Optimizing Conversion Rate

Mitul Shah

03/07/2025

Goal

Optimizing conversion rate is likely the most common work of a data scientist, and rightfully so. The data revolution has a lot to do with the fact that now we are able to collect all sorts of data about people who buy something on our site as well as people who don't. This gives us a tremendous opportunity to understand what's working well (and potentially scale it even further) and what's not working well (and fix it).

The goal of this challenge is to build a model that predicts conversion rate and, based on the model, come up with ideas to improve it.

Challenge Description

We have data about all users who hit our site: whether they converted or not as well as some of their characteristics such as their country, the marketing channel, their age, whether they are repeat users and the number of pages visited during that session (as a proxy for site activity/time spent on site).

Our task is to predict conversion rate and come up with recommendations for the product team and the marketing team to improve conversion rate.

Load the data into R

```
## Load the data
data = read.csv("/Users/mitulshah/Downloads/Downloads/Giulio Palombo Book
Solutions/Optimizing Conversion Rate/conversion_project.csv", header = TRUE)

## Look at the first few rows of the data
knitr::kable(head(data))
```

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

country	age	new_user	source	total_pages_visited	converted
UK	25	1	Ads	1	0
US	23	1	Seo	5	0
US	28	1	Seo	4	0

country	age	new_user	source	total_pages_visited	converted
China	39	1	Seo	5	0
US	30	1	Seo	6	0
US	31	0	Seo	1	0

country : user country based on the IP address

age : user age. Self-reported at sign-up step

new_user : whether the user created the account during this session or had already an account and simply came back to the site

source : marketing channel source Ads: came to the site by clicking on an advertisement Seo: came to the site by clicking on search results Direct: came to the site by directly typing the URL on the browser

total_pages_visited: number of total pages visited during the session. This can be seen as a proxy for time spent on site and engagement

converted: this is our label. 1 means they converted within the session, 0 means they left without buying anything. The company goal is to increase conversion rate: # conversions / total sessions

Inspecting Data Quality

Firstly, let's inspect the data to look for weird behavior/wrong data. Data is never perfect in real life and requires to be cleaned. Identifying wrong data and dealing with it is a crucial step.

```
## Look at the summary of the data
knitr::kable(summary(data))
```

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

country	age	new_user	source	total_pages_visited	converted
Length:316200	Min. : 17.00	Min. :0.0000	Length:316200	Min. : 1.000	Min. :0.00000
Class :character	1st Qu.: 24.00	1st Qu.:0.0000	Class :character	1st Qu.: 2.000	1st Qu.:0.00000
Mode :character	Median :	Median :1.0000	Mode :character	Median : 4.000	Median :0.00000

country	age	new_user	source	total_pages_visited	converted
	30.00				
NA	Mean : 30.57	Mean :0.6855	NA	Mean : 4.873	Mean :0.03226
NA	3rd Qu.: 36.00	3rd Qu.:1.0000	NA	3rd Qu.: 7.000	3rd Qu.:0.00000
NA	Max. :123.00	Max. :1.0000	NA	Max. :29.000	Max. :1.00000

```r
## Convert country, new user and source to factor variable
data$country = as.factor(data$country)
data$new_user = as.factor(data$new_user)
data$source = as.factor(data$source)
```

A few quick observations:

1. The site is probably a US site, although it does have a large Chinese user base as well
2. User base is pretty young
3. Conversion rate at around 3% is industry standard. It makes sense
4. Everything seems to make sense here except for max age 123 yrs! Let's investigate it:

```r
## Sort all ages in descending order
sort(unique(data$age), decreasing=TRUE)
```

```
##  [1] 123 111  79  77  73  72  70  69  68  67  66  65  64  63  62  61  60
59  58
## [20]  57  56  55  54  53  52  51  50  49  48  47  46  45  44  43  42  41
40  39
## [39]  38  37  36  35  34  33  32  31  30  29  28  27  26  25  24  23  22
21  20
## [58]  19  18  17
```

Those 123 and 111 values seem unrealistic. How many users are we talking about:

```r
## Number of users with unrealistic age
knitr::kable(subset(data, age>79))
```

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

	country	age	new_user	source	total_pages_visited	converted
90929	Germany	123	0	Seo	15	1
295582	UK	111	0	Ads	10	1

It is just 2 users! In this case, we can remove them, nothing will change.

In general, depending on the problem, you can:

1. Remove the entire row saying you don't trust the data
2. Treat them as NAs
3. If there is a pattern, try to figure out what went wrong.

That being said, wrong data is worrisome and can be an indicator of some bug in the logging code. Therefore, when working, you will want to talk to the software engineer who implemented the logging code to see if, perhaps, there are some bugs which affect the data significantly.

Here, let's just get rid of those two rows:

```
## Remove unrealistic ages
data = subset(data, age<80)
```

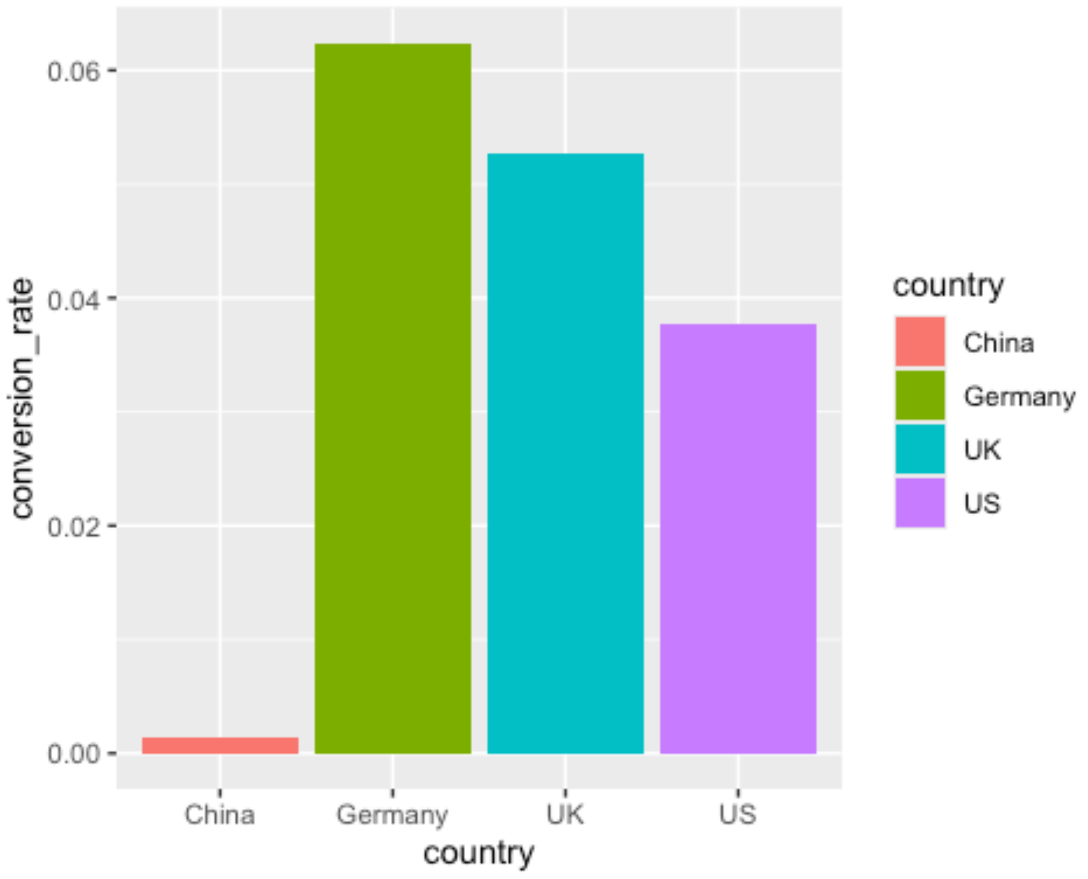
Exploratory Data Analysis

Now, let's quickly investigate the variables and how their distribution differs for the two classes. This will help us understand whether there is any information in our data in the first place and get a sense of the data.

Let's look at the conversion rate for each of the variables.

```
## Load the required libraries
require(dplyr)

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

require(ggplot2)

## Loading required package: ggplot2
```

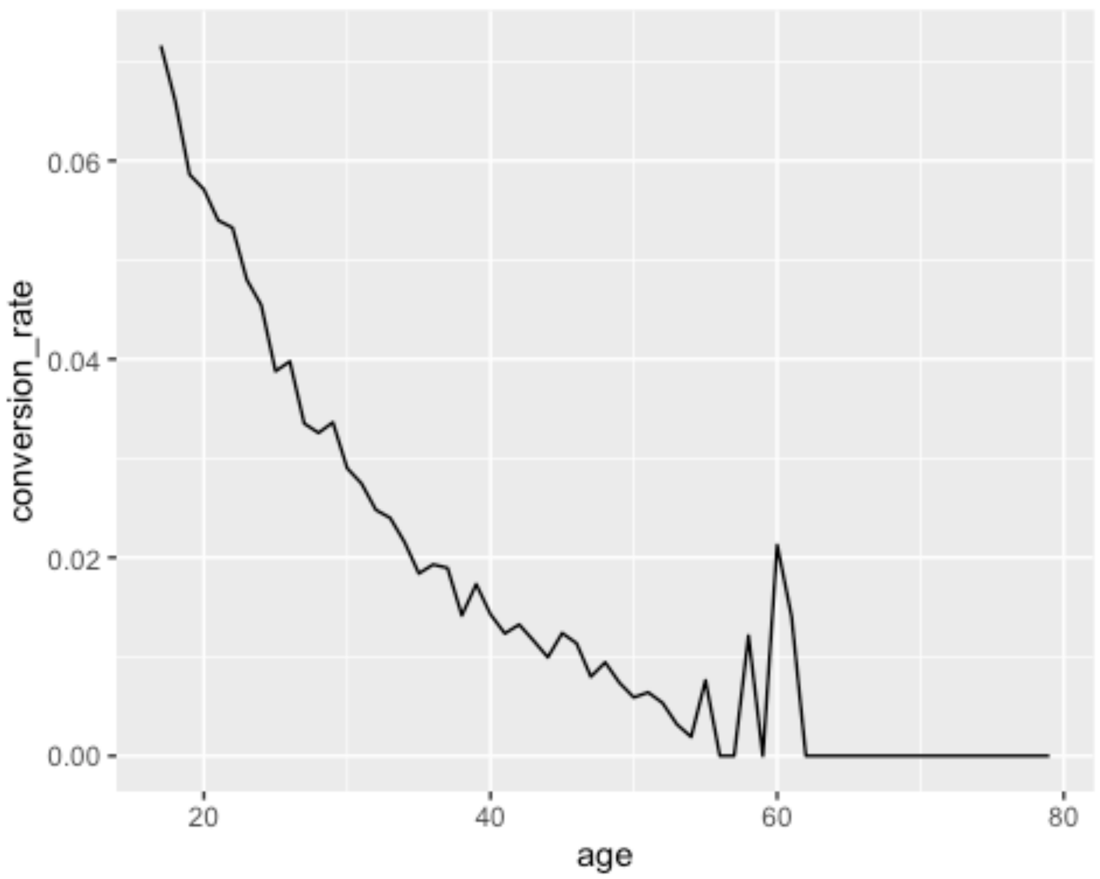
```r
## Create data country dataset showing conversion rate by country
data_country = data %>%
                group_by(country) %>%
                summarise(conversion_rate = mean(converted))

## Plot the conversion rate by country
ggplot(data=data_country, aes(x=country, y=conversion_rate))+
       geom_bar(stat = "identity", aes(fill = country))
```



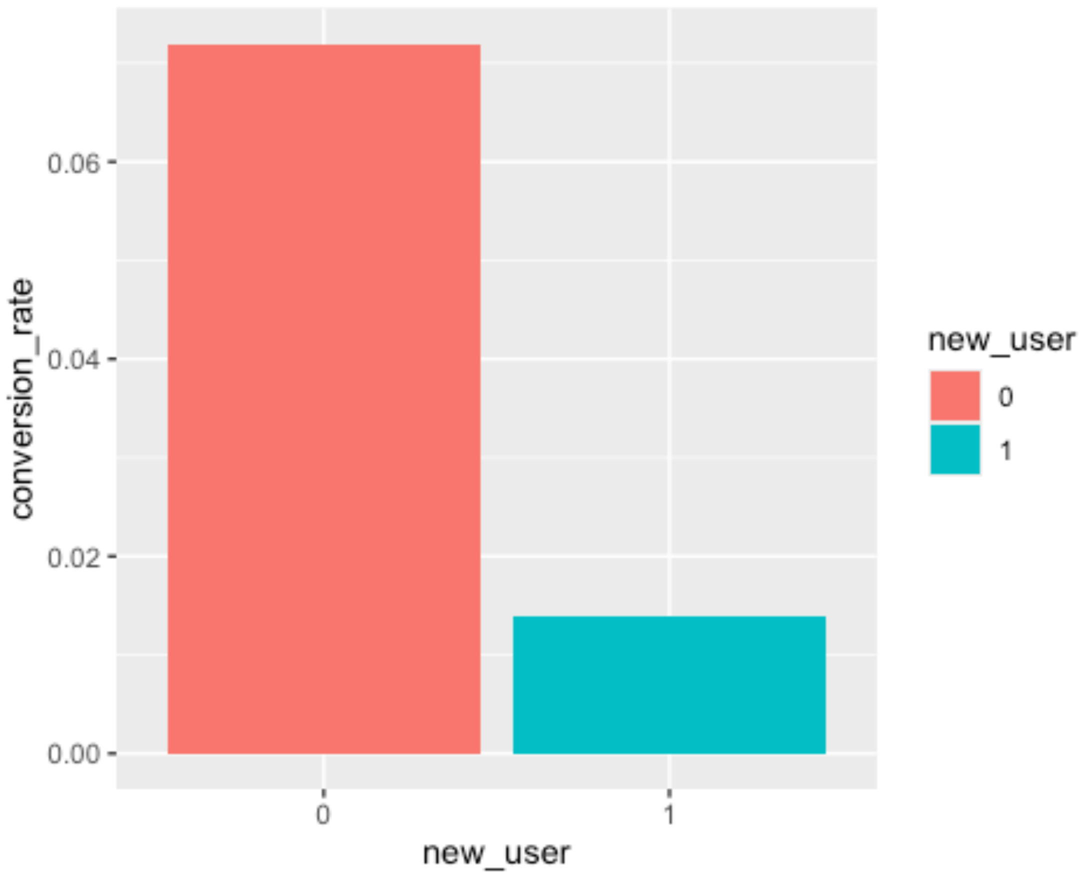
Here it clearly looks like Chinese convert at a much lower rate than other countries!

```r
## Create data age dataset showing conversion rate by age
data_age = data %>%
                group_by(age) %>%
                summarise(conversion_rate = mean(converted))

## Plot the conversion rate by pages
qplot(age, conversion_rate, data=data_age, geom="line")
```

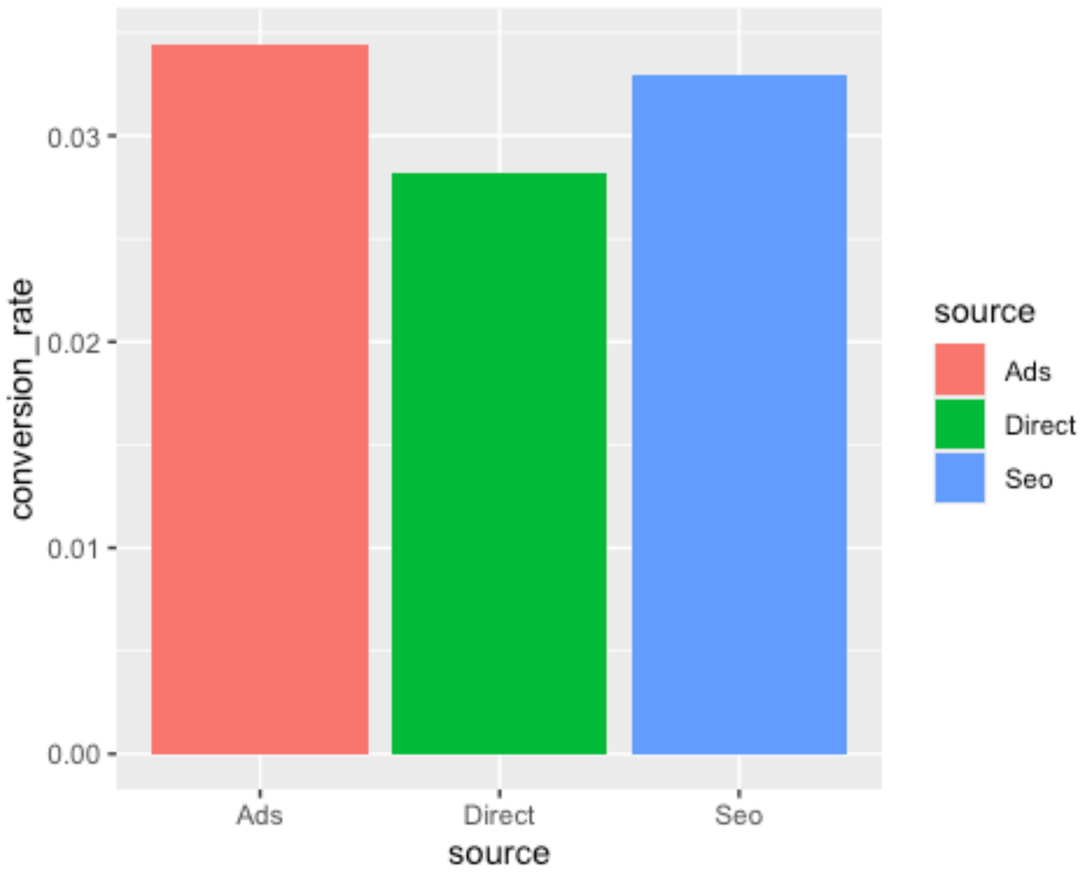
```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



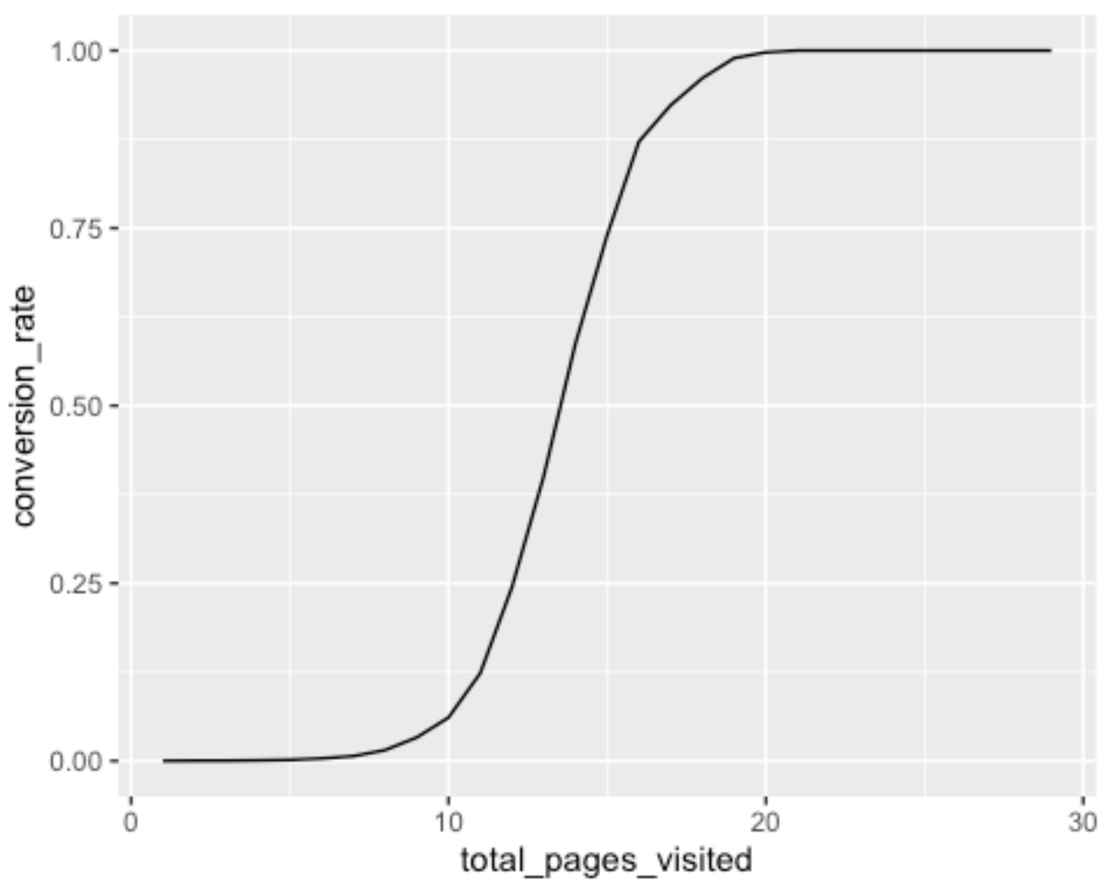
```r
## Create data new user dataset showing conversion rate by new user
data_new_user = data %>%
                group_by(new_user) %>%
                summarise(conversion_rate = mean(converted))

## Plot the conversion rate by country
ggplot(data=data_new_user, aes(x=new_user, y=conversion_rate))+
       geom_bar(stat = "identity", aes(fill = new_user))
```



```r
## Create data country dataset showing conversion rate by country
data_source = data %>%
                group_by(source) %>%
                summarise(conversion_rate = mean(converted))

## Plot the conversion rate by country
ggplot(data=data_source, aes(x=source, y=conversion_rate))+
       geom_bar(stat = "identity", aes(fill = source))
```



```r
## Create data pages dataset showing conversion rate by pages
data_pages = data %>%
                group_by(total_pages_visited) %>%
                summarise(conversion_rate = mean(converted))

## Plot the conversion rate by pages
qplot(total_pages_visited, conversion_rate, data=data_pages, geom="line")
```



Machine Learning

Let's now build a model to predict conversion rate. Outcome is binary and you care about insights to give product and marketing team project ideas.

I am going to pick a random forest to predict conversion rate. I picked a random forest because: it usually requires very little time to optimize it (its default params are often close to be the best ones) and it is strong with outliers, irrelevant variables, continuous and discrete variables. I will use the random forest to predict conversion, then I will use its partial dependence plots and variable importance to get insights. Also, I will build a simple tree to find the most obvious user segments.

Firstly, "converted" should really be a factor here. So let's change it:

```r
## Convert converted to factor variable
data$converted = as.factor(data$converted)

## Shorter name for Germany, better for the plots
levels(data$country)[levels(data$country)=="Germany"]="DE"
```

Create test/training set with a standard 66% split (if the data were too small, I would cross-validate). Then, I build the forest with standard values for the 3 important parameters (100 trees, trees as large as possible, 3 random variables selected at each split).

```
## Load the library randomforest
require(randomForest)

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.2

## 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

## Set the seed
set.seed(4321)

## Create train/test split with 66% data as training data
train_sample = sample(nrow(data), size = nrow(data)*0.66)

## Training data
train_data = data[train_sample,]

## Test data
test_data = data[-train_sample,]

## Random Forest Model
rf = randomForest(y=train_data$converted, x = train_data[, -
ncol(train_data)],
                  ytest = test_data$converted, xtest = test_data[, -
ncol(test_data)],
                  ntree = 100, mtry = 3, keep.forest = TRUE)

## Print the Random Forest Model
rf

##
## Call:
##  randomForest(x = train_data[, -ncol(train_data)], y =
train_data$converted,      xtest = test_data[, -ncol(test_data)], ytest =
test_data$converted,      ntree = 100, mtry = 3, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 1.44%
## Confusion matrix:
##        0    1 class.error
## 0 201075  882 0.004367266
## 1   2124 4609 0.315461161
##                 Test set error rate: 1.48%
## Confusion matrix:
##        0    1 class.error
## 0 103607  436 0.004190575
## 1   1152 2313 0.332467532
```

So, OOB error and test error are pretty similar around 1.4%. We are confident we are not overfitting.

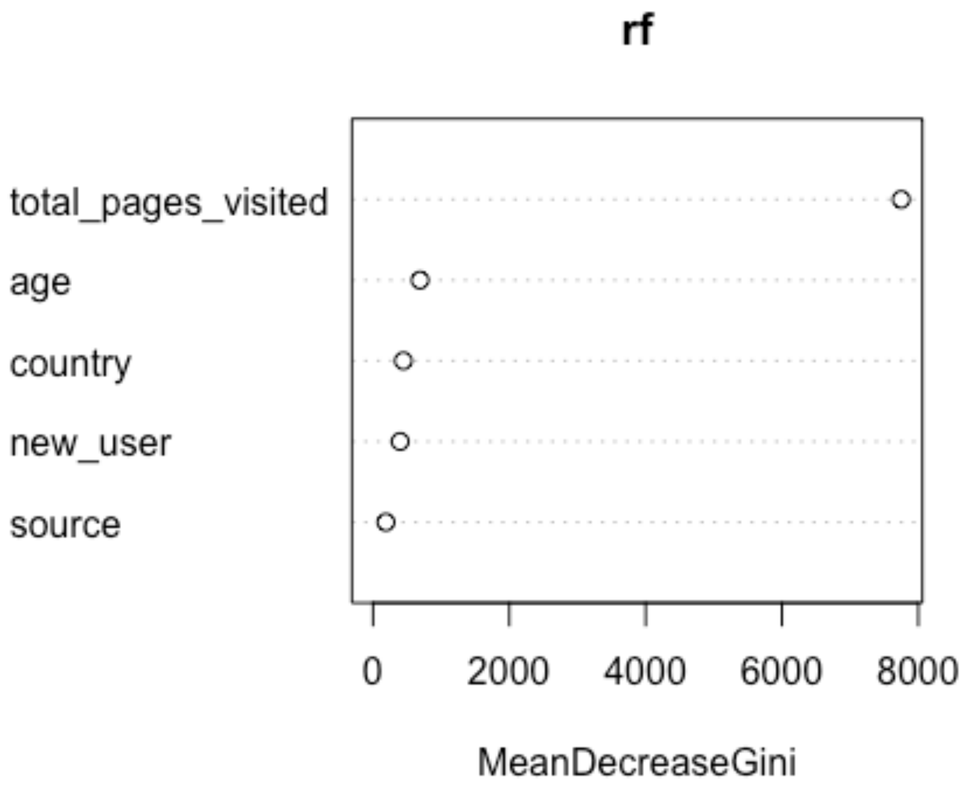
Error is pretty low. However, we started from a 97% accuracy (that's the case if we classified everything as a "non converted"). So, ~98.6% is good, but nothing shocking. Indeed, 30% of conversions are predicted as "non conversion".

If we cared about the very best possible accuracy or specifically minimizing false positive/false negative, we would find the best cut-off point. Since in this case that doesn't appear to be particularly relevant, we are fine with the default 0.5 cutoff value used internally by the random forest to make the prediction.

If you care about insights, building a model is just the first step. You need to check that the model predicts well and, if it does, you can now extract insights out of it.

Let's start by checking variable importance:

```
## Variable Importance Plot
varImpPlot(rf,type=2)
```



Total pages visited is the most important one, by far. Unfortunately, it is probably the least "actionable". People visit many pages because they already want to buy. Also, in order to buy, you have to click on multiple pages. Let's rebuild the RF without that variable. Since classes are heavily unbalanced and we don't have that very powerful variable anymore, let's change the weights a bit, just to make sure we will get something classified as 1.

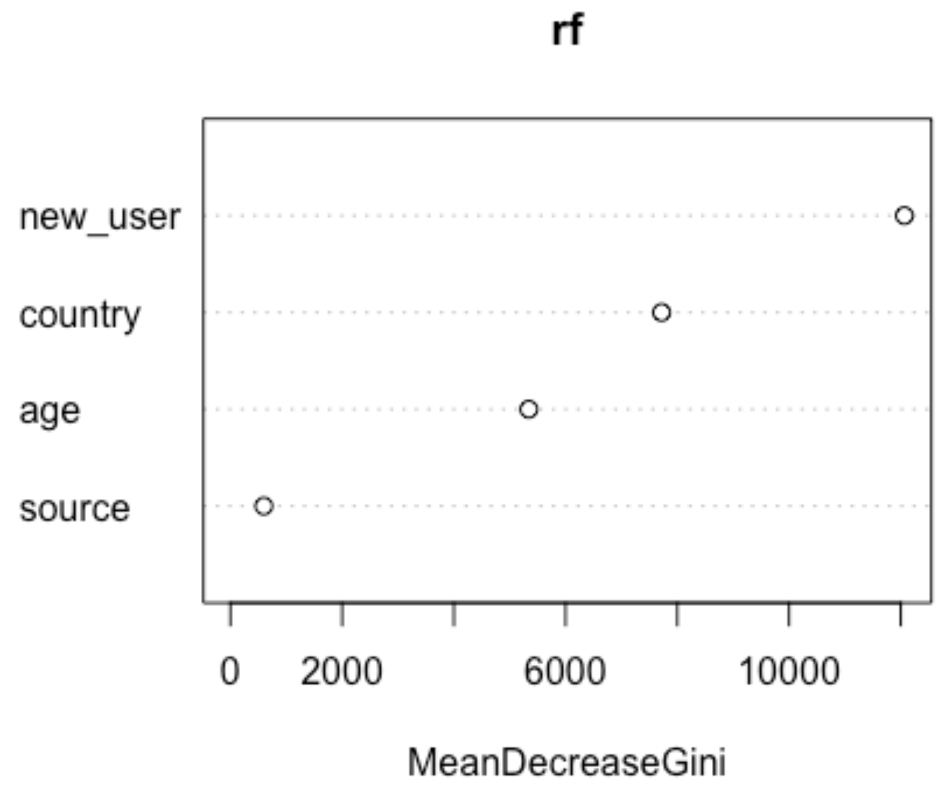
```
## Random Forest Model without pages visited variable
rf = randomForest(y=train_data$converted, x = train_data[, -c(5,
ncol(train_data))],
          ytest = test_data$converted, xtest = test_data[, -c(5,
ncol(train_data))],
                  ntree = 100, mtry = 3, keep.forest = TRUE, classwt =
c(0.7,0.3))

## Print the output of Random Forest Model
rf

##
## Call:
##  randomForest(x = train_data[, -c(5, ncol(train_data))], y =
train_data$converted,      xtest = test_data[, -c(5, ncol(train_data))],
ytest = test_data$converted,      ntree = 100, mtry = 3, classwt = c(0.7,
0.3), keep.forest = TRUE)
```

```
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.87%
## Confusion matrix:
##        0     1 class.error
## 0 176143 25814   0.1278193
## 1   3129  3604   0.4647260
##                 Test set error rate: 13.64%
## Confusion matrix:
##       0     1 class.error
## 0 90991 13052   0.1254481
## 1  1610  1855   0.4646465
```

Accuracy went down, but that's fine. The model is still good enough to give us insights.

Let's recheck variable importance:

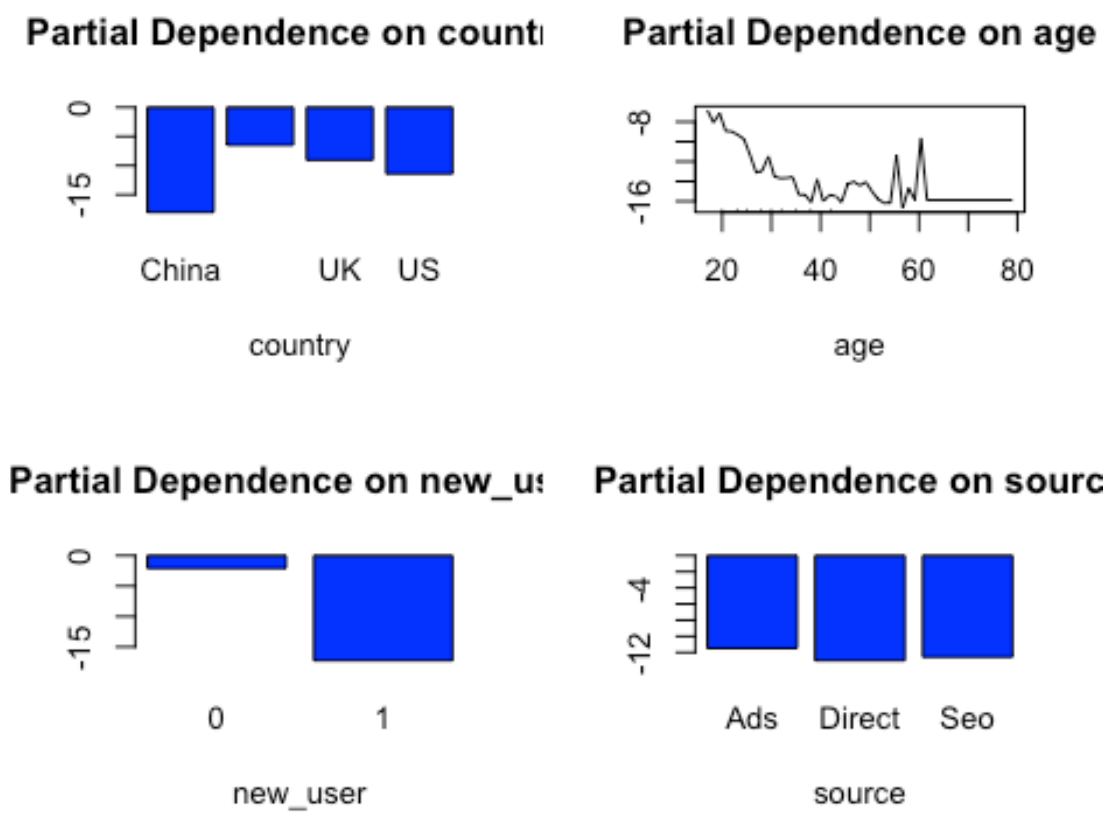
```
## Variable Importance Plot
varImpPlot(rf,type=2)
```



Interesting! New user is the most important one. Source doesn't seem to matter at all.

Let’s check partial dependence plots for the 4 vars:

```
## Partial Dependence Plots
op <- par(mfrow=c(2, 2))
partialPlot(rf, train_data, country, 1)
partialPlot(rf, train_data, age, 1)
partialPlot(rf, train_data, new_user, 1)
partialPlot(rf, train_data, source, 1)
```



```
par(op)
```

This shows that:

1. Users with an old account are much better than new users
2. China is really bad, all other countries are similar with Germany being the best
3. The site works very well for young people and gets worse for >30 yr old
4. Source is basically irrelevant

Let’s now build a simple decision tree and check the 2 or 3 most important segments:

```
## Load the rpart library
require(rpart)
```

```
## Loading required package: rpart

## Build the decision tree
tree = rpart(data$converted ~ ., data[, -c(5,ncol(data))],
             control = rpart.control(maxdepth = 3),
             parms = list(prior = c(0.7, 0.3))
             )

## Print the tree
tree

## n= 316198
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 316198 94859.4000 0 (0.70000000 0.30000000)
##    2) new_user=1 216744 28268.0600 0 (0.84540048 0.15459952) *
##    3) new_user=0 99454 66591.3400 0 (0.50063101 0.49936899)
##      6) country=China 23094   613.9165 0 (0.96445336 0.03554664) *
##      7) country=DE,UK,US 76360 50102.8100 1 (0.43162227 0.56837773)
##       14) age>=29.5 38341 19589.5200 0 (0.57227507 0.42772493) *
##       15) age< 29.5 38019 23893.0000 1 (0.33996429 0.66003571) *
```

A simple small tree confirms exactly the random forest findings.

Conclusion and next step

1. The site is working very well for young users. Definitely let's tell marketing to advertise and use channels which are more likely to reach young people.

2. The site is working very well for Germany in terms of conversion. But the summary showed that there are few Germans coming to the site: way less than UK, despite a larger population. Again, marketing should get more Germans. Big opportunity.

3. Users with old accounts do much better. Targeted emails with offers to bring them back to the site could be a good idea to try.

4. Maybe go through the UI and figure out why older users perform so poorly? From ~30 y/o conversion clearly starts dropping. A good actionable metric here is conversion rate for people >=30 yr old. Building a team whose goal is to increase that number would be interesting.

5. Something is wrong with the Chinese version of the site. It is either poorly translated, doesn't fit the local culture, or maybe some payment issue. Given how many users are based in China, fixing this should be a top priority. Huge opportunity.