# Conversion Rate

## 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 revenue.

This challenge is significantly easier than all others in this collection. There are no dates, no tables to join, no feature engineering required, and the problem is really straightforward. Therefore, it is a great starting point to get familiar with data science take-home challenges.

**You should not move to the other challenges until you fully understand this one**.

---

## Challenge Description

We have data about 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).

Your project is to:

- Predict conversion rate
- Come up with recommendations for the product team and the marketing team to improve conversion rate

## Data

We have 1 table downloadable by clicking **here**.

```
The table is "conversion_data". It has information about signed-in users
during one session. Each row is a user session.
```

**Columns:**

- **country** : user country based on the IP address
- **age** : user age. Self-reported at sign-in 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 is a proxy for time spent on site and engagement during the session.
- **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.

# Example

```
Let's now check the characteristics of the user in the first row.
```

**head(conversion_data, 1)**

| Field | Value | Description |
|---|---|---|
| country | UK | the user is based in the UK |
| age | 25 | the user is 25 yr old |
| new_user | 1 | she created her account during this session |
| source | Ads | she came to the site by clicking on an ad |
| total_pages_visited | 1 | she visited just 1 page during that session |
| converted | 0 | this user did not buy during this session. These are the users whose behavior we want to change! |

# Solution: Conversion Rate

```
#libraries needed
require(dplyr)
require(rpart)
require(ggplot2)
require(randomForest)
```

Let's read the dataset into R.

```
data = read.csv('conversion_data.csv')
```

We should get something like this:

```
head(data)
```

```
##   country age new_user source total_pages_visited converted
## 1      UK  25        1    Ads                   1         0
## 2      US  23        1    Seo                   5         0
## 3      US  28        1    Seo                   4         0
## 4   China  39        1    Seo                   5         0
## 5      US  30        1    Seo                   6         0
## 6      US  31        0    Seo                   1         0
```

Let's check the structure of the data:

```
str(data)
```

```
## 'data.frame':    316200 obs. of  6 variables:
##  $ country            : Factor w/ 4 levels "China","Germany",..: 3 4 4 1 4 4 1
4 3 4 ...
##  $ age                : int  25 23 28 39 30 31 27 23 29 25 ...
##  $ new_user           : int  1 1 1 1 1 0 1 0 0 0 ...
##  $ source             : Factor w/ 3 levels "Ads","Direct",..: 1 3 3 3 3 3 3 1 2
1 ...
##  $ total_pages_visited: int  1 5 4 5 6 1 4 4 4 2 ...
##  $ converted          : int  0 0 0 0 0 0 0 0 0 0 ...
```

Now, let's inspect the data to look for weird behavior/wrong data. Data is never perfect in real life and requires to be cleaned. Often takehome challenges have wrong data which has been put there on purpose. **Identifying the wrong data and dealing with it is part of the challenge.**

R summary function is usually the best place to start:

```
summary(data)
```

```
##     country            age            new_user         source
##  China  : 76602   Min.   : 17.00   Min.   :0.0000   Ads    : 88740
##  Germany: 13056   1st Qu.: 24.00   1st Qu.:0.0000   Direct: 72420
##  UK     : 48450   Median : 30.00   Median :1.0000   Seo    :155040
##  US     :178092   Mean   : 30.57   Mean   :0.6855
##                   3rd Qu.: 36.00   3rd Qu.:1.0000
##                   Max.   :123.00   Max.   :1.0000
##  total_pages_visited   converted
##  Min.   : 1.000       Min.   :0.00000
##  1st Qu.: 2.000       1st Qu.:0.00000
##  Median : 4.000       Median :0.00000
##  Mean   : 4.873       Mean   :0.03226
##  3rd Qu.: 7.000       3rd Qu.:0.00000
##  Max.   :29.000       Max.   :1.00000
```

A few quick observations:

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

```
sort(unique(data$age), decreasing=TRUE)
```

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

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

```
subset(data, age>79)
```

```
##        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:

- remove the entire row saying you don't trust the data
- treat those values as NAs
- if there is a pattern, try to figure out what went wrong.

In doubt, always go with removing the row. It is the safest choice.

You probably also want to emphasize in the text that wrong data is worrisome and can be an indicator of some bug in the logging code. Therefore, you'd like to talk to the software engineer who implemented the code to see if, perhaps, there are some bugs which affect the data significantly.

Anyway, here is probably just users who put wrong data. So let's remove them:

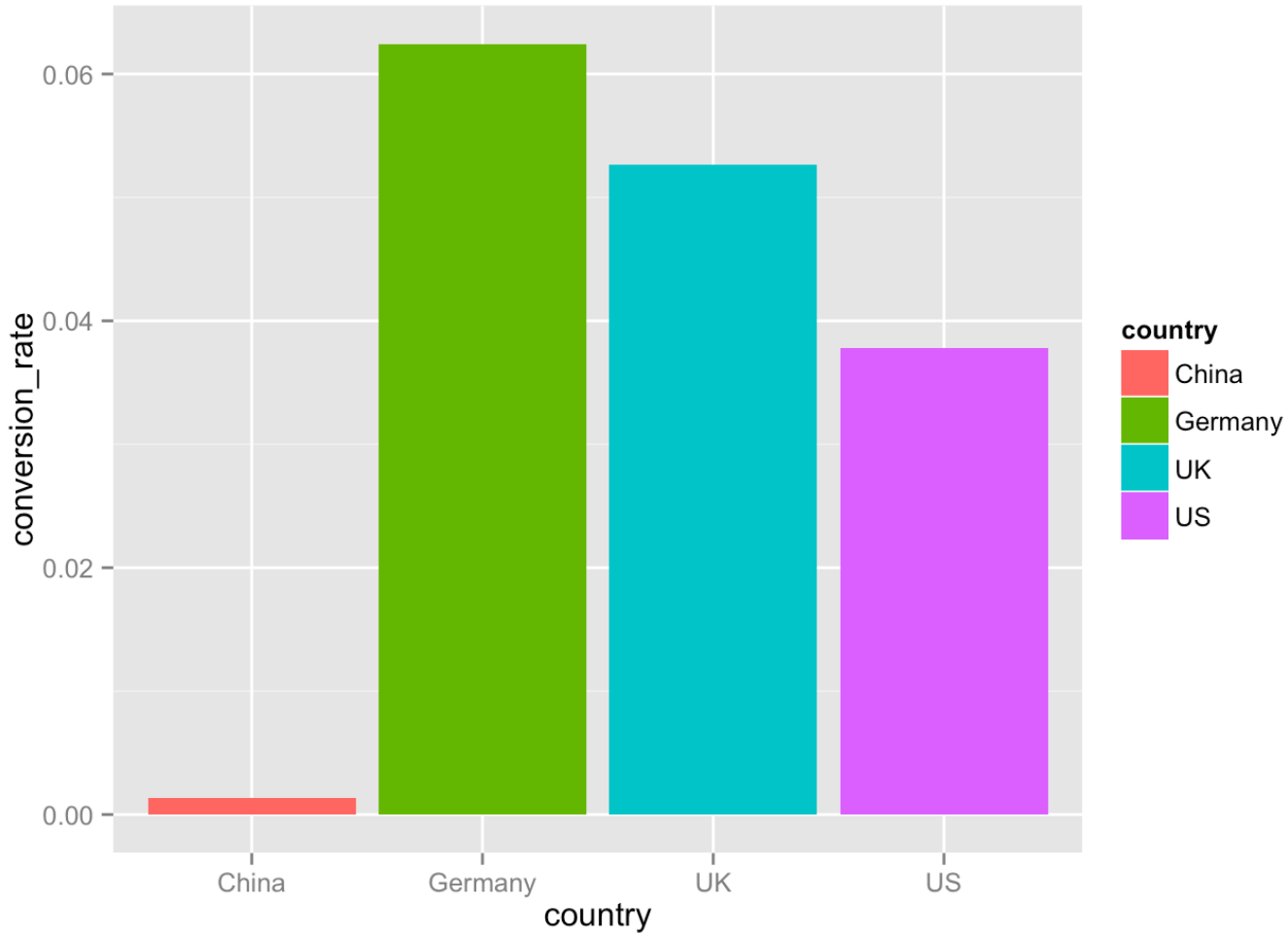
```
data = subset(data, age<80)
```

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.

**Never start by blindly building a machine learning model. Always first get a sense of the data.**

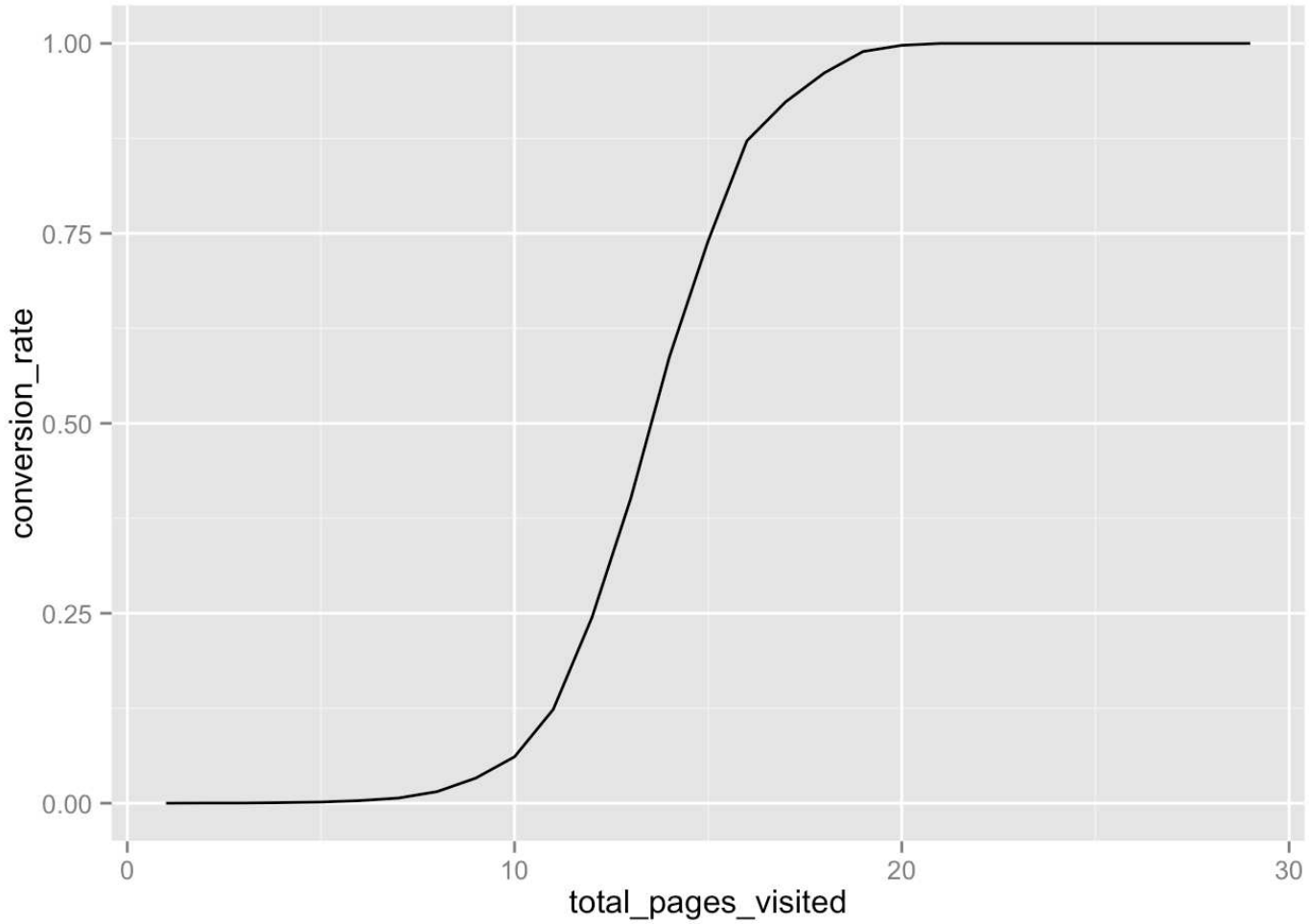
Let's just pick a couple of vars as an example, but you should do it with all:

```
data_country = data %>%
                    group_by(country) %>%
                    summarise(conversion_rate = mean(converted))

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!

```
 data_pages = data %>%
                group_by(total_pages_visited) %>%
                summarise(conversion_rate = mean(converted))
qplot(total_pages_visited, conversion_rate, data=data_pages, geom="line")
```




Definitely spending more time on the site implies higher probability of conversion!

**Focus on your strengths in the challenge.** If visualization is your main strength, spend as much time as you wish on that and come up with something great. You might be hired as a great data scientist - visualization. If you have other strengths, spend more time on those. Take-home challenges are pretty open ended by design. By seeing where you spend more time, hiring managers can also understand your strengths, what you like doing the most, and where you would fit best.

# 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 some ideas. You should probably choose among the following models:

- Logistic regression
- Decision Trees
- RuleFit (this is often your best choice)

- Random Forest in combination with partial dependence plots

Pick the one you know the best. Don't spend too much time optimizing it. Just explain why you picked it and say that with more time you would spend ~1 day trying other models/different params and you would pick the best.

Ex: I am going to pick a random forest to predict conversion rate. I pick a random forest cause: it usually requires very little time to optimize it (its default params are often close to 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 about how it got information from the variables. Also, I will build a simple tree to find the most obvious user segments and see if they agree with RF partial dependence plots.

Firstly, "Converted" should really be a factor here as well as new_user. So let's change them:

```
data$converted = as.factor(data$converted) # let's make the class a factor
data$new_user = as.factor(data$new_user) #also this a factor
levels(data$country)[levels(data$country)=="Germany"]="DE" # Shorter name, easier
to plot.
```

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

```
train_sample = sample(nrow(data), size = nrow(data)*0.66)
train_data = data[train_sample,]
test_data = data[-train_sample,]

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)

rf
```

```
##
## Call:
##  randomForest(x = train_data[, -ncol(train_data)], y = train_data$converted,
xtest = test_data[, -ncol(test_data)], ytest = test_data$converted,      ntree = 1
00, 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: 1.47%
## Confusion matrix:
##        0    1 class.error
## 0 201008  935 0.004630019
## 1   2133 4614 0.316140507
##                 Test set error rate: 1.41%
## Confusion matrix:
##        0    1 class.error
## 0 103589  468 0.004497535
## 1   1045 2406 0.302810779
```

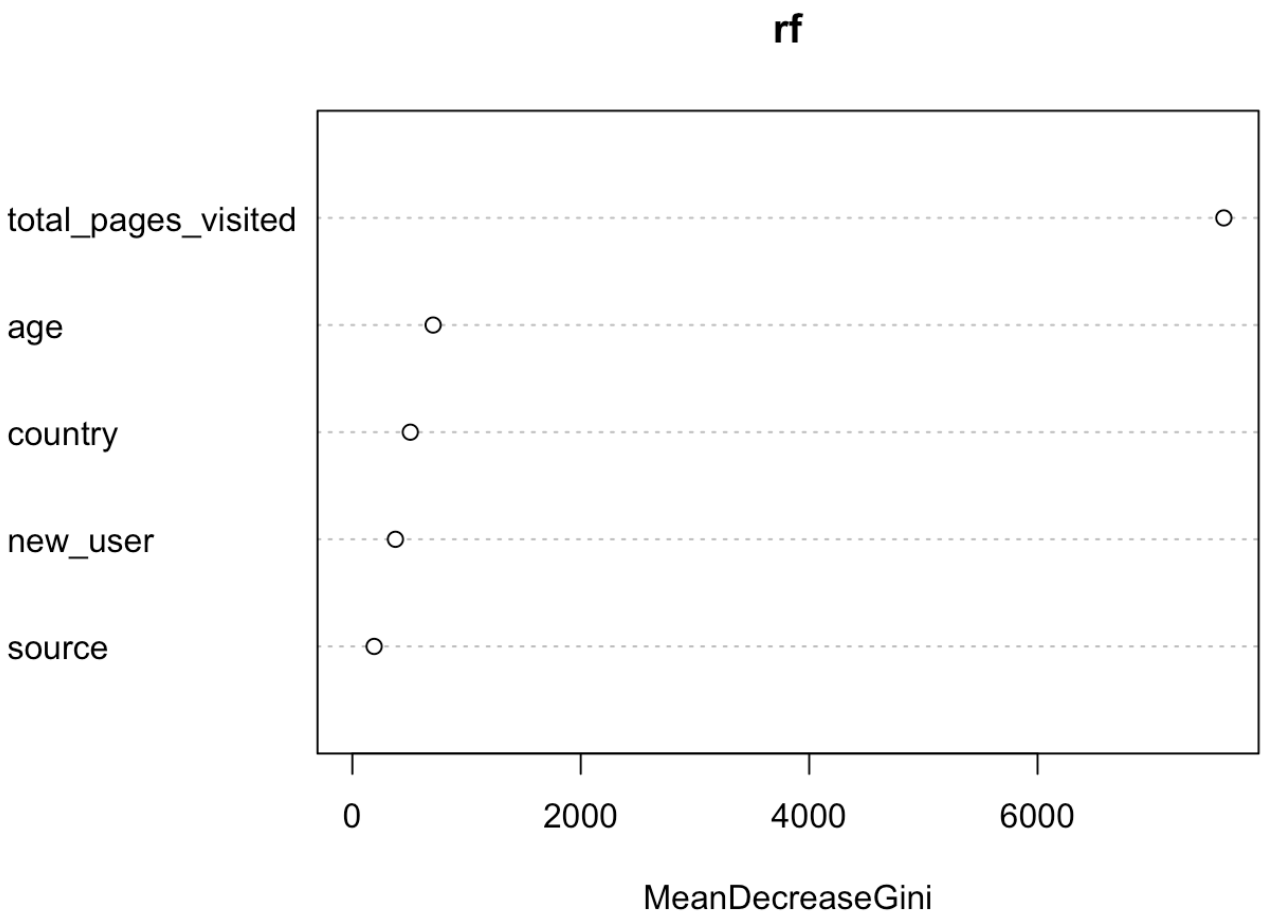
So, OOB error and test error are pretty similar: 1.5% and 1.4%. We are confident we are not overfitting. Error is pretty low. However, we started from a 97% accuracy (that's the case if we classified everything as "non converted"). So, 98.5% is good, but nothing shocking. Indeed, 30% of conversions are predicted as "non conversion".

If we cared about the very best possible accuracy or specifically minimizing false positive/false negative, we would also use ROCR and 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. Again, if ROC and cut-off analysis is something you know very well, you should do it no matter what.

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 checking variable importance:

```
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 cause 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 weight a bit, just to make sure we will get something classified as 1.

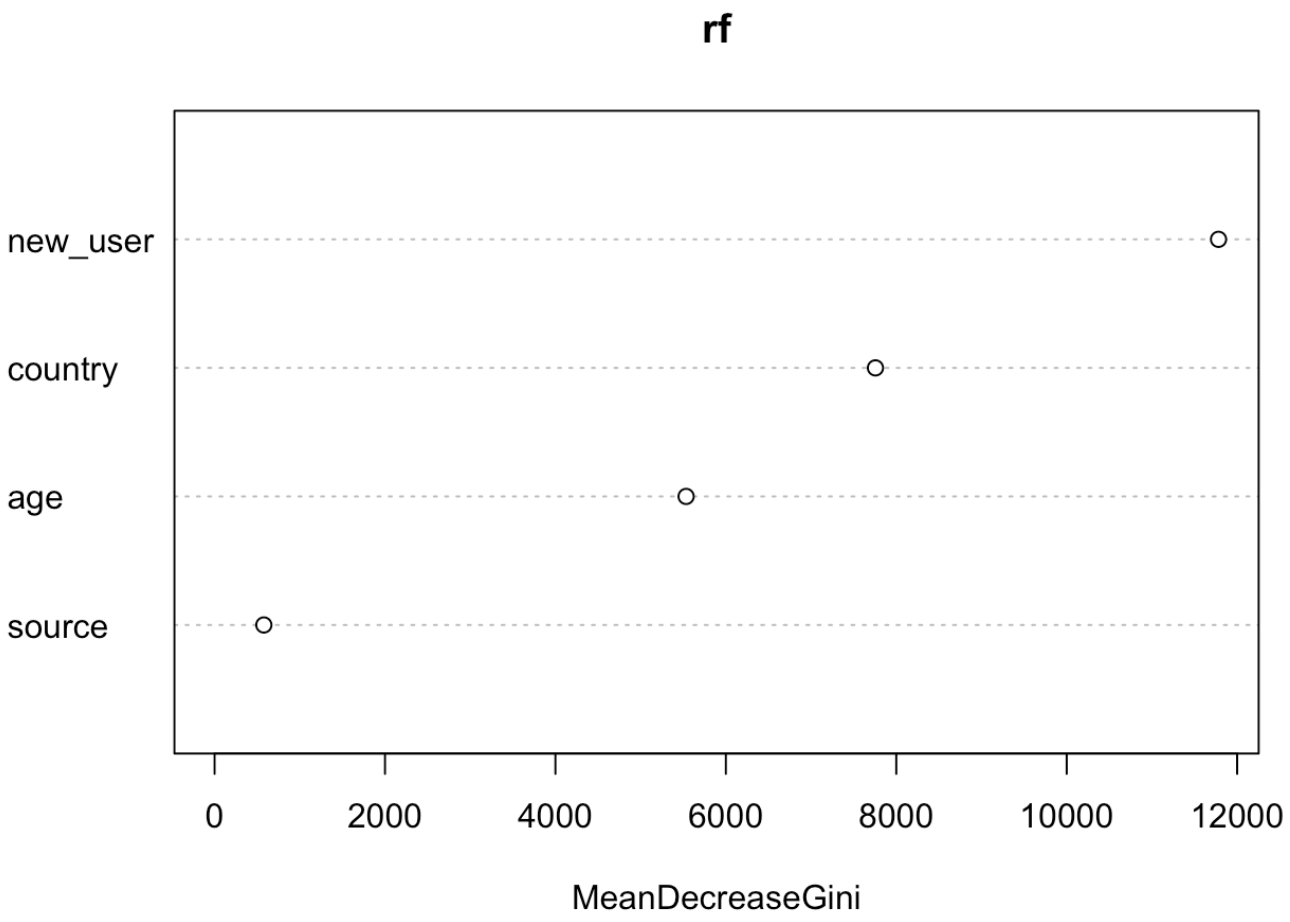
```
rf = randomForest(y=train_data$converted, x = train_data[, -c(5, ncol(train_dat
a))],
          ytest = test_data$converted, xtest = test_data[, -c(5, ncol(train_dat
a))],
                  ntree = 100, mtry = 3, keep.forest = TRUE, classwt = c(0.7,0.3))

rf
```

```
##
## Call:
##  randomForest(x = train_data[, -c(5, ncol(train_data))], y = train_data$convert
ed,      xtest = test_data[, -c(5, ncol(train_data))], ytest = test_data$converte
d,      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: 14.09%
## Confusion matrix:
##        0     1 class.error
## 0 175596 26347   0.1304675
## 1   3051  3696   0.4522010
##                 Test set error rate: 13.91%
## Confusion matrix:
##       0     1 class.error
## 0 90703 13354   0.1283335
## 1  1601  1850   0.4639235
```

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

Let's recheck variable importance:

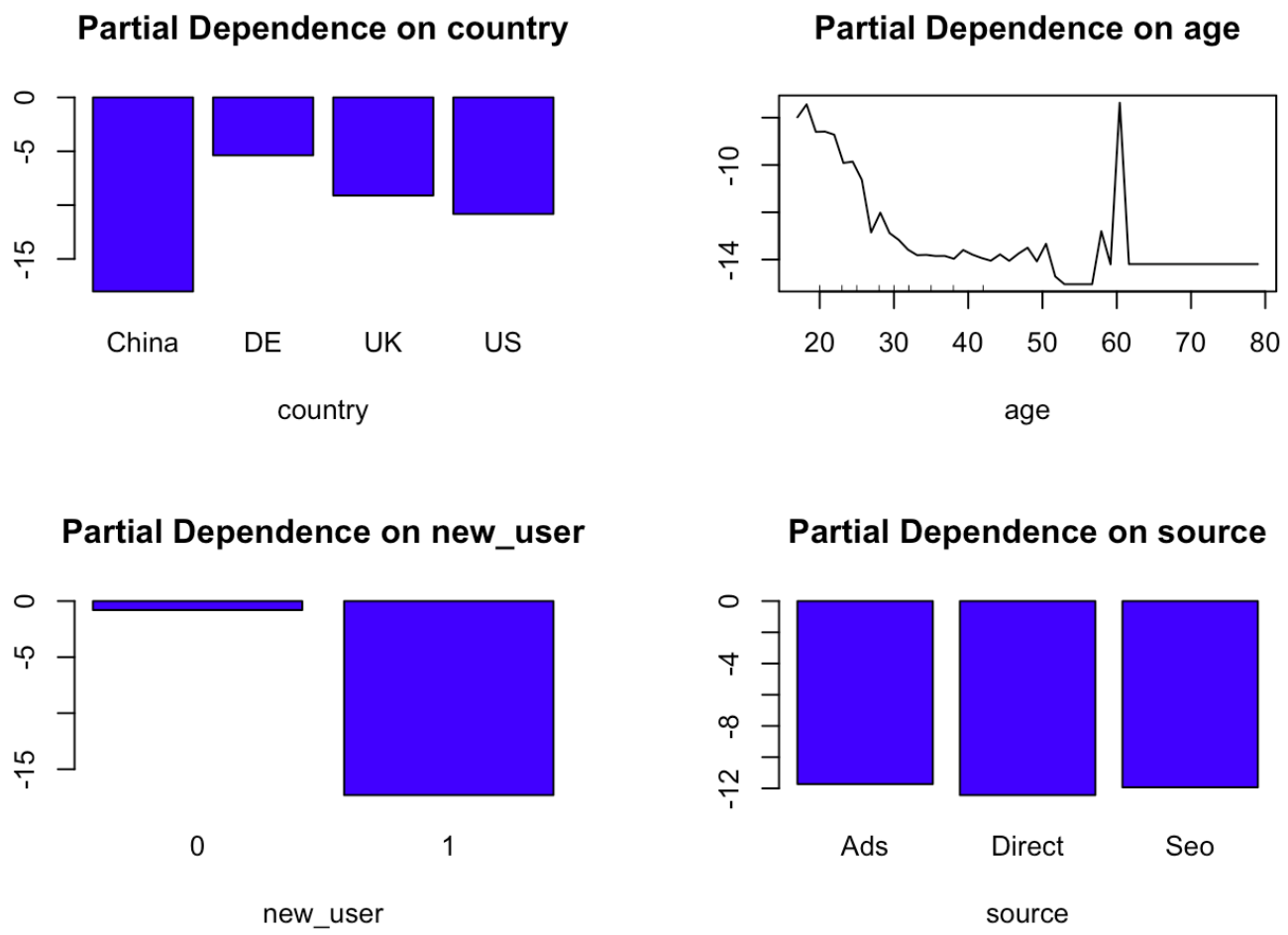
```
varImpPlot(rf,type=2)
```




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:

```
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)
```




In partial dependence plots, we just care about the trend, not the actual y value. So this shows that:

- Users with an old account are much better than new users
- China is really bad, all other countries are similar with Germany being the best
- The site works very well for young people and bad for less young people (>30 yrs old)
- Source is irrelevant

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

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

Some conclusions and suggestions:

1. The site is working very well for young users. Definitely let's tell marketing to advertise and use marketing channel 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. Something is wrong with the Chinese version of the site. It is either poorly translated, doesn't fit the local culture, some payment issue or maybe it is just in English! Given how many users are based in China, fixing this should be a top priority. Huge opportunity.
5. Maybe go through the UI and figure out why older users perform so poorly? From 30 y/o conversion clearly starts dropping.
6. If I know someone has visited many pages, but hasn't converted, she almost surely has high purchase intent. I could email her targeted offers or sending her reminders. Overall, these are probably the easiest users to make convert.

As you can see, conclusions usually end up being about:

1. tell marketing to get more of the good performing user segments
2. tell product to fix the experience for the bad performing ones