Fraud Detection Case Study

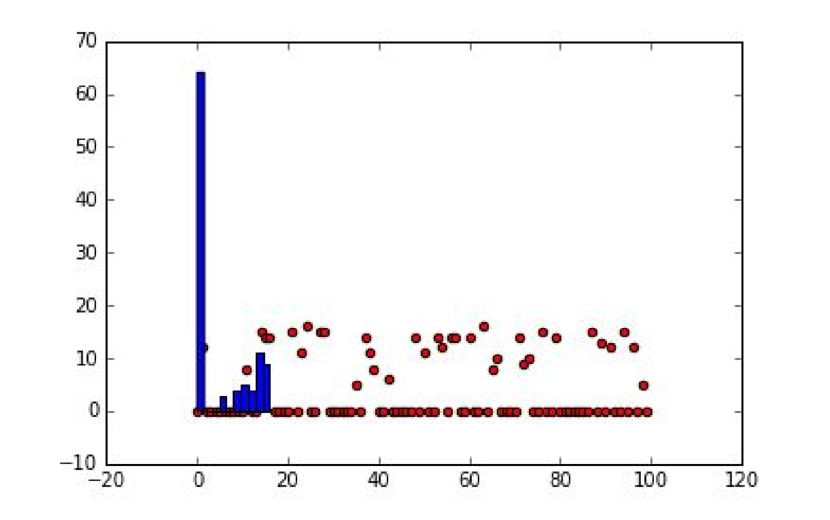
Jim, Kayla, Trent, Juno

Objectives

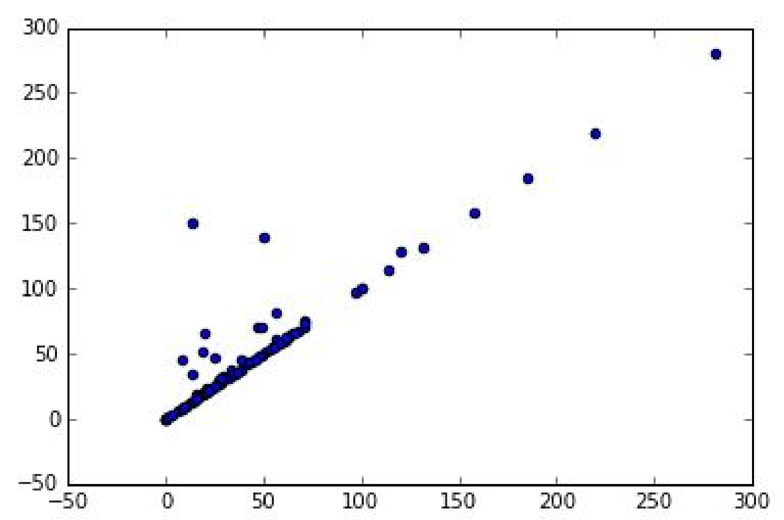
- Create a model that helps EventBrite identify fraudsters
- Provide sustainable software by deploying our model in the cloud

Exploratory Data Analysis

twitter_org



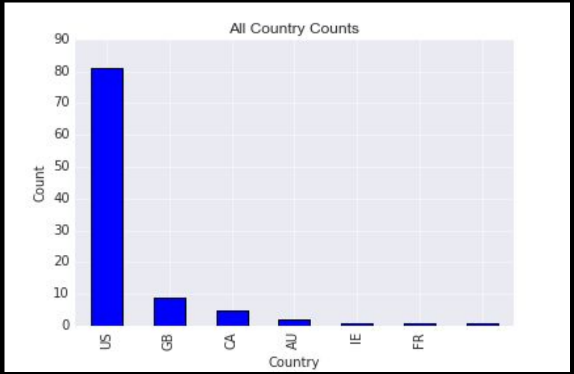
sale_duration versus sale_duration2

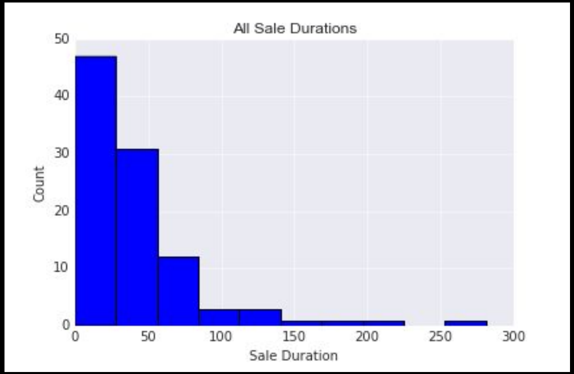


Country

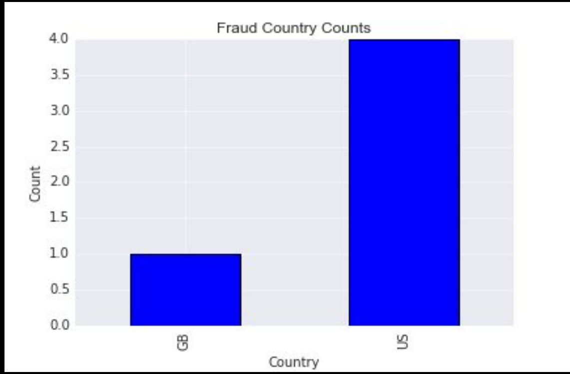
Sale Duration

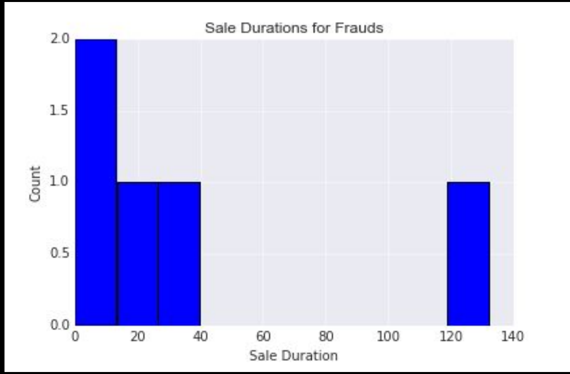
All Accounts





Fraudulent Accounts





Data Preprocessing

- Drop collinear features (eg. event dates/duration, body length/description)
- Drop features with > 10% null values (header, venue location details)
- Impute null values with mean for other features
- Extracted # previous payments from previous payouts feature

  Most frauds had 0 previous payments while others had 10+

- Included this in `munging.py` which created `.clean_data.csv`

Selected Features

delivery_method

sale_duration

previous_payouts

ticket_types

user_type

fb_published

channels

Model Selection

Random Forest Classifier

Gradient Boosting Classifier

Random Forest Classifier

	Predicted No	Predicted Fraud
True No	3265	16
True Fraud	27	277

Accuracy: 0.9880

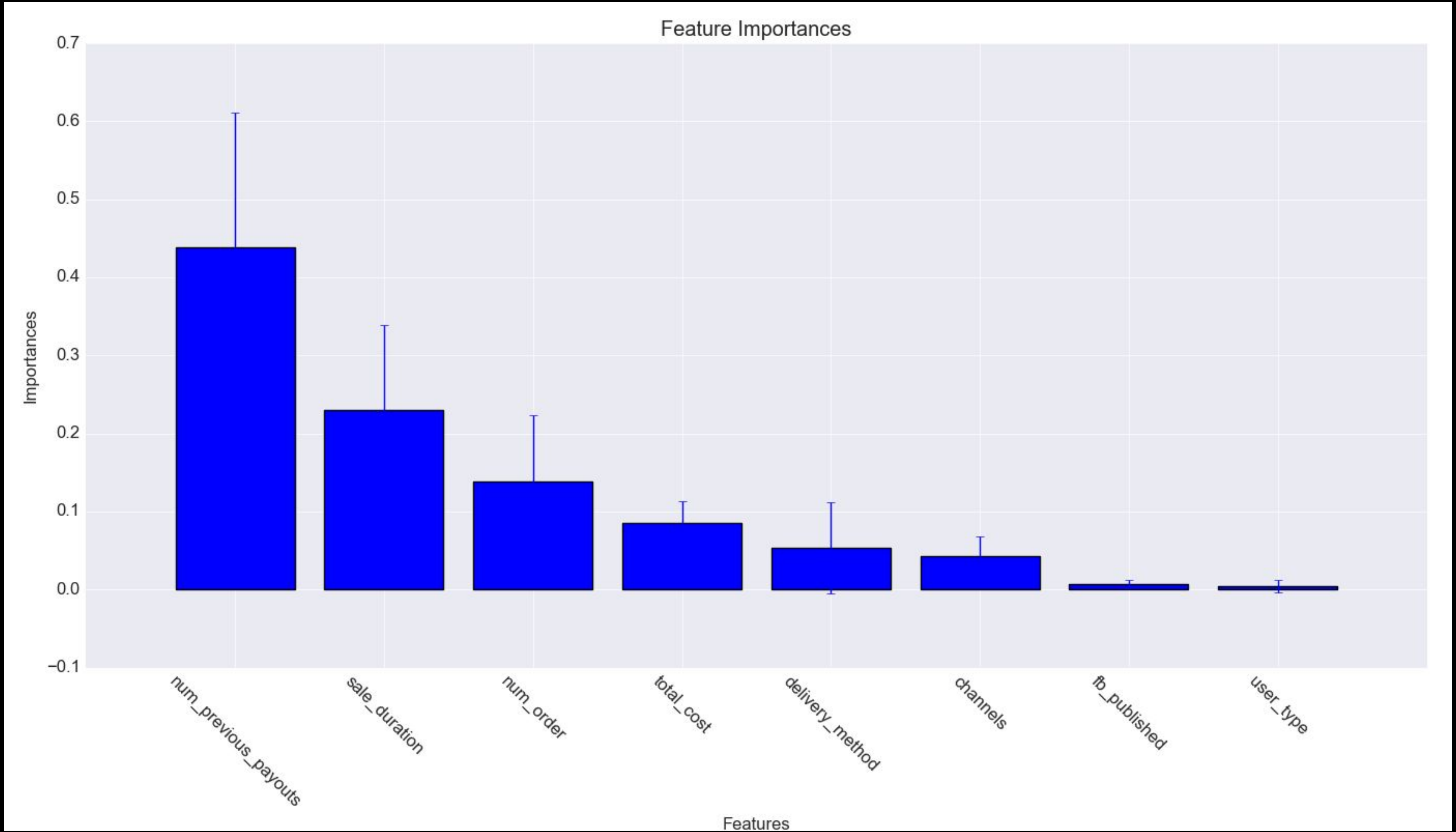
Recall: 0.9112

Gradient Boosting Classifier

	Predicted No	Predicted Fraud
True No	3207	58
True Fraud	48	272

Accuracy: 0.9704

Recall: 0.8500


Feature Importances
Importances
Features
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
num_previous_payouts
sale_duration
num_order
total_cost
delivery_method
channels
fb_published
user_type

Development Process

data.json → munging.py → clean.csv → model.py

munging_stream.py

model.pkl

app.py

data stream

database

Challenges

- Feature Engineering
- Time Constraints: waiting for POSTs
- Converting DATA stream to list of dictionaries

Demo