Identifying Fraudulent Activities

Mitul Shah

05/06/2019

Goal

E-commerce websites often transact huge amounts of money. And whenever a huge amount of money is moved, there is a high risk of users performing fraudulent activities, e.g. using stolen credit cards or doing money laundry.

Machine Learning really excels at identifying fraudulent activities. Any website where you put your credit card information has a risk team in charge of avoiding frauds via machine learning.

The goal of this challenge is to build a machine learning model that predicts the probability that the first transaction of a new user is fraudulent.

Challenge Description

Company XYZ is an e-commerce site that sells hand-made clothes.

You have to build a model that predicts whether a user has a high probability of using the site to perform some illegal activity or not. This is a super common task for data scientists.

You only have information about the user first transaction on the site and based on that you have to make your classification ("fraud/no fraud").

These are the tasks you are asked to do:

1. For each user, determine their country based on the IP address
2. Build a model to predict whether an activity is fraudulent or not. Explain how different assumptions about the cost of false positives vs false negatives would impact the model
3. Your boss is a bit worried about using a model she doesn't understand for something as important as fraud detection. How would you explain her how the model is making the predictions? Not from a mathematical perspective (she couldn't care less about that), but from a user perspective. What kinds of users are more likely to be classified as at risk? What are their characteristics?
4. Let's say you now have this model which can be used live to predict in real time if an activity is fraudulent or not. From a product perspective, how would you use it? That is, what kind of different user experiences would you build based on the model output?

Data

Let's read the datasets and see the variables we have

```
## Run this every new R session. It converts scientific notation within
results into decimals for this R session
options(scipen = 9999)

## Load 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(magrittr)

## Loading required package: magrittr

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:dplyr':
##
##     combine

require(ROCR)

## Loading required package: ROCR

## Read the data
data=read.csv("/Users/mitulshah/Downloads/Downloads/Giulio Palombo Book
Solutions/Identifying Fraudulent Activities/Fraud_data.csv")
```

```
## Look at the head 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")
```

user_id	signup_time	purchase_time	purchase_value	device_id	source	browser	sex	age	ip_address	class
22058	2015-02-24 22:55:49	2015-04-18 02:47:11	34	QVPSPJUOCKZAR	SEO	Chrome	M	39	732758369	0
333320	2015-06-07 20:39:50	2015-06-08 01:38:54	16	EOGFQPIZPYXFZ	Ads	Chrome	F	53	350311388	0
1359	2015-01-01 18:52:44	2015-01-01 18:52:45	15	YSSKYOSJHPPLJ	SEO	Opera	M	53	2621473820	1
150084	2015-04-28 21:13:25	2015-05-04 13:54:50	44	ATGTXKYKUDUQN	SEO	Safari	M	41	3840542444	0
221365	2015-07-21 07:09:52	2015-09-09 18:40:53	39	NAUITBZFJKHWW	Ads	Safari	M	45	415583117	0
159135	2015-05-21 06:03:03	2015-07-09 08:05:14	42	ALEYXFXINSXLZ	Ads	Chrome	M	18	2809315200	0

user_id : Id of the user. Unique by user

signup_time : the time when the user created her account (GMT time)

purchase_time : the time when the user bought the item (GMT time)

purchase_value : the cost of the item purchased (USD)

device_id : the device id. You can assume that it is unique by device. I.e., same device ID means that the same physical device was used for the transaction

source : user marketing channel: ads, SEO, Direct (i.e. came to the site by directly typing the site address on the browser)

browser : the browser used by the user

sex : user sex: Male/Female

age : user age

ip_address : user numeric ip address

class : this is what we are trying to predict: whether the activity was fraudulent (1) or not (0)

And the second dataset can be used to get user country based on their Ip address. For each country, it gives a range. If the numeric ip address falls within that range, then the ip address belongs to the corresponding country:

```
## Read the ip addresses data
ip_addresses=read.csv("/Users/mitulshah/Downloads/Downloads/Giulio Palombo
Book Solutions/Identifying Fraudulent Activities/IpAddress_to_Country.csv")

## Look at the head of the data
knitr::kable(head(ip_addresses))

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

lower_bound_ip_address	upper_bound_ip_address	country
16777216	16777471	Australia
16777472	16777727	China
16777728	16778239	China
16778240	16779263	Australia
16779264	16781311	China
16781312	16785407	Japan

lower_bound_ip_address : the lower bound of the numeric ip address for that country

upper_bound_ip_address : the upper bound of the numeric ip address for that country

country : the corresponding country. If a user has an ip address whose value is within the upper and lower bound, then she is based in this country

1. For each user, determine their country based on the IP address

```
## Let's add the country to the original data set by using the ip address
data_country = rep(NA, nrow(data))

for (i in 1: nrow(data)){
  ## use data$ip_address value for each user to subset ip_address dataset and
then pick the corresponding country
  tmp = as.character(ip_addresses [data$ip_address[i] >=
ip_addresses$lower_bound_ip_address &
                                   data$ip_address[i] <=
ip_addresses$upper_bound_ip_address,
                                   "country"]
                     )
  if (length(tmp) == 1)  {data_country[i] = tmp}

}

data$country = data_country

knitr::kable(head(data.frame(sort(table(data$country), decreasing=T)), 10))
```

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

Var1	Freq
United States	58049
China	12038
Japan	7306
United Kingdom	4490
Korea Republic of	4162
Germany	3646
France	3161
Canada	2975
Brazil	2961
Italy	1944

2 and 3. Build a model to predict whether an activity is fraudulent or not. Explain how different assumptions about the cost of false positives vs false negatives would impact the model. Your boss is a bit worried about using a model she doesn't understand for something as important as fraud detection. How would you explain her how the model is making the predictions? Not from a mathematical perspective (she couldn't care less about that), but from a user perspective. What kinds of users are more likely to be classified as at risk? What are their characteristics?

Before jumping into building a model, think about whether you can create new powerful variables. This is called feature engineering and it is the most important step in machine learning. However, feature engineering is quite time consuming. In a take-home you should just give an idea of how you would do it and emphasize that with more time you would go deeper into it.

A few obvious variables that can be created here could be:

1. Time difference between sign-up time and purchase time
2. If the device id is unique or certain users are sharing the same device (many different user ids using the same device could be an indicator of fake accounts)
3. Same for the ip address. Many different users having the same ip address could be an indicator of fake accounts

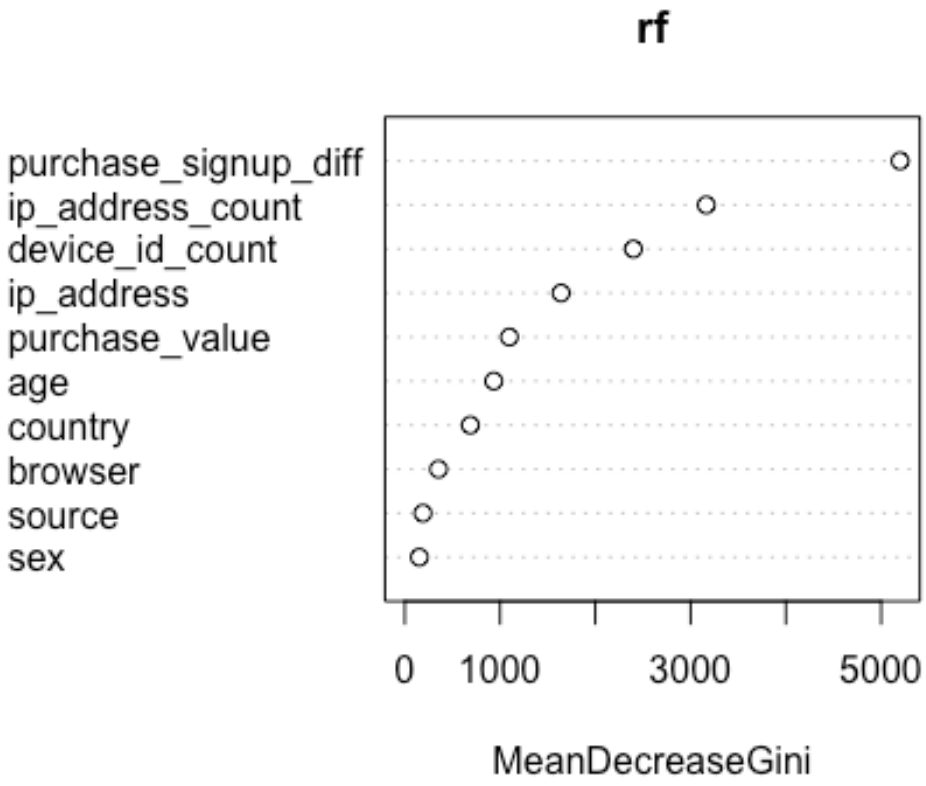
```
## Setting the seed
set.seed(4321)

## Make them dates
data[, "signup_time"] = as.POSIXct(data[, "signup_time"], tz="GMT")
data[, "purchase_time"] = as.POSIXct(data[, "purchase_time"], tz="GMT")
## and take the difference
data$purchase_signup_diff =  as.numeric(difftime(data$purchase_time,
data$signup_time, unit="secs"))

## Check how for each device id, how many different users had it
data = data %>%
       group_by(device_id) %>%
       mutate (device_id_count = n())

## Check how for each ip address, how many different users had it
data = data.frame(data %>%
                  group_by(ip_address) %>%
                  mutate (ip_address_count = n())
                  )

## Data set for the model. Drop first 3 vars and device id.
```

```r
data_rf = data[, -c(1:3, 5)]

## Replace the NA in the country var
data_rf$country = as.character(data_rf$country)
data_rf$country[is.na(data_rf$country)]="Not_found"

## Just keep the top 50 country, everything else is "other"
data_rf$country = ifelse(data_rf$country %in%
names(sort(table(data_rf$country), decreasing = TRUE )
)[51:length(unique(data_rf$country))], # after top 50 countries
                         "Other", data_rf$country
                         )

## Make class a factor
data_rf$class = as.factor(data_rf$class)

## All characters become factors
data_rf[sapply(data_rf, is.character)] <- lapply(data_rf[sapply(data_rf,
is.character)], as.factor)

## Train/test split
train_sample = sample(nrow(data_rf), size = nrow(data)*0.66)
train_data = data_rf[train_sample,]
test_data = data_rf[-train_sample,]

## Build a random forest
rf = randomForest(y=train_data$class, x = train_data[, -7],
                  ytest = test_data$class, xtest = test_data[, -7],
                  ntree = 50, mtry = 3, keep.forest = TRUE)

rf

##
## Call:
##  randomForest(x = train_data[, -7], y = train_data$class, xtest =
test_data[,      -7], ytest = test_data$class, ntree = 50, mtry = 3,
keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4.57%
## Confusion matrix:
##       0    1 class.error
## 0 90070  306 0.003385855
## 1  4253 5104 0.454526023
##                 Test set error rate: 4.32%
## Confusion matrix:
##       0    1 class.error
```

```
## 0 46485  100 0.002146614
## 1  2119 2675 0.442010847
```

Let's start by checking variable importance:

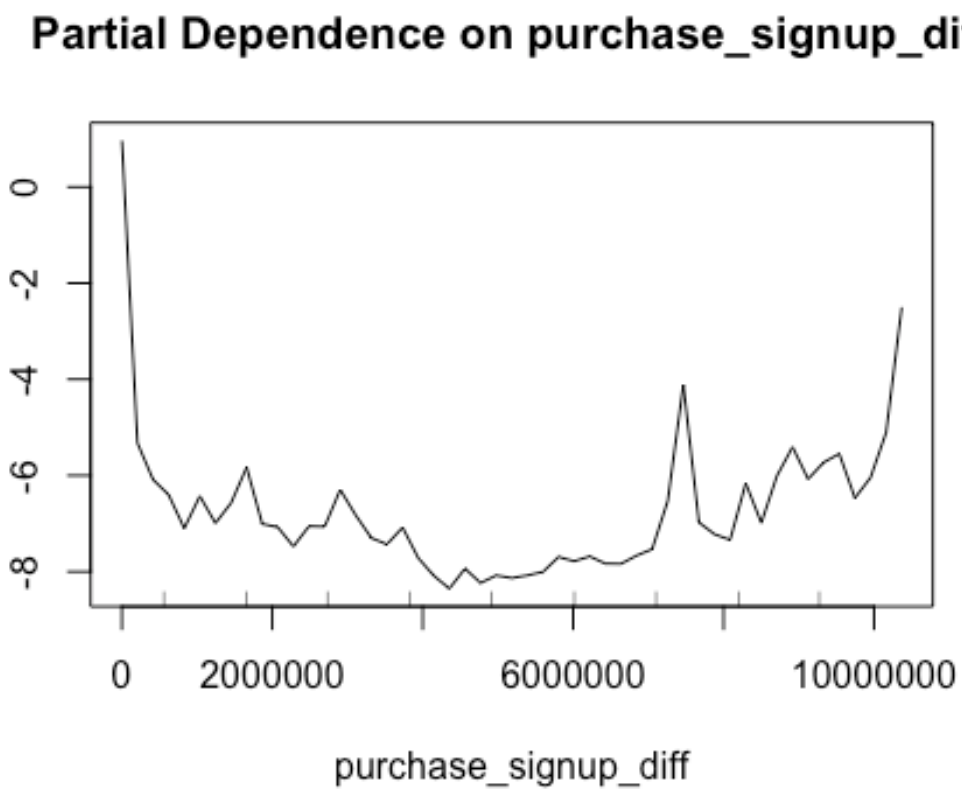
```
## Variable Importance Plot
varImpPlot(rf,type=2)
```



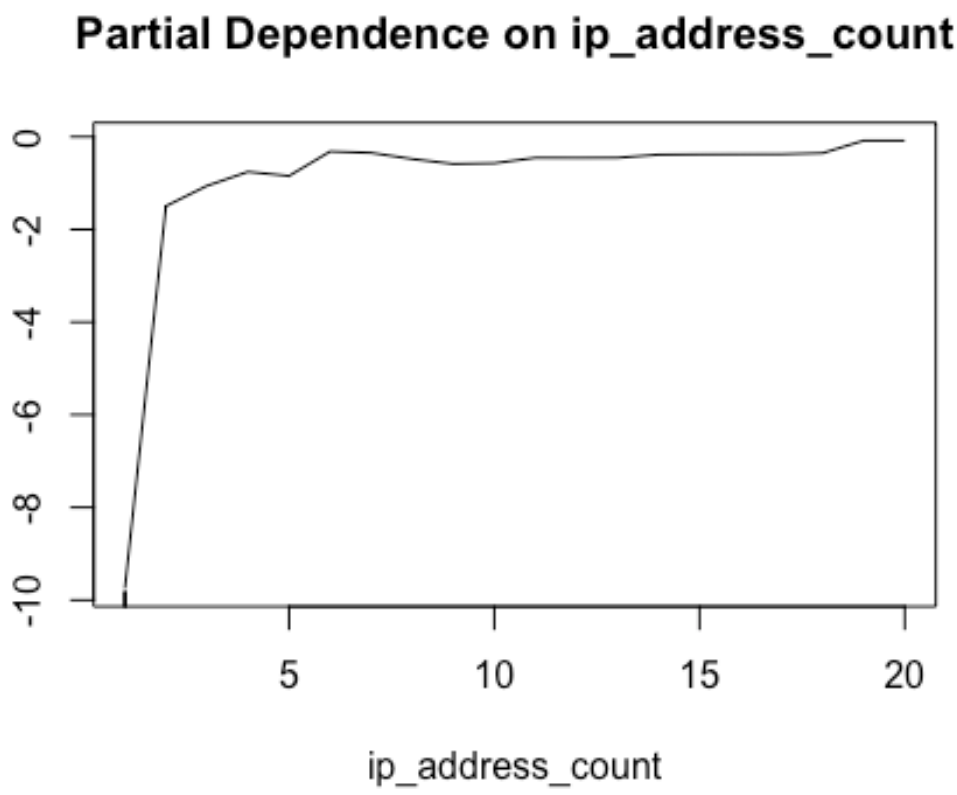
Interesting! Purchase_Signup_Difference is the most important one. Source and Sex doesn't seem to matter at all.

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

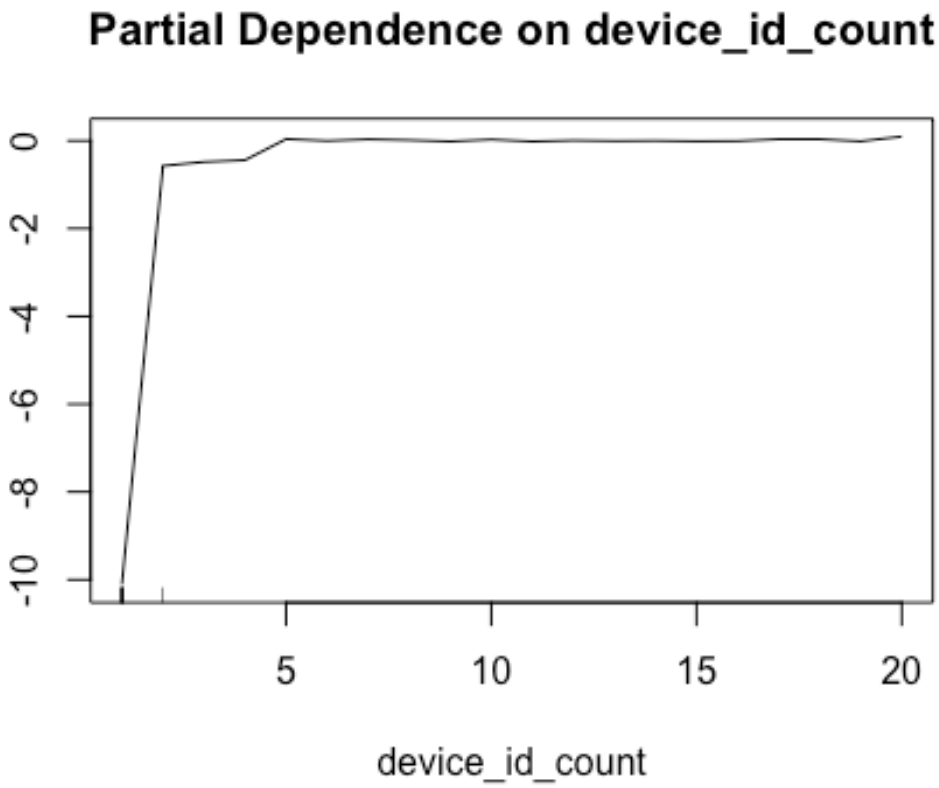
```
## Partial Dependence Plots
partialPlot(rf, train_data, purchase_signup_diff, 1)
```



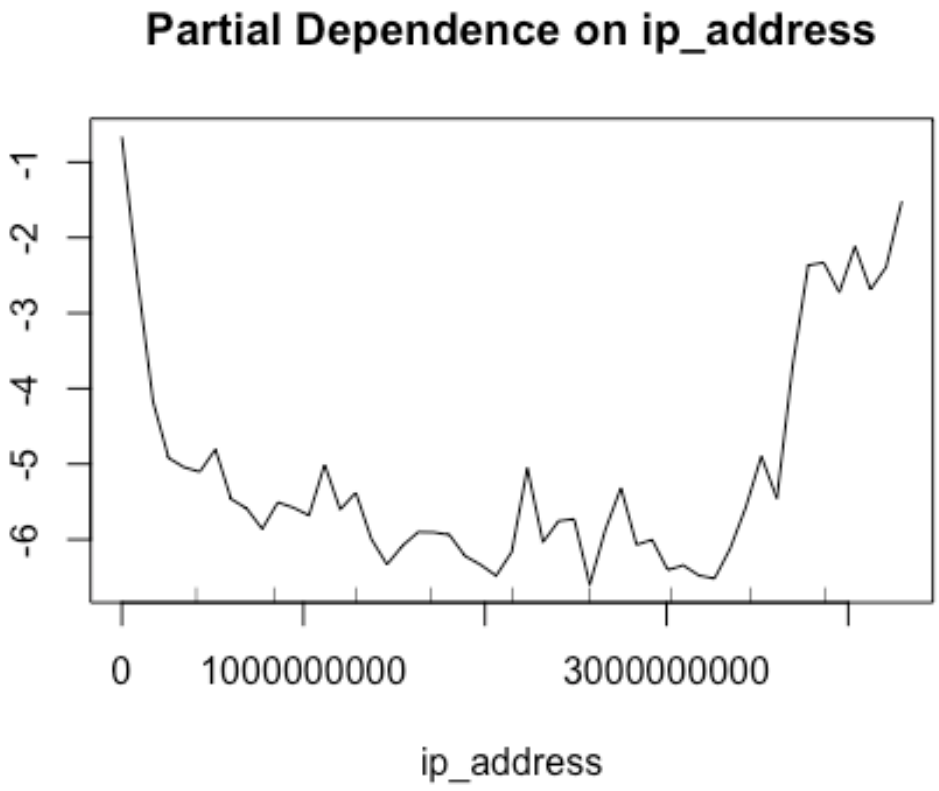

```
partialPlot(rf, train_data, ip_address_count, 1)
```



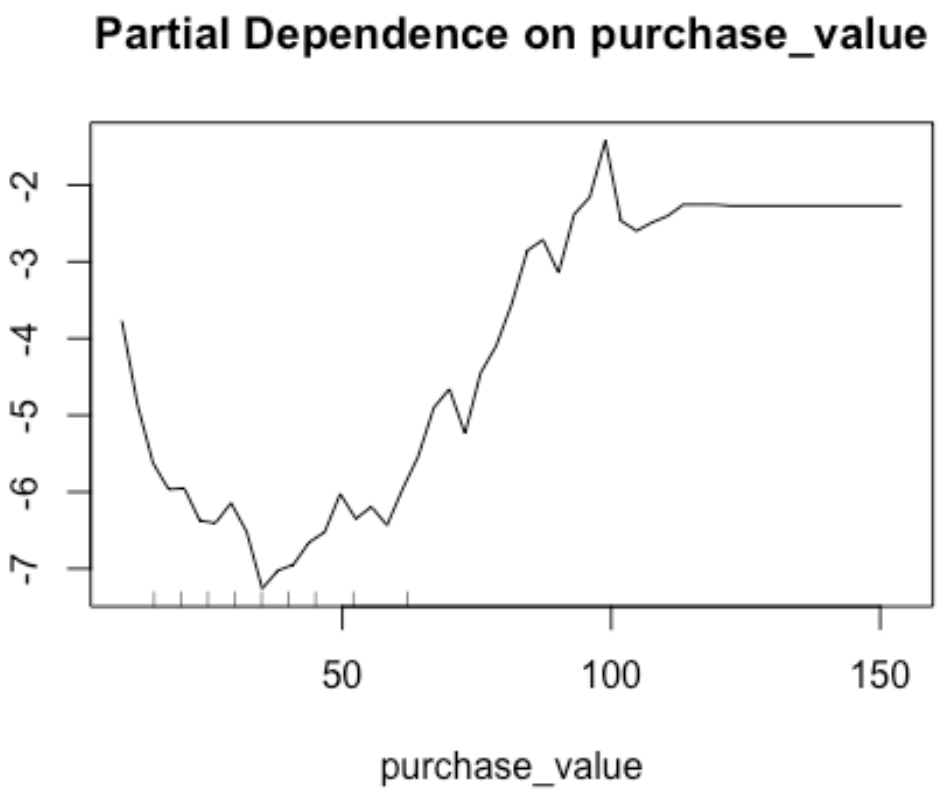

```
partialPlot(rf, train_data, device_id_count, 1)
```



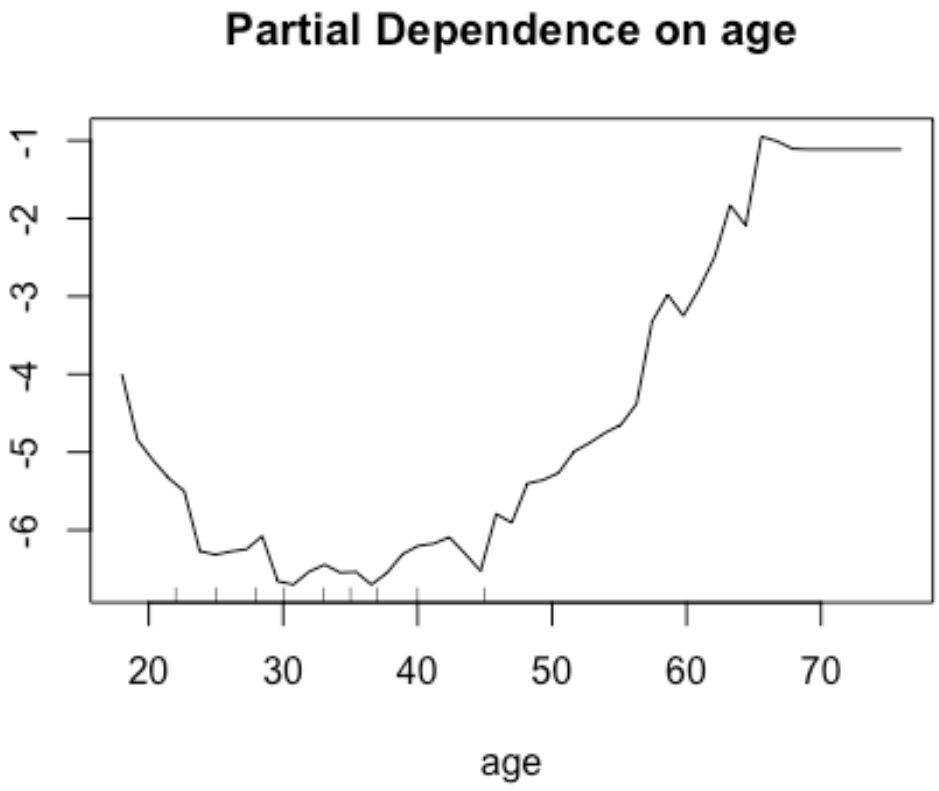

```
partialPlot(rf, train_data, ip_address, 1)
```



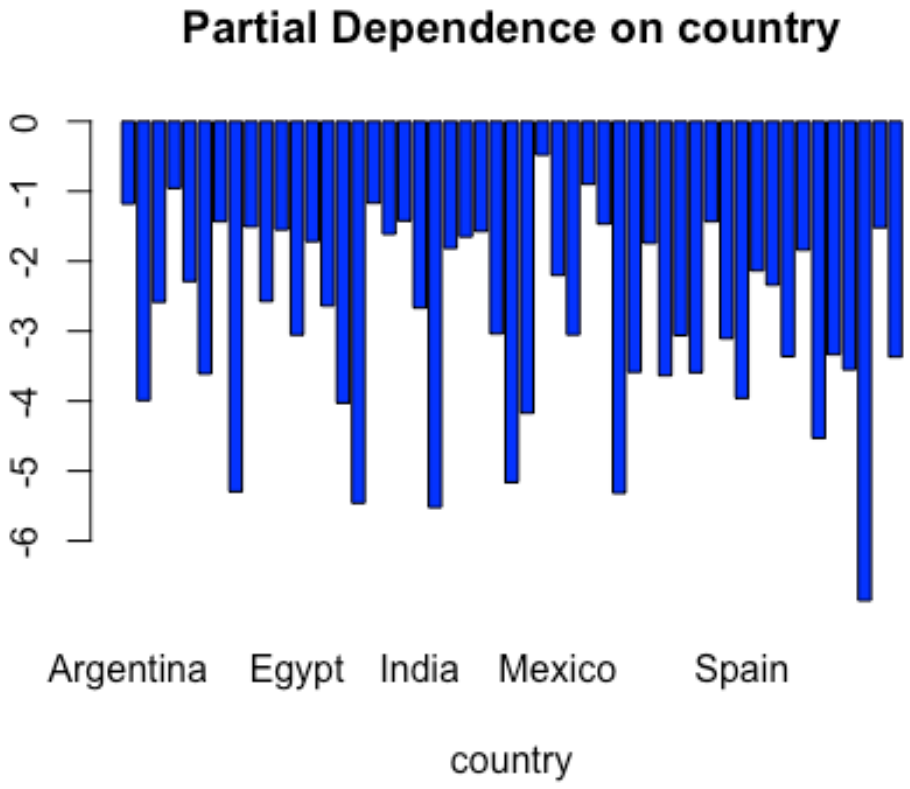

```
partialPlot(rf, train_data, purchase_value, 1)
```



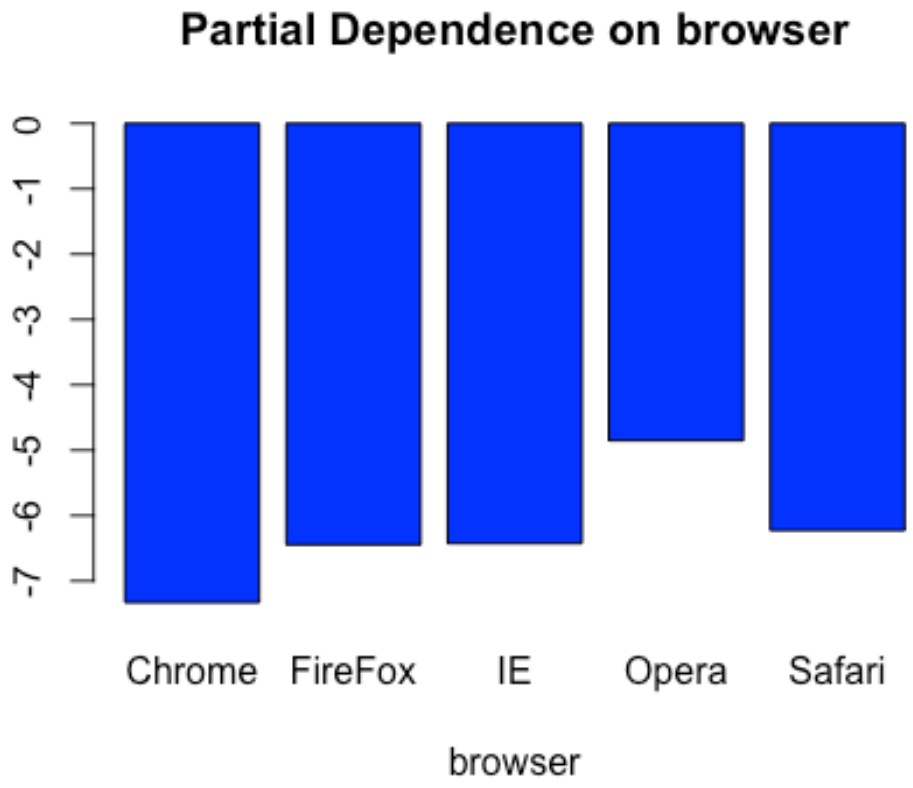

```r
partialPlot(rf, train_data, age, 1)
```



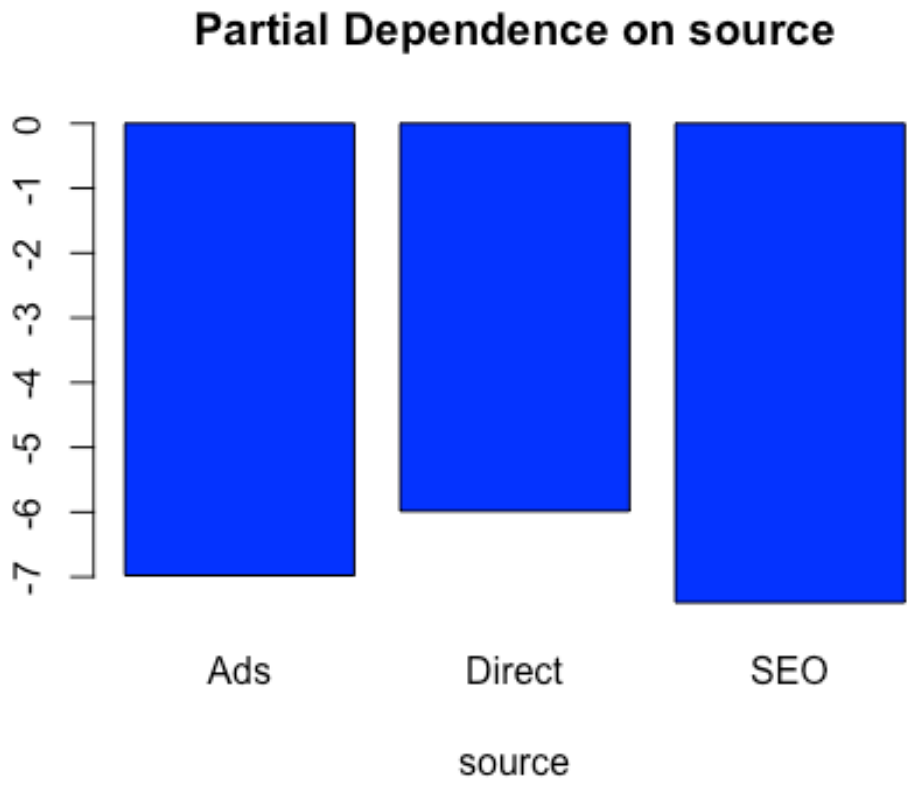

```
partialPlot(rf, train_data, country, 1)
```



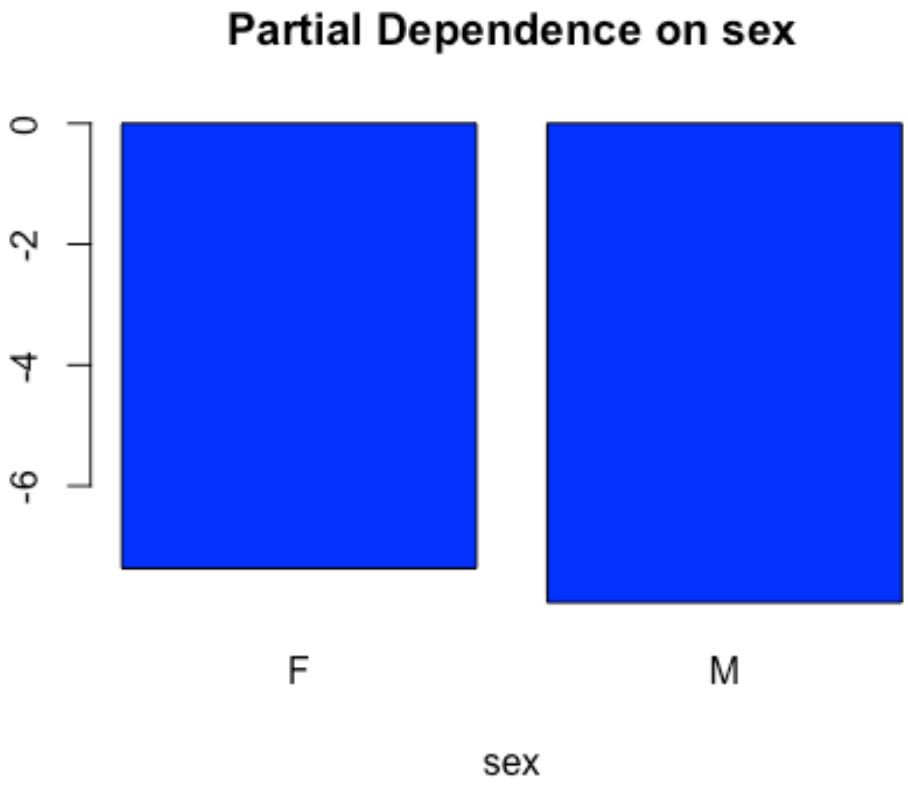

```
partialPlot(rf, train_data, browser, 1)
```




```
partialPlot(rf, train_data, source, 1)
```




```
partialPlot(rf, train_data, sex, 1)
```



Let's print the Random Forest Model again.

```
## Random Forest Model
rf

##
## Call:
##  randomForest(x = train_data[, -7], y = train_data$class, xtest =
test_data[,      -7], ytest = test_data$class, ntree = 50, mtry = 3,
keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4.57%
## Confusion matrix:
##       0    1 class.error
## 0 90070  306 0.003385855
## 1  4253 5104 0.454526023
##                 Test set error rate: 4.32%
## Confusion matrix:
##       0    1 class.error
```

```
## 0 46485  100 0.002146614
## 1  2119 2675 0.442010847
```

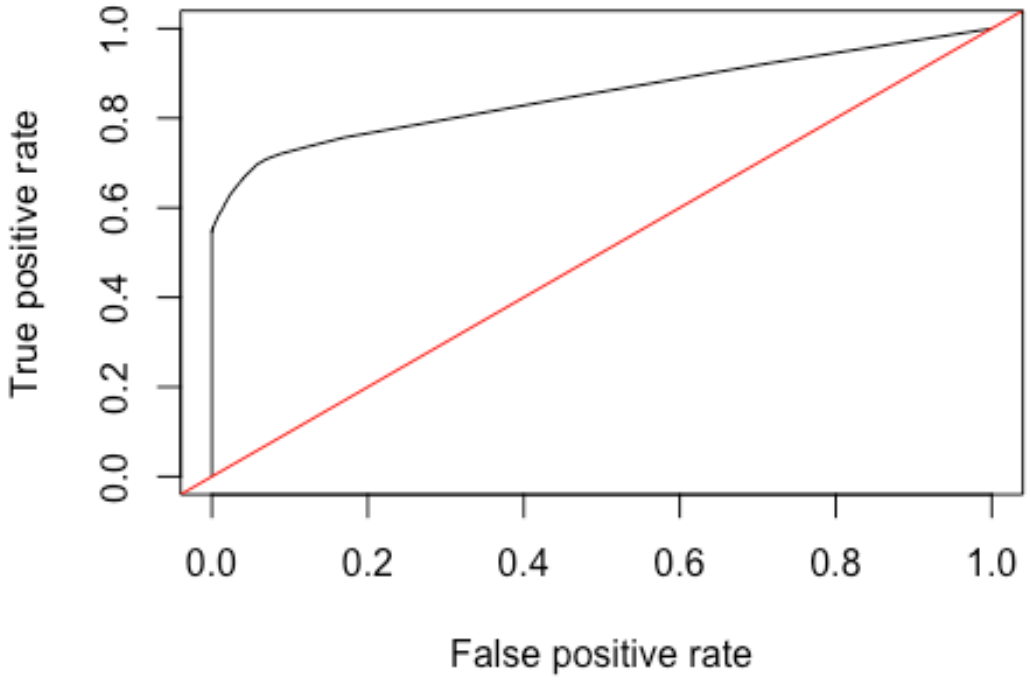
The confusion matrix looks good. We are not overfitting given that OOB and test results are very similar. Class 0 error is almost zero, and we are doing surprisignly well also with regards to class 1 error, which is pretty rare.

However, since the challenge asks about false positives and false negatives, this usually implies building the ROC and look for possible cut-off points. And in general, especially when dealing with fraud, you should always do the cut-off analysis. There is no reason why the default 0.5 value has to be the best one.

```
## Let's combine in one data set model predictions and actual values.The
first column are the actual classes in our test set and the second one the rf
predicted scores
rf_results = data.frame (true_values = test_data$class,
                         predictions = rf$test$votes[,2]
                         )
```

Is really 0.5 the best possible cut-off? It really depends on what we are optimizing for (accuracy? true positive? true negative? etc.)

```
## This creates an object with all the information you can possibly need
about how different cutoff values impact all possible metrics: i.e. true
positive, true negative, false positive, false negative
pred = prediction (rf_results$predictions, rf_results$true_values)

## Now let's just plot the ROC and look at true positive vs false positive
perf = performance (pred, measure = 'tpr', x.measure = "fpr")
plot(perf) + abline(a=0, b=1, col = 'red') ## the red line is randomness
```




```
## integer(0)
```

Based on the ROC, if we care about minimizing false positive, we would choose a cut-off that would give us true positive rate of ~0.5 and false positive rate almost zero (this was similar to the random forest output). However, if we care about maximizing true positive rate, we will have to decrease the cut-off. This way we will classify more events as "1": some will be true ones (so true positive goes up) and many, unfortunately, will be false ones (so false positive will also go up).

If we want to be a bit more accurate, we can use the approach of maximizing true positive rate - false positive rate, this is the same as maximizing (1-class1_error) -class0_error.

```
## Get true/false positives/negatives for different cutoff points
error_cutoff = data.frame (pred@cutoffs,
                           pred@tn,
                           pred@fn,
                           pred@fp,
                           pred@tp,
                           row.names = NULL)
colnames(error_cutoff) = c("cutoff", "tn", "fn", "fp", "tp")
## Let's add to this dataset class 0 and class 1 error and accuracy.
## Class 0 error, aka false positive rate, is 1 - tn / (tn + fp)
error_cutoff$class0_error = 1 - error_cutoff$tn/(error_cutoff$tn +
```

```
error_cutoff$fp)
## Class 1 error is 1 - tp / (tp + fn), aka  1 - true positive rate
error_cutoff$class1_error = 1 - error_cutoff$tp/(error_cutoff$tp +
error_cutoff$fn)
## Combine two class errors into 1 metric
error_cutoff$optimal_value = 1 - error_cutoff$class1_error -
error_cutoff$class0_error
## find best value
knitr::kable (error_cutoff[order(error_cutoff$optimal_value,
decreasing=TRUE), c("cutoff", "class1_error", "class0_error")][1,])

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

	cutoff	class1_error	class0_error
36	0.14	0.2918231	0.0692712

The best value is for ~71% true positive rate (specifically 1-0.29) and 6% false positive rate. You can find the corresponding point on the ROC curve.

4. Let's say you now have this model which can be used live to predict in real time if an activity is fraudulent or not. From a product perspective, how would you use it? That is, what kind of different user experiences would you build based on the model output?

You now have a model that assigns to each user a probability of committing a fraud. And, despite our model doing pretty well, no model is perfect. So you will have some misclassifications. It is crucial now to think about building a product that minimizes the impact (aka cost) of those misclassifications. A very commonly used approach is to think about creating different experiences based on the model score. For instance:

If predicted fraud probability < X, the user has the normal experience (the high majority of users should fall here)

If X <= predicted fraud probability < Z (so the user is at risk, but not too much), you can create an additional verification step, like phone number verifications via a code sent by SMS or asking to log in via social network accounts

If predicted fraud probability >= Z (so here is really likely the user is trying to commit a fraud), you can tell the user his activity has been put on hold, send this user info to someone who reviews it manually, and finally either block the user or decide it is not a fraud so the session is resumed.