wp_data <- wp_data %>%  mutate(total_live_tools =  live_popups + live_workflows +
live_landings + live_contests)
```

There was too much differentiation on the amount of leads and amount of total live tools by each customer, making it harder to analyze the most frequent level for these two variable. Therefore, we decided to categorize into fewer levels based on a range of leads captured and total live tools.

Business industry variable had too much differentiation and only a few industries had frequent values. Therefore, in order to evaluate the most frequent business industries it was used the group_by function to group the top 15 industries and then rename the other industries to “Other”

```r
# Discovering the top 15 business industries
# group by video ---> https://www.youtube.com/watch?v=jWjqLW-u3hc --> 24min
biz_grouped <- wp_data %>% group_by(business_industry)
top_biz <- biz_grouped %>% count(business_industry) %>% arrange(desc(n))
top_15_biz <- head(top_biz, 15)

# Filtering the top 15 industries
top_biz_db <- wp_data %>% group_by(business_industry) %>% filter(n() >= 131,
business_industry != "Unknown")
top_biz_col <- wp_data %>% group_by(business_industry) %>% filter(n() >= 131,
business_industry != "Unknown") %>% select(pid, business_industry)

# Renaming businesses industry that are not part of the top 15
other_biz_col <- wp_data %>% group_by(business_industry) %>% filter(n() < 131,
business_industry != "Unknown") %>% select(pid, business_industry)
other_biz_col$business_industry <-  "Other"
biz_unknown <- wp_data %>% filter(business_industry == "Unknown") %>% select(pid,
business_industry)
biz_unknown$business_industry <- as.character(biz_unknown$business_industry)
top_biz_col$business_industry <- as.character(top_biz_col$business_industry)
other_biz_col$business_industry <- as.character(other_biz_col$business_industry)
all_biz <- bind_rows(top_biz_col, other_biz_col, biz_unknown)
wp_data$business_industry <- all_biz$business_industry
```

Fourth Part

Lastly, some variables were renamed and rearranged for better visualization of the database.

```r
#Renaming column name
wp_data <- wp_data %>% rename( read_blog = has_seen_the_blog_overlay)
wp_data <- wp_data %>% rename( language = locale)
wp_data <- wp_data %>% rename( support_agent = support_agent_name)
wp_data <- wp_data %>% rename( cancel_contact_support = current_cancel_contact_support)
wp_data <- wp_data %>% rename( cancel_feeling = current_cancel_feeling)

#Rearranging the Data-base
wp_data <- wp_data %>% select(pid, business_industry, business_size, selected_service,
live_workflows, live_popups, total_live_tools, live_contests, live_landings, total_leads, interest,
read_blog, sales_agent, sales, affiliated, plan_name, website, support_agent,
cancel_contact_support, cancel_feeling, Unsubscribed)
```

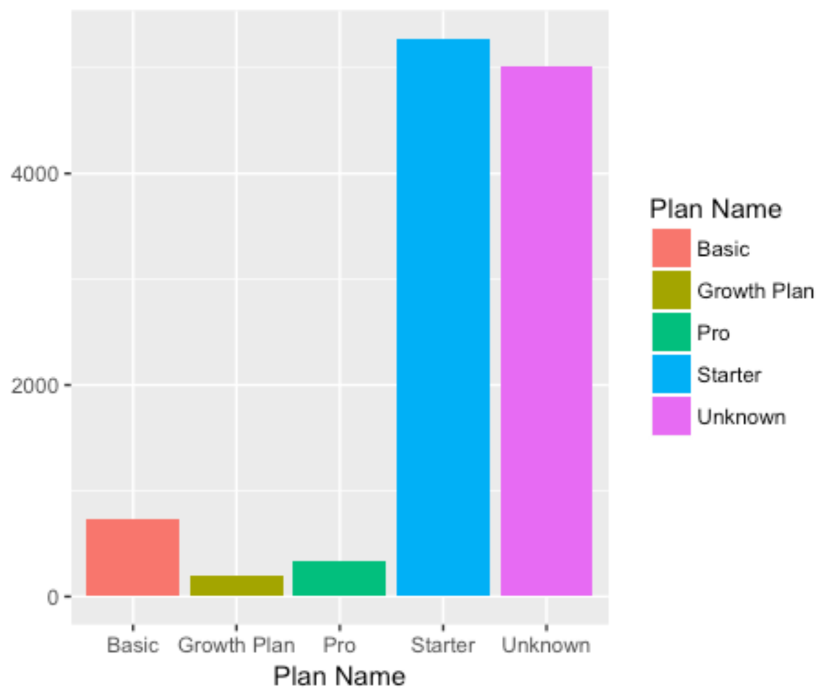
Data Exploration

In order to identify which data might be useful in predicting the main factors for someone to upgrade their account on Wishpond, exploratory analysis was performed to compare the different variables to the customers’ plan.

All Wishpond Subscribers

The graph below show us what the most popular plans at Wishpond are. The interesting thing here is that most of the customers are on the Starter plan, which means that they have not paid Wishpond yet. Even though starter plan does not contain much data, it shows a great opportunity for Wishpond to grow. If we understand what it takes customers to use Wishpond, we might be able to get people from the Starter plan to sign-up with Wishpond.

```r
wp_data_subscribed_all <- wp_data %>%  filter(Unsubscribed == "FALSE")
ggplot(wp_data_subscribed_all, aes(plan_name)) +
 geom_bar(stat = "count", aes(fill = plan_name)) +
 xlab("Plan Name") +
 ylab("") +
 ylab("") +
 scale_fill_discrete(name = "Plan Name")
```



Paying Subscribers

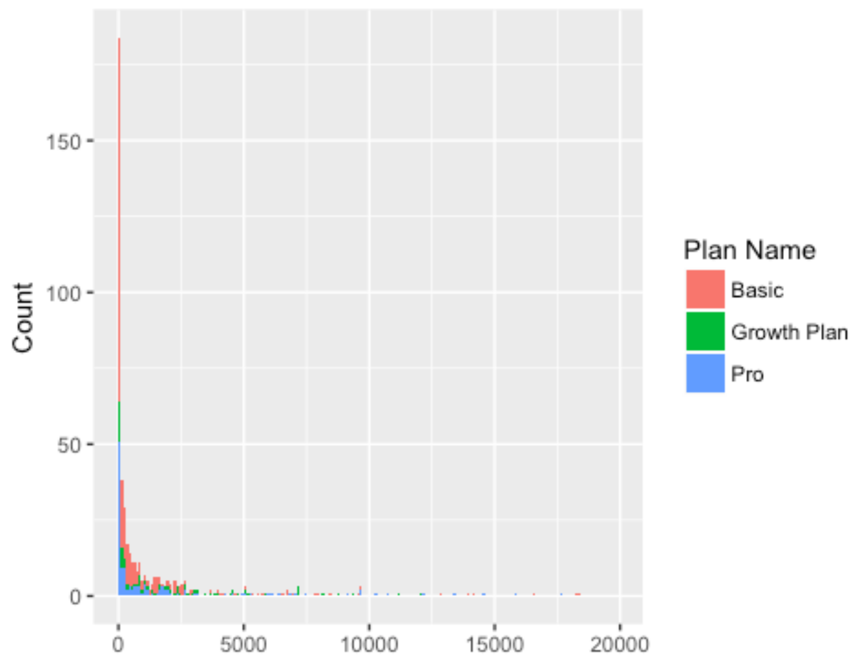
Next we will focus on paying customers only. Thus, we will filter out all the “Unknown” and “Starters” values from plan name variable.

```
wp_data_subscribed <- wp_data %>% filter(Unsubscribed == "FALSE", plan_name !=
"Unknown", plan_name != "Starter")
```

Total Amount of Leads

For this graph, we start with those who have acquired at least one lead, as if we count those who haven’t captured any lead, it will distort the graph as the number is much greater than any other. This is not the best signal as it means that there are lot of customers that are not being able to generate any lead. Another interesting point to analyze is that there are a lot of customers on the basic plan that have acquired more leads than the maximum allowed under their current plan (500 leads), which means that they upgraded their account to run a campaign and then went back to basic plan.

```
ggplot(wp_data_subscribed, aes(total_leads, fill = plan_name)) +
 geom_histogram(breaks = seq(1,20000, by = 100))  +
 labs(x = "Total Leads", y = "Count") +
 scale_fill_discrete(name = "Plan Name")
```

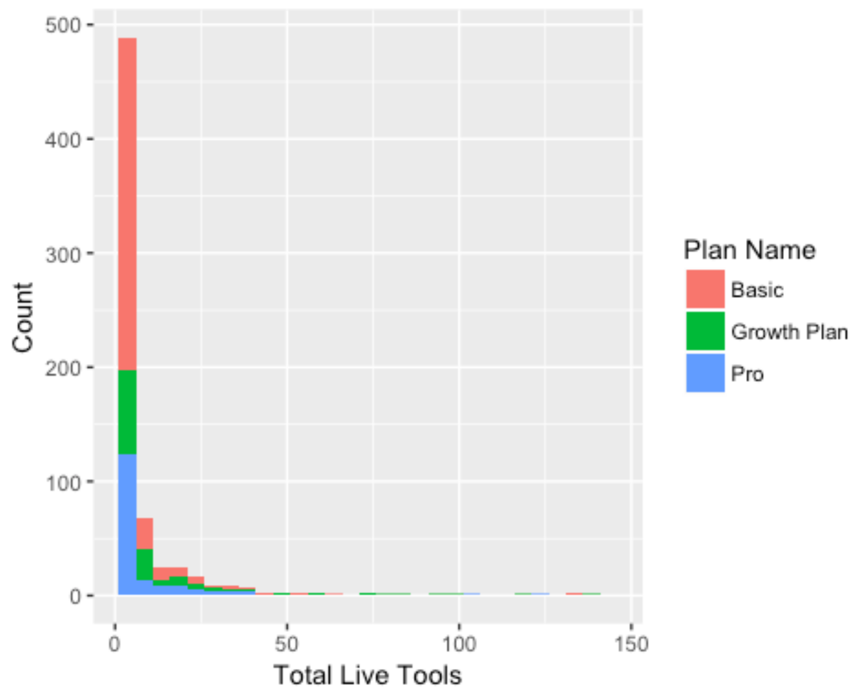


Total Leads

Total Amount of Live Tools

For this graph, we start with those who have created at least 1 tool, because if we count those who haven't created anything, it will distort the graph as the number of zero live tool is much greater than any other. Another interesting point to analyze is that most of those who have more than 25 live tools are on the Pro and Growth plan.

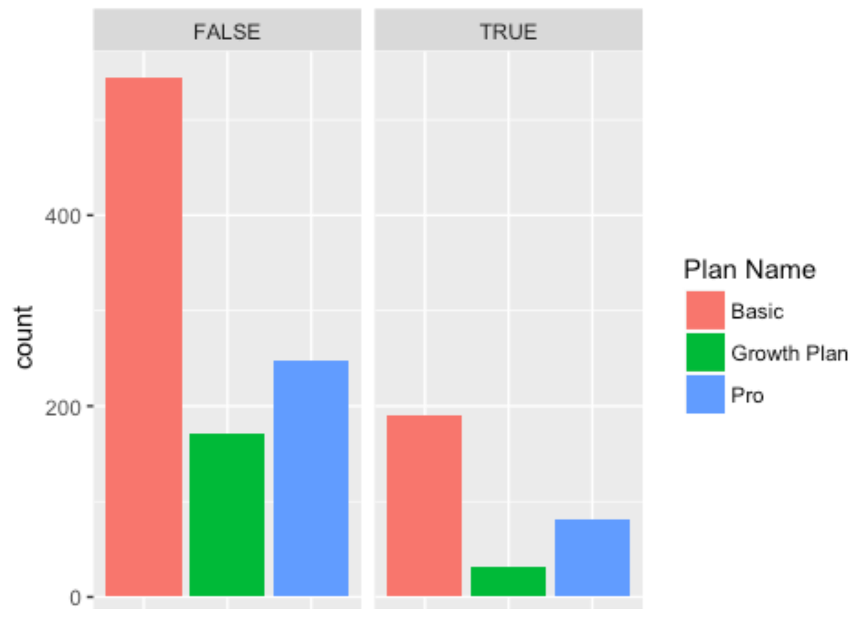
```
ggplot(wp_data_subscribed, aes(total_live_tools, fill = plan_name)) +
 geom_histogram(breaks = seq(1,150, by = 5)) +
 labs(x = "Total Live Tools", y = "Count") +
 scale_fill_discrete(name = "Plan Name")
```

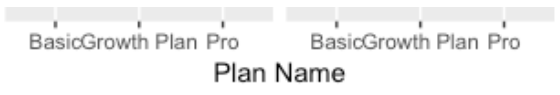


Sales

The graph shows that most of the subscriptions are not coming through sales agents, but through other means. The sales team is fairly expensive with a lot of personnel. Thus, Wishpond could reduce expenses by decreasing the sales team and increase revenue by focusing on the different avenues that are bringing more customers.

```
ggplot(wp_data_subscribed, aes(factor(wp_data_subscribed$plan_name))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_subscribed$plan_name))) +
 facet_grid(. ~ factor(wp_data_subscribed$sales)) +
 xlab("Plan Name") +
 scale_fill_discrete(name = "Plan Name")
```

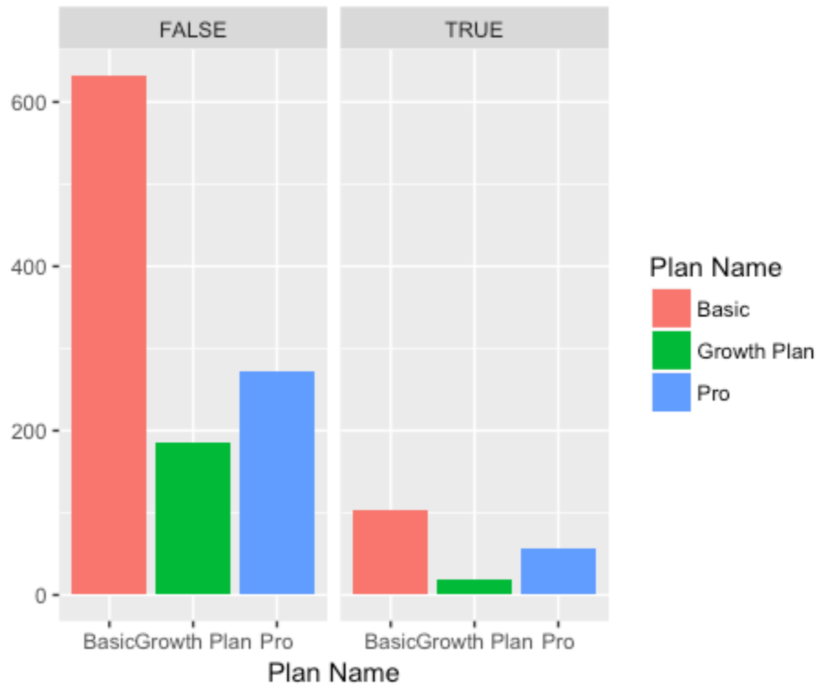




Affiliated

While affiliated links have not generated as many sales as sales agents, the number are not too far off. Also, the amount of money spent on an affiliated link is much less than a sales agent salary. Pushing the affiliated program could be a good alternative to bring more customers to Wishpond at a lower cost.

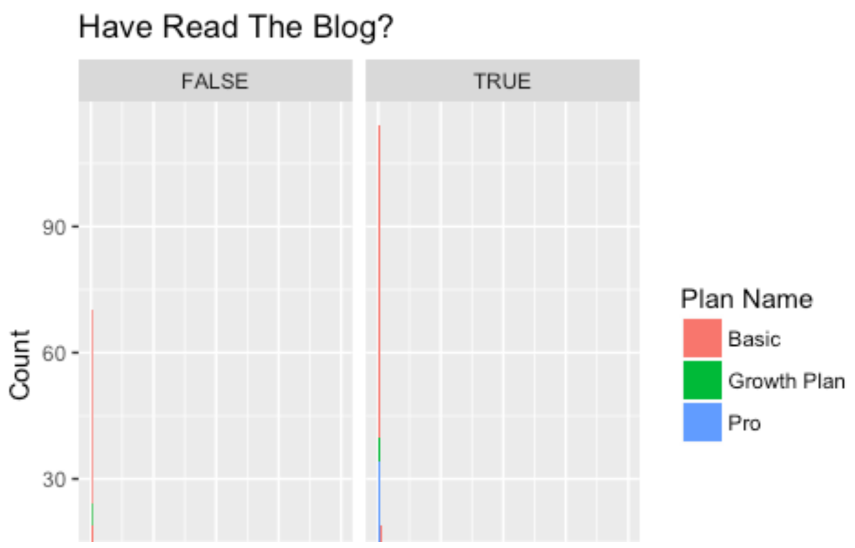
```
ggplot(wp_data_subscribed, aes(factor(wp_data_subscribed$plan_name))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_subscribed$plan_name))) +
 facet_grid(. ~ factor(wp_data_subscribed$affiliated)) +
 xlab("Plan Name") +
 ylab("") +
 scale_fill_discrete(name = "Plan Name")
```

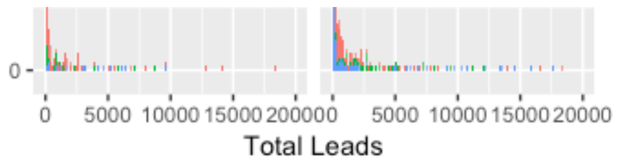


Have Read Wishpond Blog

The graph below shows us that those who have read the blog are generating more leads than those who have not read the blog. Thus, showing us that the blog might help customers to capture leads.

```
ggplot(wp_data_subscribed, aes(total_leads, fill = plan_name)) +
 geom_histogram(breaks = seq(1,20000, by = 100))  +
 facet_grid(. ~ factor(wp_data_subscribed$read_blog)) +
 labs(title = "Have Read The Blog?", x = "Total Leads", y = "Count") +
 scale_fill_discrete(name = "Plan Name")
```

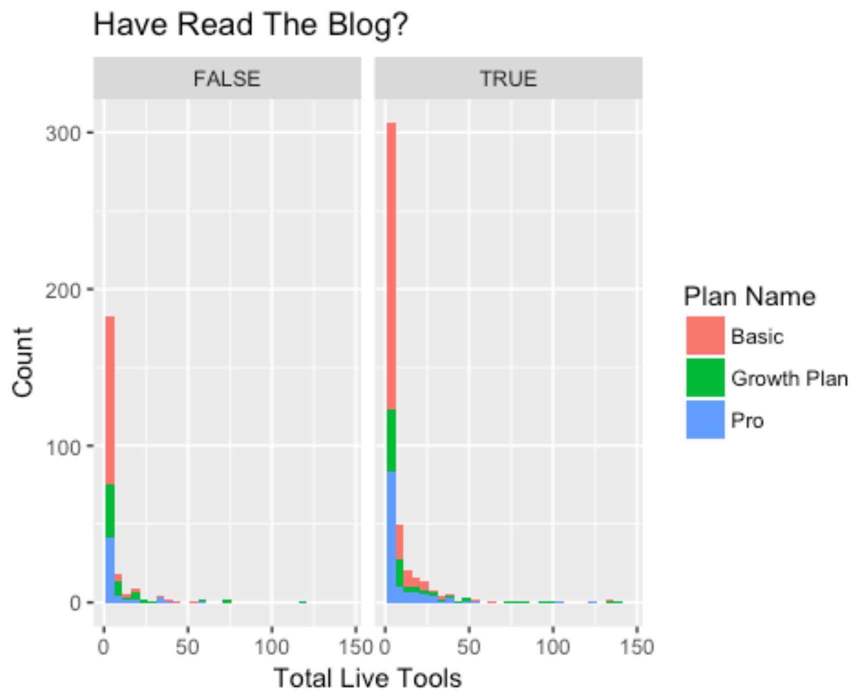




Have Read Wishpond Blog and Total Amount of Leads

The graph below shows us that those who have read the blog create more tools than those who have not. Thus, showing that the blog might influence on supporting customer to get started with Wishpond and create different tools on the platform.

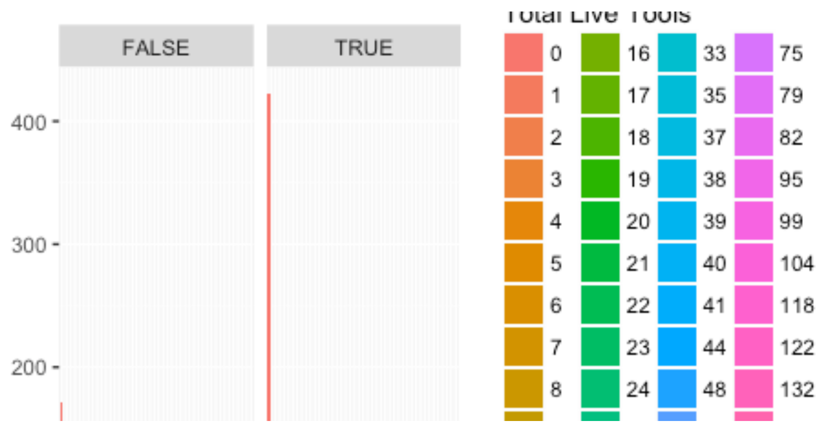
```
ggplot(wp_data_subscribed, aes(total_live_tools, fill = plan_name)) +
 geom_histogram(breaks = seq(1,150, by = 5)) +
 facet_grid(. ~ factor(wp_data_subscribed$read_blog)) +
 labs(title = "Have Read The Blog?", x = "Total Live Tools", y = "Count") +
 scale_fill_discrete(name = "Plan Name")
```

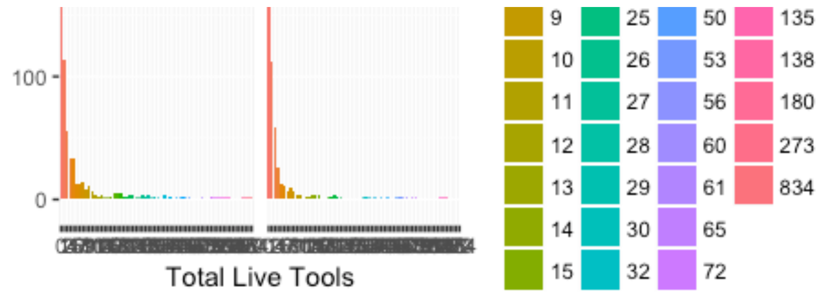


Have Read Wishpond Blog and Total Amount of Leads

Similar to the amount of leads, we want to see how the blog is influencing customers on creating different tools at Wishpond. The interesting factor on this graph is that most of the customers that have read the blog are not using Wishpond tools as much as those who have not read it. Therefore, Wishpond should allocate some space to talk about the importance of using workflows, pop-up and other tools, in order to get customers to use more of Wishpond tools.

```
ggplot(wp_data_subscribed, aes(factor(wp_data_subscribed$total_live_tools))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_subscribed$total_live_tools))) +
 facet_grid(. ~ factor(wp_data_subscribed$read_blog)) +
 xlab("Total Live Tools") +
 ylab("") +
 scale_fill_discrete(name = "Total Live Tools")
```



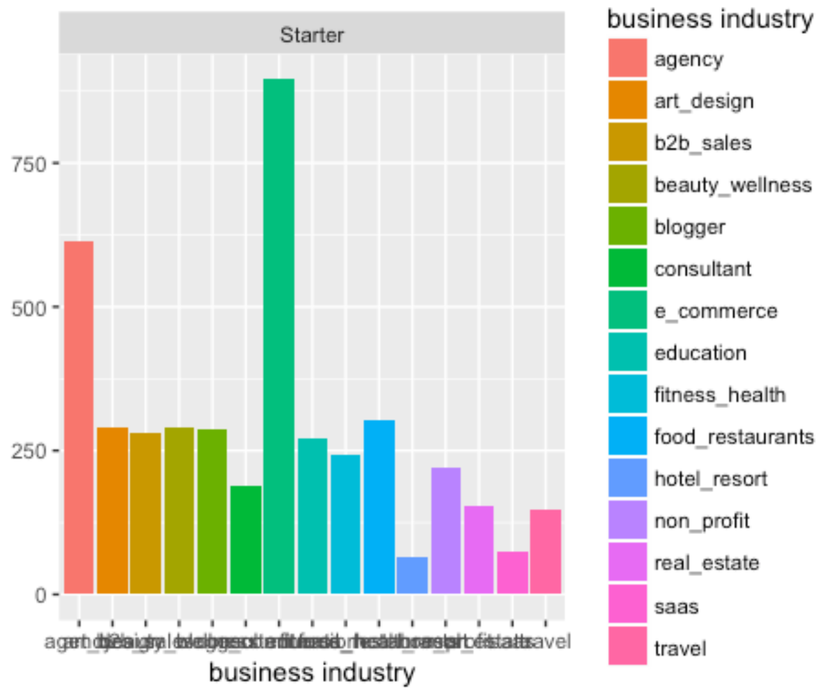


Business Industry At Each Plan

From the graph below we can see that e-commerce is the leading business industry on Wishpond, while agency has a good share of customers. By analysing the graph for the starter plan, we can understand where most of the opportunity to turn customers to paid customers are, e-commerce and agency. At the Growth plan, the most expensive plan, e-commerce is by far the most predominant business industry. Thus, it allows us to see where Wishpond should start focusing in order to bring more customers.

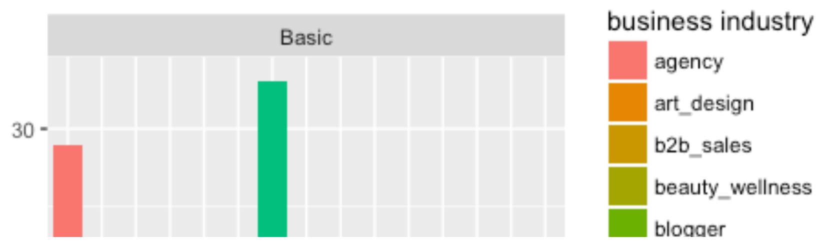
Starter Plan & Business Industry

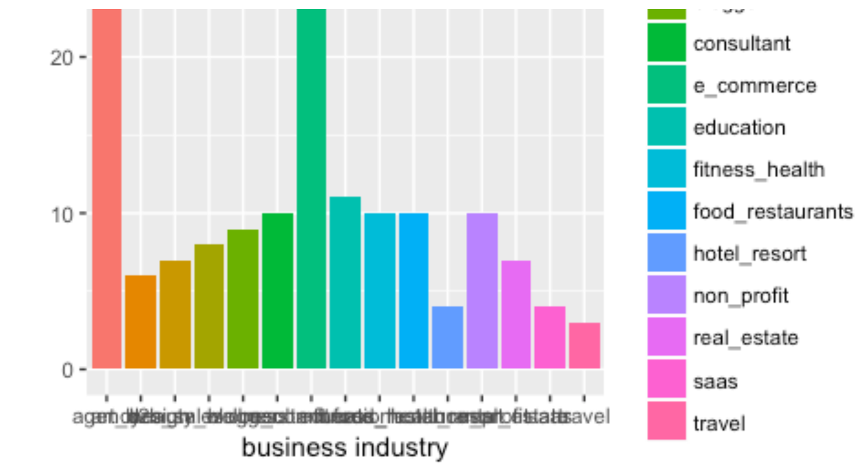
```
wp_data_starter <- wp_data %>% filter(plan_name == "Starter", Unsubscribed == "FALSE",
business_industry != "Other", business_industry != "Unknown")
ggplot(wp_data_starter, aes(factor(wp_data_starter$business_industry))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_starter$business_industry))) +
 facet_grid(. ~ factor(wp_data_starter$plan_name)) +
 xlab("business industry") +
 ylab("") +
 scale_fill_discrete(name = "business industry")
```



Basic Plan & Business Industry

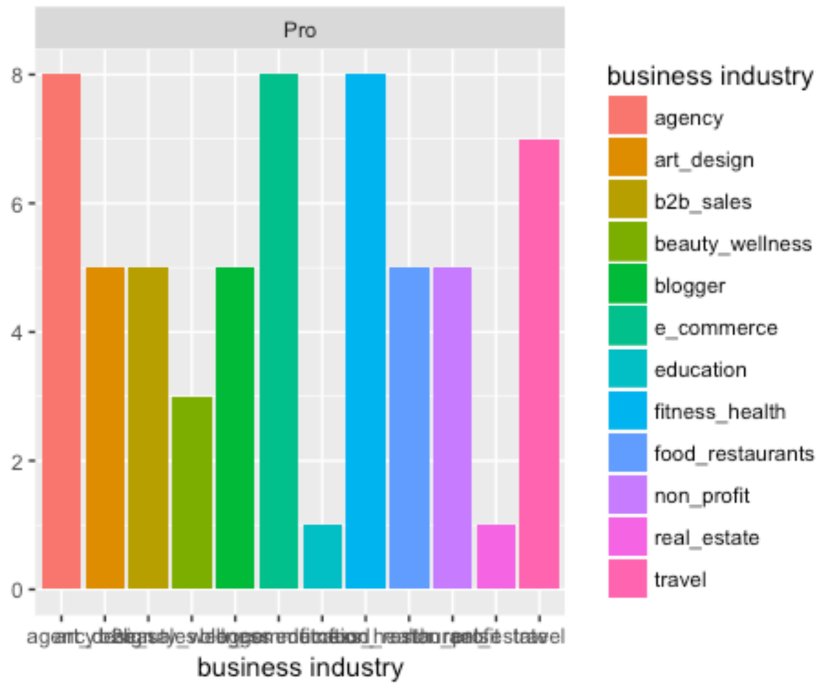
```
wp_data_basic <- wp_data %>% filter(plan_name == "Basic", Unsubscribed == "FALSE",
business_industry != "Other", business_industry != "Unknown")
ggplot(wp_data_basic, aes(factor(wp_data_basic$business_industry))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_basic$business_industry))) +
 facet_grid(. ~ factor(wp_data_basic$plan_name)) +
 xlab("business industry") +
 ylab("") +
 scale_fill_discrete(name = "business industry")
```





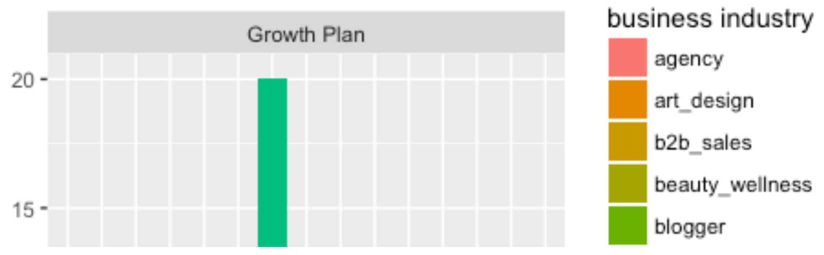
Pro Plan & Business Industry | Second Most Expensive Plan | E-Commerce by far the most relevant one. Art-design and fitness health industry coming in second

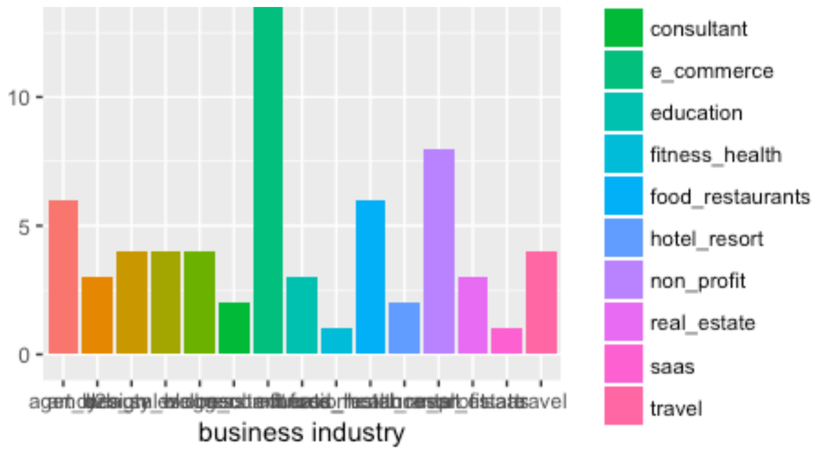
```
wp_data_pro <- wp_data %>% filter(plan_name == "Pro", Unsubscribed == "FALSE",
business_industry != "Other", business_industry != "Unknown")
ggplot(wp_data_pro, aes(factor(wp_data_pro$business_industry))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_pro$business_industry))) +
 facet_grid(. ~ factor(wp_data_pro$plan_name)) +
 xlab("business industry") +
 ylab("") +
 scale_fill_discrete(name = "business industry")
```



Growth Plan & Business | Most Expensive Plan | E-Commerce by far again bringing the most customers to Growth plan.

```
wp_data_growth <- wp_data %>% filter(plan_name == "Growth Plan", Unsubscribed ==
"FALSE", business_industry != "Other", business_industry != "Unknown")
ggplot(wp_data_growth, aes(factor(wp_data_growth$business_industry))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_growth$business_industry))) +
 facet_grid(. ~ factor(wp_data_growth$plan_name)) +
 xlab("business industry") +
 ylab("") +
 scale_fill_discrete(name = "business industry")
```

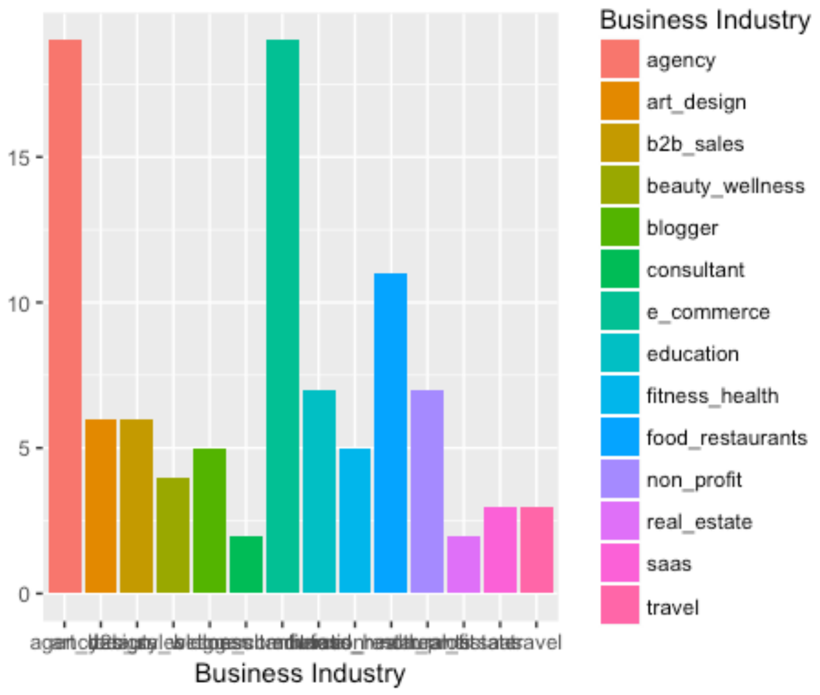




Business Industry and Unsubscribed

The graph below shows us that e-commerce and agency are the business industries with the most cancels, similar to the highest amount of customers. However, by knowing that these are the greatest numbers of cancel, we can learn what keep customers from these business industries subscribeed at Wishpond and eventually decrease the number of cancels.

```
wp_data_unsubscribed <- wp_data %>% filter(Unsubscribed == "TRUE", business_industry !=
"Other", business_industry != "Unknown")
ggplot(wp_data_unsubscribed, aes(factor(wp_data_unsubscribed$business_industry))) +
 geom_bar(stat = "count", aes(fill = factor(wp_data_unsubscribed$business_industry))) +
 xlab("Business Industry") +
 ylab("") +
 scale_fill_discrete(name = "Business Industry")
```



Machine Learning

Overview

This project will use machine learning to predict the risk of a customer unsubscribing from Wishpond.

This is a supervised problem because we are attempting to predict a specific dependent variable using a model based on a set of independent variables.

This is a classfication problem

The independent variable is Unsubscribed, FALSE or TRUE.

This project used different techniques, first it used multiple imputation to replace the missing values with plausible values through probability. Secondly, it used Lasso Regularization to identify the variables that may cause overfitting to our model. Lastly, it *used two different predictive models to evaluate which can perform better, logisitic regression and random forest regression.*

To test the predictive models, the data will be split into two different datasets train and test.

To evaluate the predictive models the data will be splitted into two different datasets train (holding larger data) and test (holding a smaller amount of data), in order to provide an unbiased evaluation of the final model.

After comparing the two models, random forest regression producde a better model than the logistic regression for over 1%, with an accuracy of 76.1%

Code

Because our goal is to identify who are at risk of unsubscribing, we will filter the database to those who are paying and who have unsubscribed from the platform

```
wp_data_ml <- wp_data
wp_data_ml <- wp_data_ml %>% filter( plan_name == "Basic" | plan_name == "Pro" |
plan_name == "Growth" | Unsubscribed == "TRUE")
```

The database has a lot of missing values, which are described as “Unknown” or as “Other” in the database. Therefore, we will replace these values with NAs in order to use multiple imputation and replace with predicted values rather than “Unknown” and “Other”

```
wp_data_ml[wp_data_ml == "Unknown"] <- NA
wp_data_ml[wp_data_ml == "Other"] <- NA
```

Some support agents have really low amount of customers because they are no longer with the company. Therefore, we will exclude those with minimun amount of customer to be replaced with NA and later be replaced with multiple imputation.

```
support_agent_del <- wp_data_ml %>% group_by(support_agent) %>% filter(n() < 10)
support_agent_del$support_agent <- NA
support_agent_ml <- wp_data_ml %>% group_by(support_agent)  %>% filter(n() >= 10)
support_agent_ml <- bind_rows(support_agent_del, support_agent_ml)
wp_data_ml$support_agent <- support_agent_ml$support_agent
```

When customers unsubscribe, they are automatically sent to Starter plan, which means that Starter is not the plan they were subscribed to. Thus, we will turn all the starters plan to NAs and then use multiple imputation to replace it with the possible plans the were once subscribed to.

```
wp_data_ml$plan_name[wp_data_ml$plan_name == "Starter"] <- NA
```

Most of the variable in the database are represented as characters, and to use the multiple imputation function, we will need to transform them into factors.

```
wp_data_ml[c(2:4, 11:21)] <- lapply(wp_data_ml[c(2:4, 11:21)], as.factor)
```

To use multiple imputation, we will select all the variables that have missing values into a new database, then we will run the function on those variables. After that, we will replace the new variables into the real database.

Basically, what multiple imputation will do is to replace missing values with plausible values through probability.

```
library("mice")
simple <- wp_data_ml[c("business_industry", "business_size", "interest", "sales_agent",
"plan_name", "website", "support_agent", "cancel_feeling")]
set.seed(98)
imputed <- mice::complete(mice(simple))
## Warning: Number of logged events: 27
```

Now, we will replace the old variables with missing data with the new imputed variables that have no NAs.

```
wp_data_ml$business_industry <- imputed$business_industry
wp_data_ml$business_size <- imputed$business_size
wp_data_ml$interest <- imputed$interest
wp_data_ml$sales_agent <- imputed$sales_agent
wp_data_ml$website <- imputed$website
wp_data_ml$support_agent <- imputed$support_agent
wp_data_ml$cancel_feeling <- imputed$cancel_feeling
wp_data_ml$plan_name <- imputed$plan_name
```

There are some variables that are not needed and therefore, we will exclude from the database. All the variables with live related tools are going to be exlcuded because it doesnt reflect the amount of live tools customers had correctly since when a customer unsubscribe it is automatically set to 0.

```
wp_data_ml <- wp_data_ml %>% select(-pid, -selected_service, -live_workflows, -live_popups, -
live_contests, -live_landings, -total_live_tools)
```

Before applying regression analysis, we will run Lasso Regularization to see what variables may cause overfitting in our model, thus indicating which variables will best fit our model.

In order to run Lasso we need all variables to be integers.

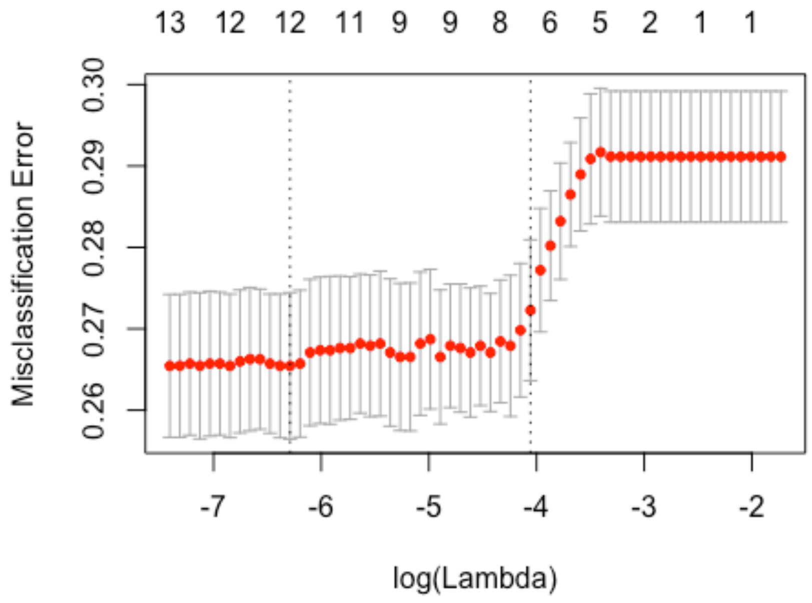
```
library(glmnet)
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following object is masked from 'package:tidyr':
##
##     expand
## Loading required package: foreach
## Loaded glmnet 2.0-16
wp_data_num <- wp_data_ml
wp_data_num[c(1:2, 4:14)] <- lapply(wp_data_num[c(1:2, 4:14)], as.integer)
wp_data_num[c(1:2, 4:14)] <- wp_data_num[c(1:2, 4:14)] - 1
wp_data_num[c(1:2, 4:14)] <- lapply(wp_data_num[c(1:2, 4:14)], as.integer)
```

Then we need to separate the independent variable from the dependent variables

```
unsubscribed_db <- wp_data_num$Unsubscribed
wp_data_num <- wp_data_num %>% select(-Unsubscribed)
```

Now we will run a cv.glmnet function to identify the best lambda values, which can be identified through the graph, below. The line of the left shows us the lambda value with the minimun misclassification errors, while the line on the right shows us the most regularized model that is still within 1 standard error from our lambda value with minimun misclassification error.

```
CV <- cv.glmnet(as.matrix(wp_data_num), unsubscribed_db, family="binomial", type.measure =
"class", alpha = 1, nlambda=100)
plot(CV)
```



On the function below we will use the lambda values we found on the function above to show us what are the variables that may cause overfitting in our model. The variables that may cause overfitting will be shown with a coefficient of 0.

```
fit <- glmnet(as.matrix(wp_data_num), unsubscribed_db, family="binomial", alpha = 1,
lambda=CV$lambda.1se)
fit$beta[,1]
##     business_industry        business_size          total_leads
##            0.00000000           0.00000000           0.00000000
##              interest            read_blog          sales_agent
##           -0.24297123           0.00000000           0.00000000
##                 sales           affiliated            plan_name
##           -0.15802921          -0.30129253           0.00000000
##               website        support_agent cancel_contact_support
##           -0.40612507          -0.02641073           3.09780109
##        cancel_feeling
##            0.00000000
```

Now for the regression analysis, we will split the database into a training dataset and test dataset. And because we have a large database, we decided to use a split ratio of 65% on the training database and 35% on the test database.

```
set.seed(1000)
split <- sample.split(wp_data_ml$Unsubscribed, SplitRatio = 0.65)
train <- subset(wp_data_ml, split == TRUE)
test <- subset(wp_data_ml, split == FALSE)
```

Below, is the logistic regression function, comparing the independent variable "Unsubscribed" against all the other dependent variables.

```
wp_data_Log <- glm(Unsubscribed ~ business_industry + total_leads + interest + sales_agent +
sales + affiliated + website + support_agent + cancel_contact_support, data = train, family =
binomial)
summary(wp_data_Log)
## 
## Call:
## glm(formula = Unsubscribed ~ business_industry + total_leads + 
##     interest + sales_agent + sales + affiliated + website + support_agent + 
##     cancel_contact_support, family = binomial, data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -3.5851  -0.8666   0.1048   0.8415   2.0868  
## 
## Coefficients:
##                                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -1.080e-01  4.775e-01  -0.226 0.821122
## business_industryart_design        1.044e-01  3.865e-01   0.270 0.787158
## business_industryb2b_sales         6.548e-02  3.441e-01   0.190 0.849099
## business_industrybeauty_wellness  -1.576e-01  2.852e-01  -0.553 0.580428
## business_industryblogger           1.826e-01  3.242e-01   0.563 0.573284
## business_industryconsultant       -2.070e-01  3.048e-01  -0.679 0.497054
## business_industrye_commerce       -1.364e-01  2.169e-01  -0.629 0.529597
## business_industryeducation        -3.509e-01  2.656e-01  -1.321 0.186401
## business_industryfitness_health   -9.685e-02  2.658e-01  -0.364 0.715629
## business_industryfood_restaurants  2.635e-01  3.346e-01   0.788 0.430974
## business_industryhotel_resort     -1.140e+00  5.660e-01  -2.014 0.044012
## business_industrynon_profit        3.313e-01  3.310e-01   1.001 0.316910
## business_industryreal_estate      -7.993e-01  4.922e-01  -1.624 0.104398
## business_industrysaas              5.043e-01  3.692e-01   1.366 0.171996
## business_industrytravel           -2.941e-01  4.454e-01  -0.660 0.509090
## total_leads                       -9.756e-06  7.549e-06  -1.292 0.196242
## interestLanding Page              -5.542e-01  1.221e-01  -4.540 5.61e-06
## interestPop-Up                    -5.486e-01  1.661e-01  -3.302 0.000959
## sales_agentAlex Baranovsky        -3.553e-01  3.072e-01  -1.157 0.247380
## sales_agentDan Novak               2.656e-01  5.555e-01   0.478 0.632603
## sales_agentInga Zane              -5.671e-01  3.908e-01  -1.451 0.146779
## sales_agentJason Bradbury         -3.179e-01  2.249e-01  -1.413 0.157509
## sales_agentJordan Gutierrez       -8.549e-02  4.930e-01  -0.173 0.862339
## sales_agentKemal Wahju            -1.464e-01  4.469e-01  -0.327 0.743307
## sales_agentKevin Ho               -1.219e-01  3.076e-01  -0.396 0.691848
## sales_agentlaura                  -2.509e-01  2.035e-01  -1.233 0.217629
## sales_agentMarlyn                 -1.016e+00  1.246e+00  -0.816 0.414706
## sales_agentMoe Tajsekandar        -1.500e-01  2.131e-01  -0.704 0.481390
## sales_agentRose Chua              -3.880e-01  4.844e-01  -0.801 0.423173
## sales_agentRoselyn Bonito          3.481e-01  6.091e-01   0.572 0.567635
## sales_agentSam Moghaddam          -3.311e-01  3.111e-01  -1.064 0.287250
## sales_agentSara McKenzie          -1.002e-01  4.962e-01  -0.202 0.839968
## salesTRUE                         -3.266e-01  1.513e-01  -2.158 0.030905
## affiliatedTRUE                    -3.046e-01  1.944e-01  -1.567 0.117150
## websiteTRUE                       -5.778e-01  1.387e-01  -4.167 3.09e-05
## support_agentAndrea Becci          1.599e+00  6.855e-01   2.333 0.019635
## support_agentDaniel                1.416e+00  5.172e-01   2.738 0.006178
## support_agentDaniel Bae            1.225e+00  6.295e-01   1.946 0.051681
## support_agentElias                 1.715e+00  5.344e-01   3.209 0.001332
## support_agentFabrizio Baldi        1.913e+00  7.423e-01   2.576 0.009983
## support_agentJonny Fenton          2.540e+00  6.534e-01   3.887 0.000102
## support_agentKara Kruzeniski       1.418e+00  4.172e-01   3.399 0.000677
## support_agentKevin Ho              1.546e+00  8.359e-01   1.849 0.064398
## support_agentLeroy Alexander       1.374e-01  5.529e-01   0.249 0.803729
## support_agentLucy Litvanyi         7.219e-01  5.571e-01   1.296 0.195021
## support_agentPablo Lamothe         6.395e-01  4.888e-01   1.308 0.190773
## support_agentPhilip Cho            3.094e-01  4.763e-01   0.650 0.515924
## cancel_contact_supportYes          5.293e+00  5.859e-01   9.034  < 2e-16
## 
## (Intercept)
## business_industryart_design
## business_industryb2b_sales
## business_industrybeauty_wellness
## business_industryblogger
## business_industryconsultant
## business_industrye_commerce
## business_industryeducation
## business_industryfitness_health
## business_industryfood_restaurants
## business_industryhotel_resort     *
## business_industrynon_profit
## business_industryreal_estate
## business_industrysaas
## business_industrytravel
## total_leads
## interestLanding Page              ***
## interestPop-Up                    ***
## sales_agentAlex Baranovsky
## sales_agentDan Novak
## sales_agentInga Zane
## sales_agentJason Bradbury
## sales_agentJordan Gutierrez
## sales_agentKemal Wahju
## sales_agentKevin Ho
## sales_agentlaura
## sales_agentMarlyn
## sales_agentMoe Tajsekandar
## sales_agentRose Chua
## sales_agentRoselyn Bonito
## sales_agentSam Moghaddam
## sales_agentSara McKenzie
## salesTRUE                          *
## affiliatedTRUE
## websiteTRUE                        ***
## support_agentAndrea Becci          *
## support_agentDaniel                **
## support_agentDaniel Bae            .
## support_agentElias                 **
## support_agentFabrizio Baldi        **
## support_agentJonny Fenton          ***
## support_agentKara Kruzeniski       ***
## support_agentKevin Ho              .
## support_agentLeroy Alexander
## support_agentLucy Litvanyi
## support_agentPablo Lamothe
## support_agentPhilip Cho
## cancel_contact_supportYes          ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2867.4  on 2376  degrees of freedom
## Residual deviance: 2157.2  on 2329  degrees of freedom
## AIC: 2253.2
##
## Number of Fisher Scoring iterations: 8
```

From the function above we can observe some important correlations, they are:
- There is a high correlation that customers that have their own website, tend to stay subscribed with Wishpond.
- If a customer subscribe after coming from a demo with one of the sales agent there is a significant correlation of them staying subscribed
- There is a high correlation of remaining subscribed if a customer main interest to subscribe to Wishpond is to build a Landing Page or Pop-Up
- There is a high correlation that unsubscribed customers have talked to a support agent before unsubscribing.

Now, we will use the predict test to see how many customer are in risk of unsubscribing and then we will evaluate the accuracy of model by using the table function and a threshold of 50%, since we have no preference of errors in the database.

```
predicTest <- predict(wp_data_Log, type="response", newdata = test)
# Accuracy
table(test$Unsubscribed, predicTest > 0.5)
##
##        FALSE TRUE
##   FALSE   144  229
##   TRUE     79  829
# Accuracy of 74.9%
(149+811)/(811+149+224+97)
## [1] 0.7494145
# Baseline Method - 70.8%. Therefore, our model beats the baseline method by 4%, showing some sigificance.
(97+811)/(811+149+224+97)
## [1] 0.7088212
```

To analyze if the model can differentiate from high-risk to low-risk to unsubscribe customers, we will use prediction function from ROCpred library.

```
ROCRpred <- prediction(predicTest, test$Unsubscribed)
as.numeric(performance(ROCRpred, "auc")@y.values)
## [1] 0.8019171
```

Our model shows that our accuracy beats our baseline method by only 2% and the ROCR shows that the model can differentiate with 62% from the subscribed and unsubscribed customers.

Therefore, in order to see if we can increase the relevance of our model, we will compare logistic regression with random forest regression.

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
wp_data_forest <- randomForest(Unsubscribed ~ business_industry + total_leads + interest +
sales_agent + sales + affiliated + website + support_agent + cancel_contact_support, data = train,
nodesize=25, ntree=200)
PredictForest = predict(wp_data_forest, newdata = test)
table(test$Unsubscribed, PredictForest)
##        PredictForest
##         FALSE TRUE
##   FALSE   163  210
##   TRUE     99  809
# Accuracy from Random Forest Model is 76.1%, which improves our predictability by just over
1% when comparing to the logistic regression.
(155+820)/(820+155+218+88)
## [1] 0.7611241
# Our model beat baseline by 6%, which means our model did better than the baseline method.
(88+820)/(820+155+218+88)
## [1] 0.7088212
```

By using the function above, we can see that Random Forest regression improves our model by 1% when comparing to logistic regression, to a total of 73% accuracy.

Recommendation

From the machine learning analysis we were able to see that there is a high rate of customers unsubscribing from the platform, and if nothing changes 70% of the customers will continue to unsubscribe from the platform.

However, from our model we observed that there independent variables with significant correlation that encourages clients to remain subscribed. Moreover, based on the findings we will provide four recommendations to Wishpoond.

First, increase focus on those who are interested in building Landing Pages and Pop-ups, as both shows high correlation with remaining subscribed to Wishpond. The increase in focus can be done by focusing marketing efforts such as ads and blogs towards people that are interest in landing pages and pop-ups.

Second, similar to the recommendation above, marketing efforts should also focus on businesses that have own have own a domain or just purchased a domain. To reach this audience the marketing team could create content on how Wishpond can integrate with their website and paid ads to those that are engaged with domain related platforms.

Third, marketing efforts should not only be focuse on acquiring new leads, but should also focus getting the lead to book a demo with a sales agent. This can be done via email marketing campaign to get them to book a demo and automated SMS to remind leads about their demo.

their demo.

Last but nont least, there were a lot of missing data, and the accuracy results could be improved if data from unsubscribed customers were automatically set to 0. Therefore, we recommend saving the data right at the moment the customer decide to unsubscribe from the platform, giving us more insights on what they did and the reason the might have cancelled.