Laterooms.com

Data Science Kata

Author: Joseph Allen

June 25, 2017

Chapter 1

Introduction

1.1 The Task

Given the data in hotel_engagement.csv.tar.gz, which contains information about sessions on the Hotel Details page:

- How do LateRooms visitors use the our site?
- What insights can LateRooms act on?

1.2 Getting Started

our data source has 2363267 lines which is too many to inspect trivially so I will make a new file of just the first 1000,10000 and 50000 rows using the following:

```
head -1000 input > 1000.csv
head -10000 input > 10000.csv
head -50000 input > 50000.csv
```

I started by investigating the "transaction" column hoping that I could use this to label the behavior of successful bookers, In my sample of 50000 I got a 0 in every column, implying no bookings were made in my rows. This could perhaps be because I used the head function instead of taking a random sample and perhaps caught a strange time of day where the website had high bot traffic, or is perhaps a time where users are much less likely to book.

Now instead of using the `head` command, I will instead take a random sample with hopes of retrieving a more meaningful distribution.

1.3 Conversion

I wrote a short python script which finds the conversion for our sample giving the following outputs:

Number of Rows	Conversion
1000	3.40%
10000	4.25%
50000	4.30%
2363267	4.24%

The script is as follows:

```
# imports
import sys
import csv

transaction_count = 0.
count = 0.

# Read in CSV
print "Importing :", sys.argv[1]
with open(sys.argv[1], 'rb') as csvfile:
     spamreader = csv.reader(csvfile, delimiter=',', quotechar='|')
     for row in spamreader:
         count += 1
         if(row[1] == '1'):
             transaction_count += 1

# Outputs
print "number of transactions: ", transaction_count
print "number of sessions: ", count
print "conversion :", transaction_count/count
```

What we have understood here is the average conversion for the page, which we could track over time. We have also shown that our 10,000 row sample is a fairly accurate representation of the large data set. While in Data Science we are sometimes very interested in information missed by such small samples I will continue from here.

1.4 Sparsity

One of the issues of Data Science is storing too much. Something to notice in this particular data set is how the sparse it is. I have modified the last script to find how many zeros, or arguably wasted columns we have.

The script is as follows:

```
# imports
import sys
import csv

zero_count = 0.
count = 0.

# Read in CSV
print "Importing :", sys.argv[1]
with open(sys.argv[1], 'rb') as csvfile:
     spamreader = csv.reader(csvfile, delimiter=',', quotechar='|')
     for row in spamreader:
         for col in row:
            count += 1
            if(col == '0' and col > 0):
                zero_count += 1

# Outputs
print "number of zeros: ", zero_count
print "number of stored data: ", count
print "zero percantage: ", zero_count/count
```

Number of Rows	Sparsity
1000	95.89%
10000	95.84%
50000	95.86%
2363267	95.86%

As we can see, about 95.86% of the table is unused. I ponder that there is a better storage solution but I will not investigate it further here. perhaps the absence of data is equally as important. Special algorithms can be applied to these so called Sparse Matrices which could be beneficial for both storage and analysis in larger situations.

1.5 Relevance

I will briefly discuss my interpretation of some of the column headings I believe will be important to the later investigation.

- **Transaction** - I take this to mean 1 for a booking is made on this session, 0 if it is not.
- **Jumping** - There are a few columns (jump_availability, jump_reviews, jump_facilities, jump_info) which count the number of times users click to jump to certain sections on the page, these are not shown on mobile.
- **Upgrades** - we track how many upgrades are added or removed.
- **Footer and Navigation** - we track clicks to other pages from the Footer and navigation bar.
- **Reviews** - we track when review previews are clicked.
- **Click to Call** - we track when a click to call is made.

Chapter 2

My Problem

2.1 Introduction

The most trivial definition of success and failure in this scenario is whether or not a booking is made on the session, Some questions I had were the following:

- Does somebody who jumps around the page book more or less? And what does this mean?
- Do people who navigate to other pages return and are they more likely to book?
- Are upgrades added? Are they removed?
- Do reviews and calling the contact center help?

I have decided only to focus on the first question proposed.

2.2 Does Jumping affect bookings

I have written a python script which splits the CSV into separate training and test sets. And then applies a classifier. I have decided to use AdaBoost since it handles outliers well and due to the sparsity this may be important.

```
import pandas as pd
from time import time
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier

# Read in CSV
df = pd.read_csv('1000_rows.csv')
length = len(df.index)

# Sliced CSV
df_jump = df[['transactions','jump_availability','jump_reviews','jump_facilities','jun

# Get our seperate arrays
transactions = df_jump.values[:,0]
jump = df_jump.values[:,1:5]

# split labels 80-20
labels_train = transactions[:int(length*0.8)]
labels_test = transactions[int(length*0.8):length]

# split features 80-20
features_train = jump[:int(length*0.8)]
features_test = jump[int(length*0.8):length]
```

```
# Create Classifier
clf = AdaBoostClassifier(n_estimators=1000)

# Training
t0 = time()
clf.fit(features_train, labels_train)
print "training time:", round(time()-t0, 3), "s"

# Test
t0 = time()
pred = clf.predict(features_test)
print "predicting time:", round(time()-t0, 3), "s"

# Accuracy
print(accuracy_score(labels_test, pred))
```

Gives the following output for 1000 rows:

```
training time: 2.371 s
predicting time: 0.11 s
0.945
```

Gives the following output for 10000 rows:

```
training time: 3.526 s
predicting time: 0.269 s
0.9615
```

Gives the following output for 50000 rows:

```
training time: 8.887 s
predicting time: 0.885 s
0.9558
```

This has trained relatively fast so I decided to run it on the entire dataset:

```
training time: 521.778 s
predicting time: 54.707 s
0.939541398147
```

At a first glance a 94% prediction accuracy on whether a transaction is made based purely off the jump usage seems incredible. I fret that this is too good to be true due to the previously investigated sparsity of this data. The sparsity is 95.86% in this data set so simply choosing 0 as every outcome would be more significant. This leads me to believe that we these two features are unlinked. If I spent more time on this project I would perhaps focus only on the successful transactions to see if they can be predicted more accurately.