Fraud Detection

Identifying fraudulent transactions using sklearn, flask, and bootstrap.

Anton, Jake M, Matt G and Mitch

Data:

Upcoming Event

Event Calendar

Upcoming Events Calendar

Event

Special Event

Event Venue

Concert Tickets

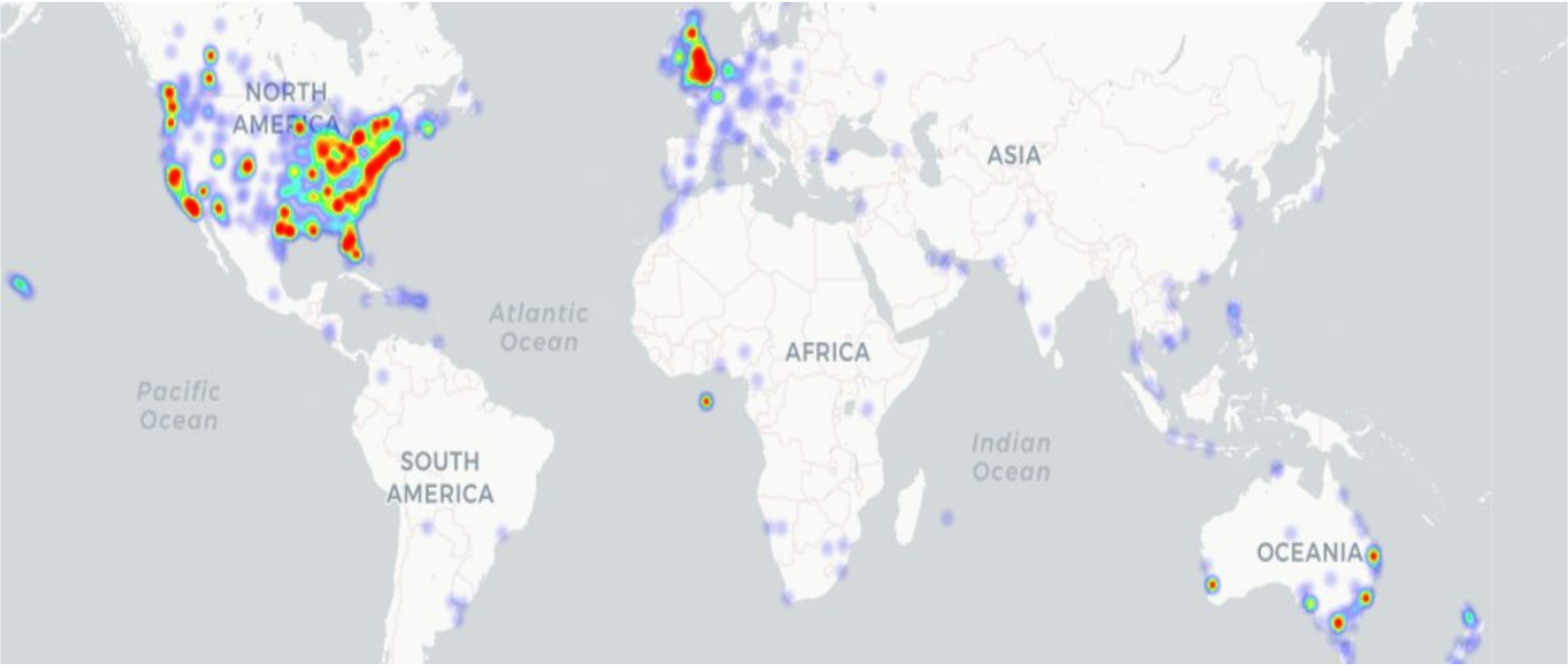
Eventbrite.com

14337 Observations

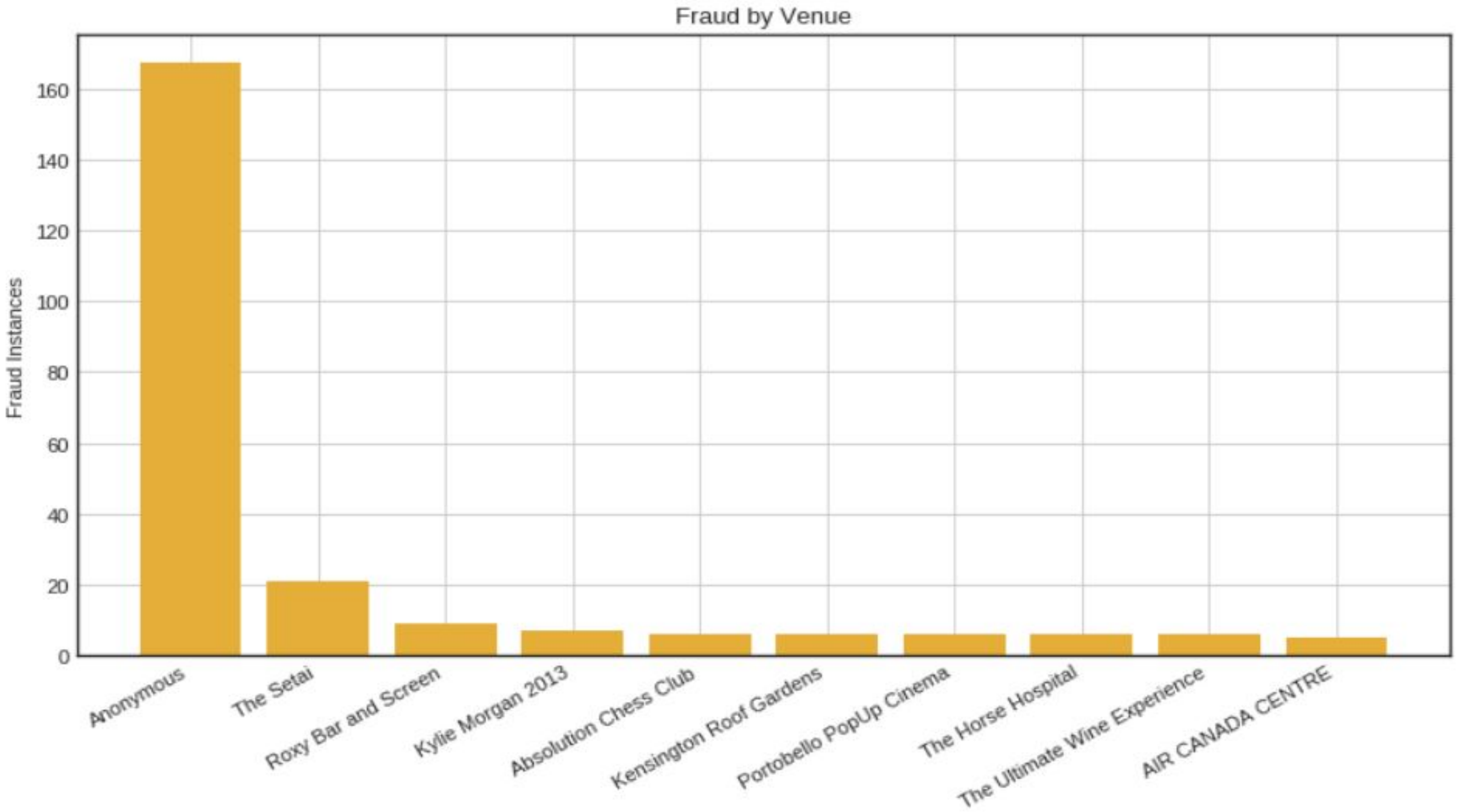
45 Features

2002 - 2013

Fraud Heatmap

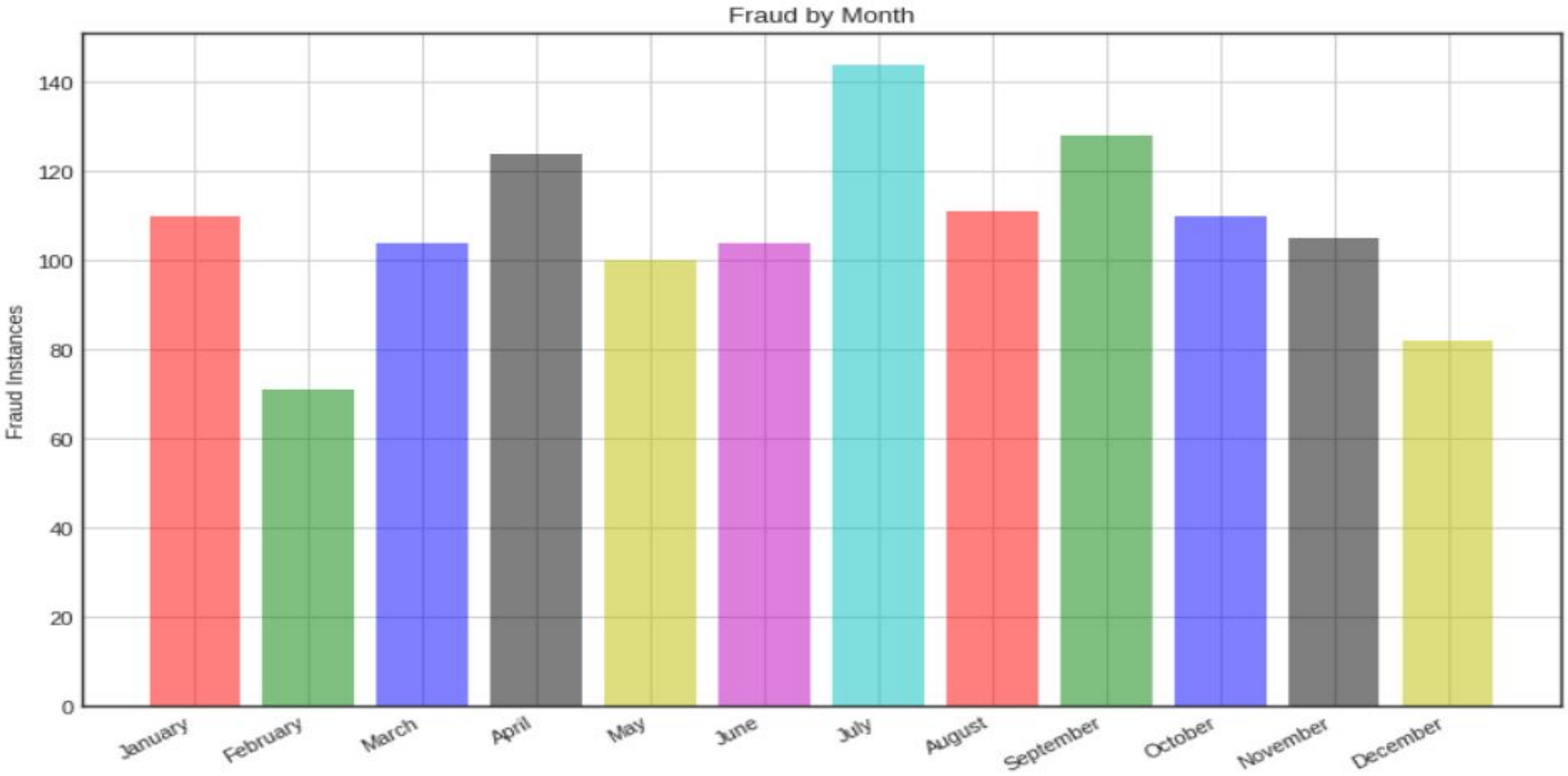


EDA



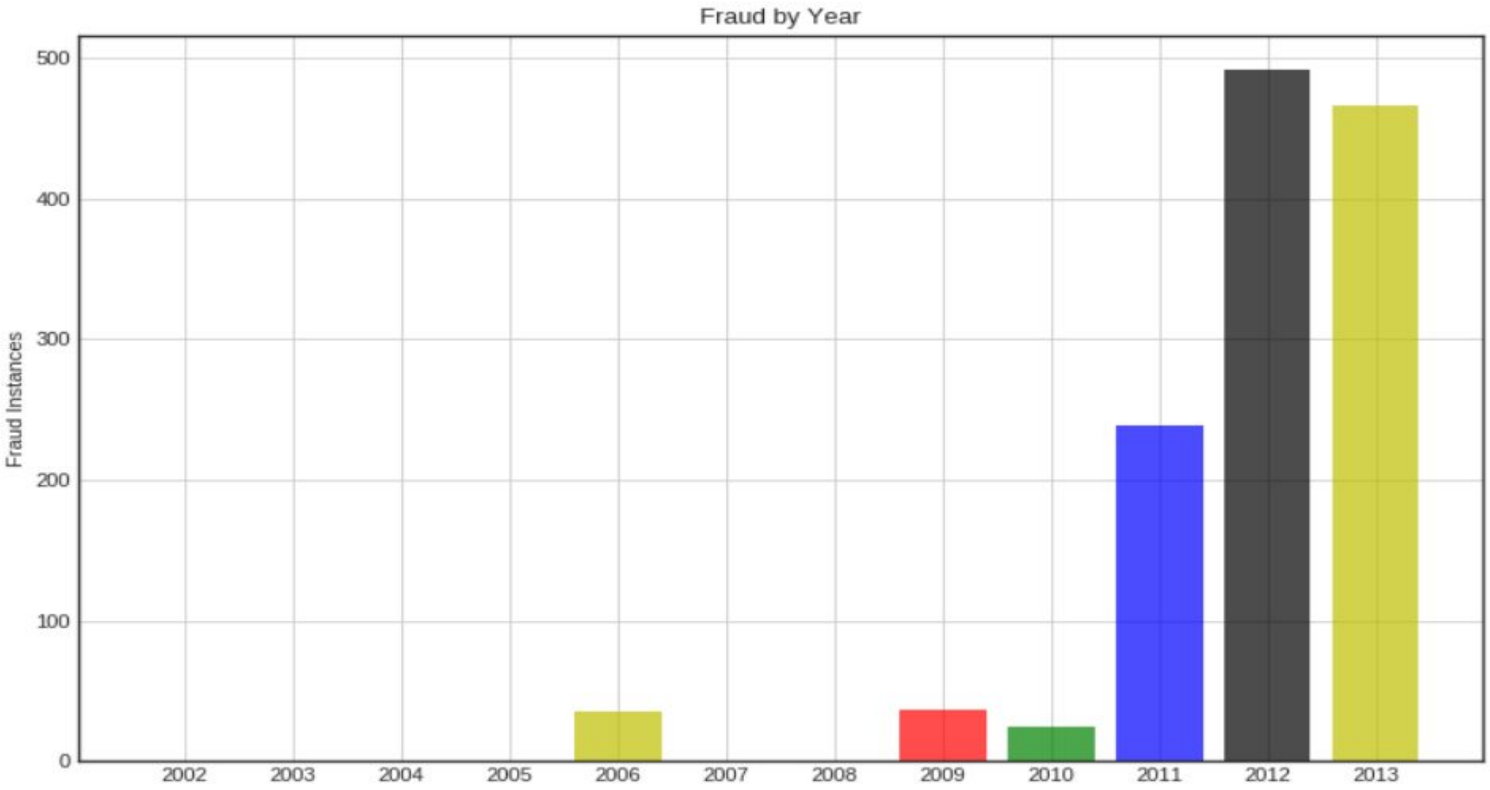
Fraud by Venue
Fraud Instances
160
140
120
100
80
60
40
20
0
Anonymous
The Setai
Roxy Bar and Screen
Kylie Morgan 2013
Absolution Chess Club
Kensington Roof Gardens
Portobello PopUp Cinema
The Horse Hospital
The Ultimate Wine Experience
AIR CANADA CENTRE

EDA



Fraud by Month
Fraud Instances
0
20
40
60
80
100
120
140
January
February
March
April
May
June
July
August
September
October
November
December

EDA


Fraud by Year
Fraud Instances
0
100
200
300
400
500
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

Solution:

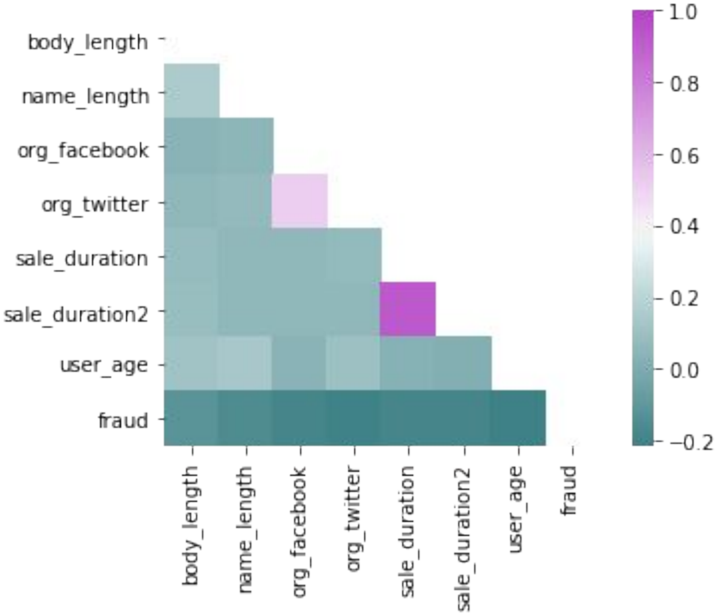
- Create a machine-learning model that will detect *potentially* fraudulent transactions, and flag them for manual review
- Considerations:
    - An oversensitive model will have too many false positives, straining our resources and costing us valuable man-hours.
    - An undersensitive model will allow too many fraudulent transactions to go through, costing us revenue and hurting our credibility with paying customers.

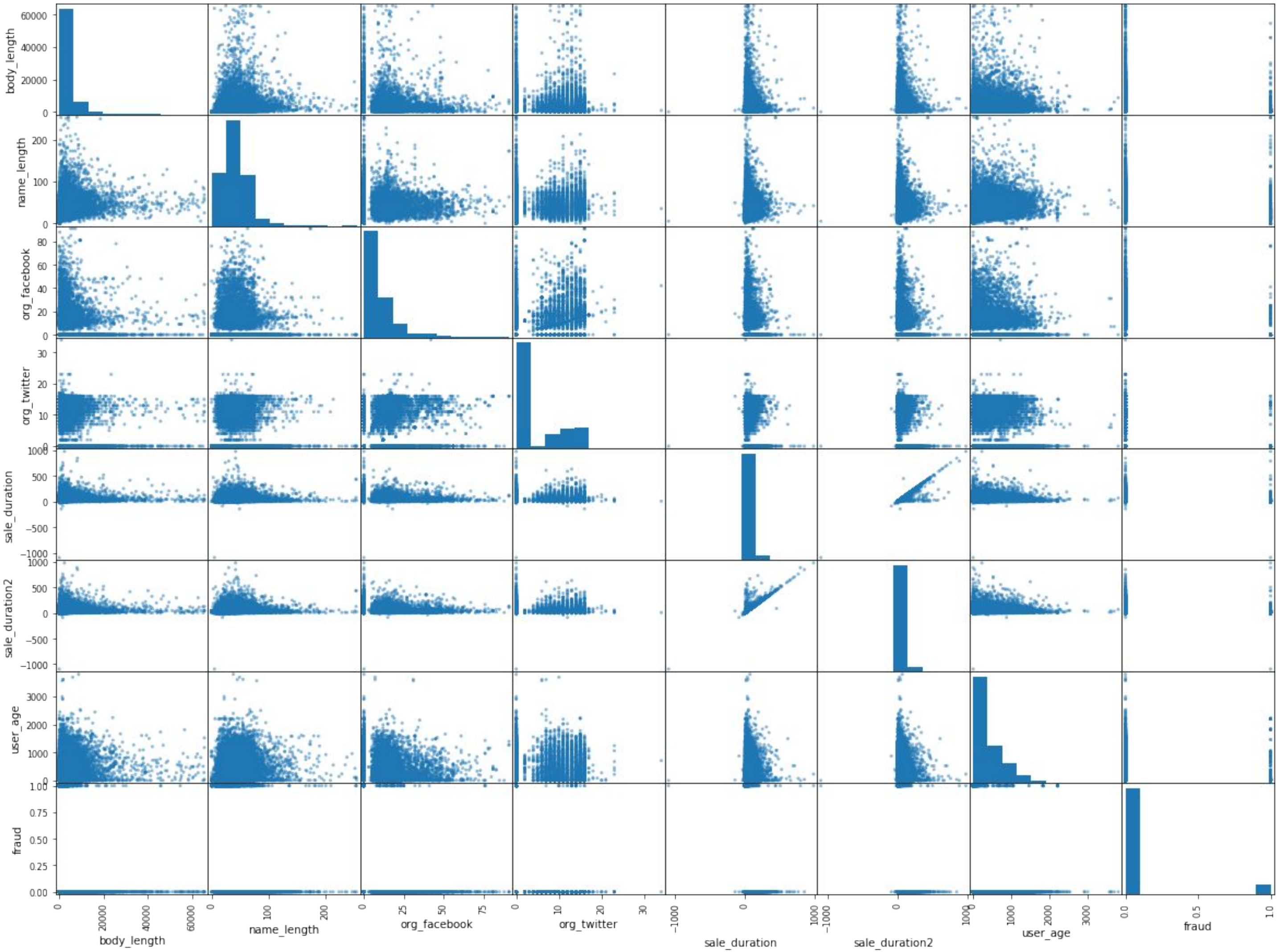
Scope:

- We initially took only the numerical and boolean features
- Trained models using Logistic Regression, RandomForest, and Gradient Boosting
- Measured success by accuracy and false positive / false negative rates

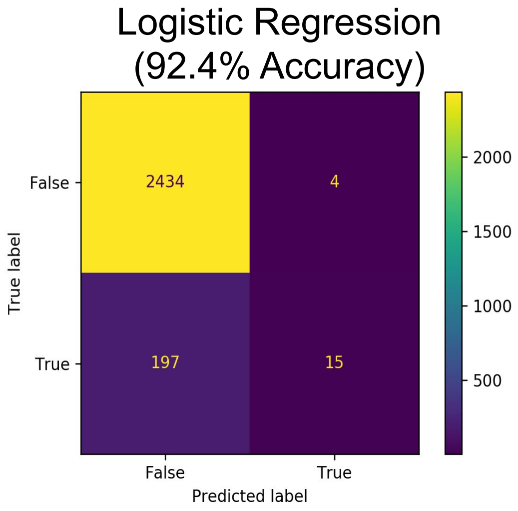
What information do we have at our disposal?

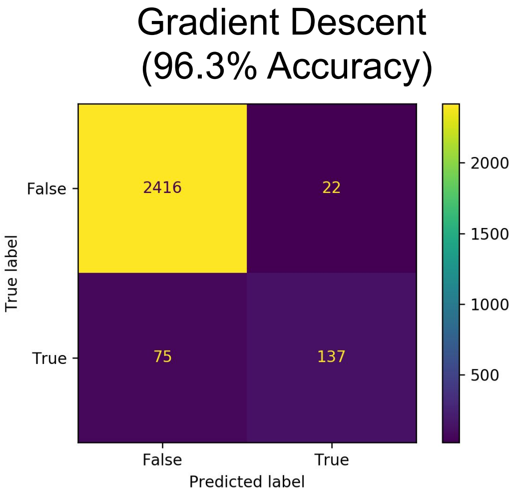
```
[180]: approx_payout_date   -0.042553
       body_length          -0.118308
       channels             -0.165358
       delivery_method      -0.194046
       event_created        -0.006436
       event_end            -0.042553
       event_published      -0.077322
       event_start          -0.044995
       fb_published         -0.099143
       gts                  -0.017875
       has_analytics        -0.084626
       has_header           -0.082101
       has_logo             -0.169485
       name_length          -0.158447
       num_order            -0.078008
       num_payouts          -0.083433
       object_id             0.026721
       org_facebook         -0.181792
       org_twitter          -0.205692
       sale_duration        -0.179512
       sale_duration2       -0.179550
       show_map             -0.076217
       user_age             -0.215929
       user_created          0.184360
       user_type            -0.213911
       venue_latitude        0.010126
       venue_longitude       0.066057
       fraud                 1.000000
       Name: fraud, dtype: float64
```

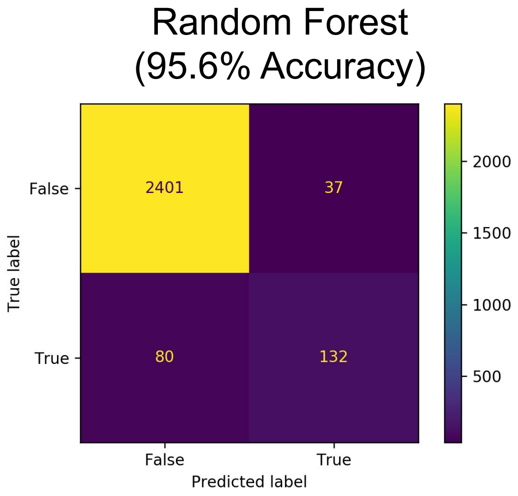



body_length
name_length
org_facebook
org_twitter
sale_duration
sale_duration2
user_age
fraud

Building the Model







*accuracies measured on validation data

Web API Demo

Actionable items:

- Our models were very simply and left much of the data unused, given the accuracy we decided to move forward with deploying our model.
- Given more time, both the text and date-time data could be used to improve our model.
- Connect to the online database to detect fraud in real-time.

Questions:

Anton: Time-Series and Geographical Analysis

Jake: Prediction Models and Web API

Matt: Feature-Engineering and Data-Engineering

Mitch: WebDev and Model Deployment

References:

1. https://www.prnewswire.com/news-releases/payment-card-fraud-losses-reach-27-85-billion-300963232.html