Previous Lecture Complete and continue

INSIGHTS: Conversion Rate with solution

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 with ideas to improve it.

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 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).

Your project is to:

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

Data

- R
- **Python**

Let's read the dataset

```
#read from google drive
data=read.csv("https://drive

head(data)
```

```
  country age new_user sourc
1      UK  25        1    Ac
2      US  23        1    Se
3      US  28        1    Se
4   China  39        1    Se
5      US  30        1    Se
6      US  31        0    Se
```

- country : user country based on the IP address 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 conversions / total sessions

Let's read the dataset

```
import pandas
pandas.set_option('display.m
pandas.set_option('display.w

#read from google drive
data=pandas.read_csv("https:

print(data.head())
```

```
  country  age  new_user sou
0      UK   25         1
1      US   23         1
2      US   28         1
3   China   39         1
4      US   30         1
```

- 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

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

Descriptive Stats

- R
- **Python**

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**

R summary function is usually the best place to start:

```
summary(data)
```

```
    country            age
 China   : 76602   Min.   : 1
 Germany: 13056   1st Qu.: 2
 UK      : 48450   Median : 3
 US      :178092   Mean   : 3
                   3rd Qu.: 3
                   Max.   :12
```

A few quick observations:

- the site is probably a US site, although it does have a large

Previous Lecture Complete and continue

- 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), decre
```

```
 [1] 123 111  79  77  73  72
[37]  40  39  38  37  36  35
```

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

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

```
       country age new_user
90929  Germany 123        0
295582      UK 111        0
```

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 them as NAs
- 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:

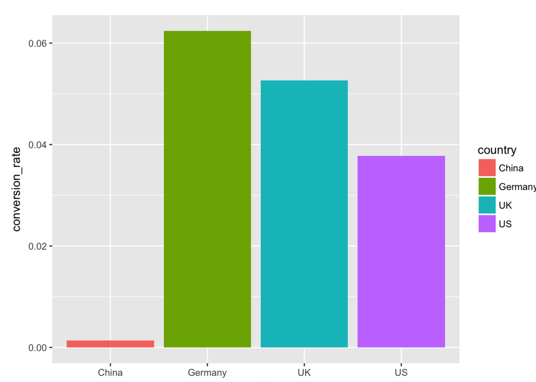
```
data = subset(data, age<80)
```

Now, let's quickly investigate the variables and how their distribution differs for the two 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

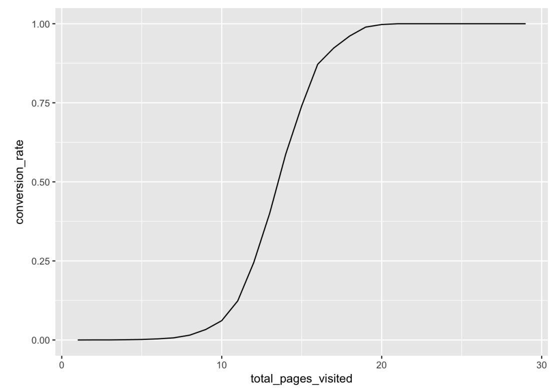
Let's just pick a couple of variables as example, but you should do it with all:

```
require(dplyr)
require(ggplot2)

data_country = data %>%
                      group_by(cou
                      summarise(cc
ggplot(data=data_country, ae
         geom_bar(stat = "iden
```



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

```
                  summarise(cc
qplot(total_pages_visited, c
```



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 the wrong data and dealing with it is a crucial step**

Describe and groupby are usually the best places to start:

```
print(data.describe())
```

```
                 age         r
count  316200.000000  31620(
mean       30.569858        (
std         8.271802        (
min        17.000000        (
25%        24.000000        (
50%        30.000000        1
```

```
print(data.groupby(['country
```

```
country
China       76602
Germany     13056
UK          48450
US         178092
dtype: int64
```

```
print(data.groupby(['source'
```

```
source
Ads        88740
Direct     72420
Seo       155040
dtype: int64
```

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:

```
print(sorted(data['age'].uni
```

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

```
print(data[data['age']>110])

        country  age  new_us
90928   Germany  123
295581       UK  111
```

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 them as NAs
- 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 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:

```
data = data[data['age']<110]
```

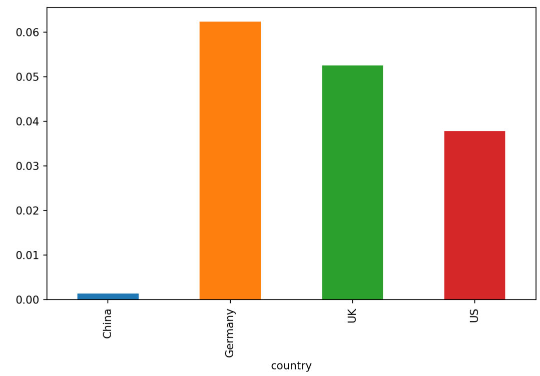
Now, let's quickly investigate the variables and how their distribution differs for the two classes. This will help us understand whether there is any information in our data in the first place and get a sense of the data.

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

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

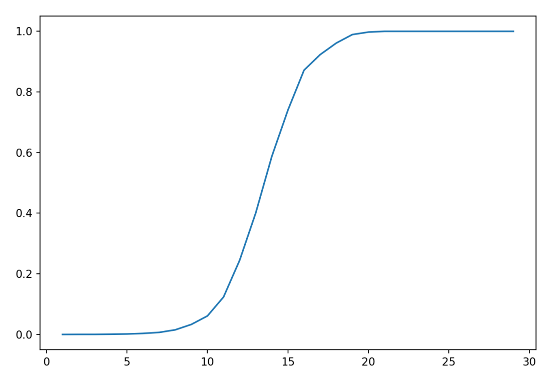
Previous Lecture Complete and continue


```
rcParams.update({'figure.aut

data.groupby(['country'])['c
plt.show()
```



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

```
data.groupby(['total_pages_v
plt.show()
```



Definitely spending more time on

of conversion?

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. You should probably choose among the following options:

- Logistic regression
- Decision Trees
- RuleFit
- Random Forest or Boosted Decision Trees in combination with partial dependence plots

It is good to add two lines to explain why you chose a given approach.

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.

- R
- **Python**

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

```
# let's make the label and 
data$converted = as.factor(c
data$new_user = as.factor(da
# Shorter name for Germany, 
levels(data$country)[levels(
```

Previous Lecture Complete and continue

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).

```
require(randomForest)
set.seed(4321)

train_sample = sample(nrow(c
train_data = data[train_samp
test_data = data[-train_samp
rf = randomForest(y=train_da
                  ytest = te
                  ntree = 1(

rf
```

```
Call:
 randomForest(x = train_data
              Type of rando
                    Number
No. of variables tried at ea

        OOB estimate of  err
Confusion matrix:
       0    1 class.error
0 201080  856 0.004238967
1   2176 4578 0.322179449
                Test set err
Confusion matrix:
       0    1 class.error
0 103629  435  0.00418012
1   1105 2339  0.32084785
```

pretty similar around 1.4%. We are confident we are not overfitting.

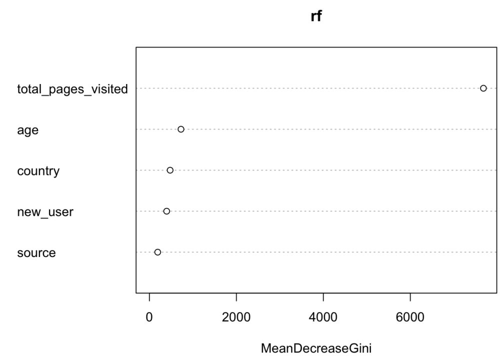
Error is pretty low. However, we started from a 97% accuracy (that's the case if we classified everything as a "non converted"). So, ~98.6% is good, but nothing shocking. Indeed, 30% of conversions are predicted as "non conversion".

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:

```
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.

```
rf = randomForest(y=train_da
         ytest = test_data$
                 ntree = 1(

rf
```

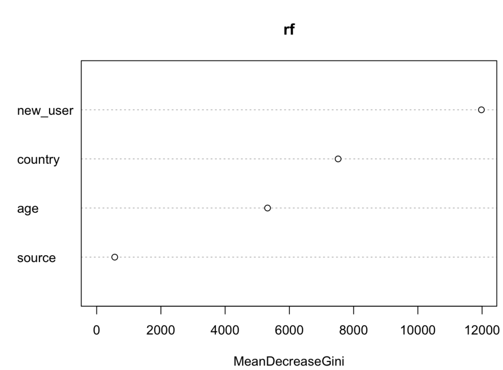
Previous Lecture Complete and continue

```
 randomForest(x = train_data
              Type of rando
                  Number
No. of variables tried at ea

        OOB estimate of  err
Confusion matrix:
       0     1 class.error
0 176171 25765   0.1275899
1   3134  3620   0.4640213
                 Test set err
Confusion matrix:
      0     1 class.error
0 90858 13206   0.1269027
1  1600  1844   0.4645761
```

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

Let's recheck variable importance:

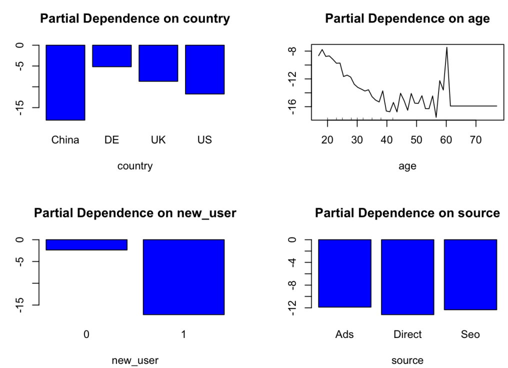
```
varImpPlot(rf,type=2)
```



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

plots for the 4 vars:

```
op <- par(mfrow=c(2, 2))
partialPlot(rf, train_data,
partialPlot(rf, train_data,
partialPlot(rf, train_data,
partialPlot(rf, train_data,


par(op)
```



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 gets worse for >30 yr old

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

```
require(rpart)

tree = rpart(data$converted
             control = rpart
             parms = list(pr
             )
tree
```

```
n= 316198

node), split, n, loss, yval,
      * denotes terminal noc

 1) root 316198 94859.4000 (
   2) new_user=1 216744 2826
   3) new_user=0 99454 66591
     6) country=China 23094
     7) country=DE,UK,US 763
      14) age>=29.5 38341 19
      15) age< 29.5 38019 23
```

A simple small tree confirms exactly the random forest findings.

Firstly, let's create dummy variables from the categorical ones:

```
#dummy variables for the cat
data_dummy = pandas.get_dumm
```

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).

```
import numpy as np
from sklearn.ensemble import

/usr/local/opt/python/Framew
  from numpy.core.umath_test

from sklearn.metrics import
from sklearn.model_selectior
np.random.seed(4684)

#split into train and test t
train, test = train_test_spl

#build the model
```

Previous Lecture
Complete and continue


```
#let's print OOB accuracy ar
print(
"OOB accuracy is",
rf.oob_score_,
"\n",
"OOB Confusion Matrix",
"\n",
pandas.DataFrame(confusion_m
)
```

```
OOB accuracy is 0.983885188
 OOB Confusion Matrix
          0     1
0  200872  1102
1    2261  4455
```

```
#and let's print test accura
print(
"Test accuracy is", rf.score
"\n",
"Test Set Confusion Matrix",
"\n",
pandas.DataFrame(confusion_m
)
```

```
Test accuracy is 0.98473601
 Test Set Confusion Matrix
          0     1
0  103483   543
1    1098  2384
```

So, OOB error and test error are pretty similar, ~1.5%. We are confident we are not overfitting.

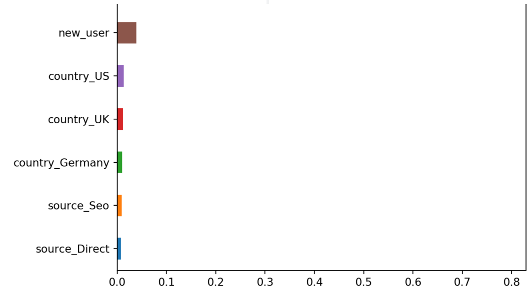
Error is pretty low. However, we started from a 97% accuracy (that's the case if we classified everything as a "non converted"). So, 98.5% is good, but nothing conversions are predicted as "non conversion".

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:

```
feat_importances = pandas.Se
feat_importances.sort_values
plt.show()
```



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, just to make sure we will get something classified as 1.

```
#build the model without tot
rf = RandomForestClassifier(
rf.fit(train.drop(['converte

#let's print OOB accuracy ar
print(
"OOB accuracy is",
rf.oob_score_,
"\n",
"OOB Confusion Matrix",
"\n",
```

```
OOB accuracy is 0.889827016l
 OOB Confusion Matrix
         0       1
0   182720   19254
1     3738    2978
```

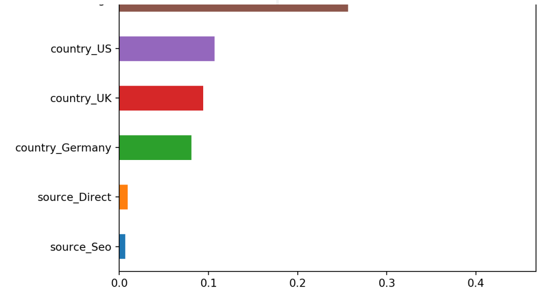
```
#and let's print test accura
print(
"Test accuracy is", rf.score
"\n",
"Test Set Confusion Matrix",
"\n",
pandas.DataFrame(confusion_m
)
```

```
Test accuracy is 0.88998028(
 Test Set Confusion Matrix
        0      1
0   94140   9886
1    1942   1540
```

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

Let’s recheck variable importance:

```
feat_importances = pandas.Se
feat_importances.sort_values
plt.show()
```

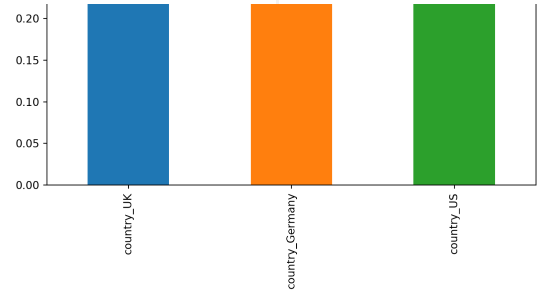


Interesting! New user is the most important one, even more important than age. And that's impressive given that continuous variables tend to always show up at the top in RF variable importance plots. It means new_user is really important. Source-related dummies don't seem to matter at all.

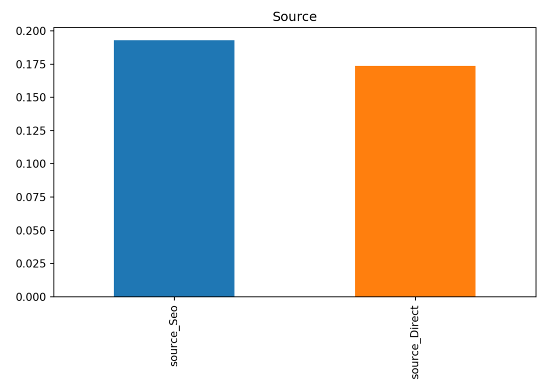
Let's check partial dependence plots for the 4 vars:

```
from pdpbox import pdp, info

#country
pdp_iso = pdp.pdp_isolate( m
                          da
                          mo
                          fe
                          nu
pdp_dataset = pandas.Series(
pdp_dataset.sort_values(asce
plt.show()
```

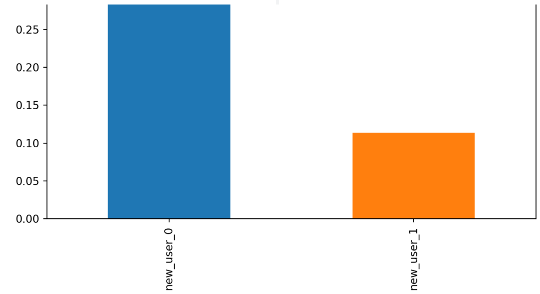
Previous Lecture Complete and continue



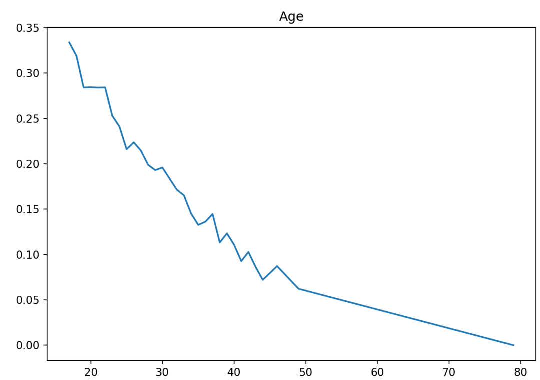

```
#source
pdp_iso = pdp.pdp_isolate( m
                          da
                          mo
                          fe
                          nu
pdp_dataset = pandas.Series(
pdp_dataset.sort_values(asce
plt.show()
```




```
#new user
pdp_iso = pdp.pdp_isolate( m
                          da
                          mo
                          fe
                          nu
pdp_dataset = pandas.Series(
pdp_dataset.sort_values(asce
plt.show()
```




```
#age
pdp_iso = pdp.pdp_isolate( m
                          da
                          mo
                          fe
                          nu

pdp_dataset = pandas.Series(
pdp_dataset.plot(title='Age'
plt.show()
```



This shows that:

- Users with an old account are much better than new users
- Germany, UK, and US are similar, with Germany being the best. Most importantly,

values. We could read this as relative to the reference level, which is China. So this means that not being from China and being from any of those 3 countries significantly increases the probability of conversion. That is, China is very bad for conversion

- The site works very well for young people and gets worse for >30 yr old
- Source is less relevant

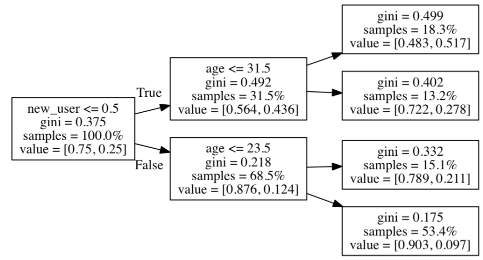
Let's now build a simple decision tree and check the 2 or 3 most important segments:

```
import graphviz
from sklearn.tree import Dec
from sklearn.tree import exp
from graphviz import Source

tree = DecisionTreeClassifie
tree.fit(train.drop(['conver

#visualize it
export_graphviz(tree, out_fi
with open("tree_conversion.c
    dot_graph = f.read()
```

Previous Lecture
Complete and continue

12% COMPLETE


```
s.view()
```



A simple small tree confirms exactly the random forest findings.

Conclusions and next steps:

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.

Previous Lecture Complete and continue

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

Previous Lecture Complete and continue

should be a top priority. Huge opportunity.

As you can see, product ideas usually end up being about:

- Identify segments that perform well, but have low absolute count (like Germany). Then tell marketing to get more of those people
- Tell product to fix the experience for the bad performing ones
- Bad performing segments with high absolute count (like China) usually provide the biggest opportunities for massive gains, if you can guess why that's happening and then build a test to validate your hypothesis