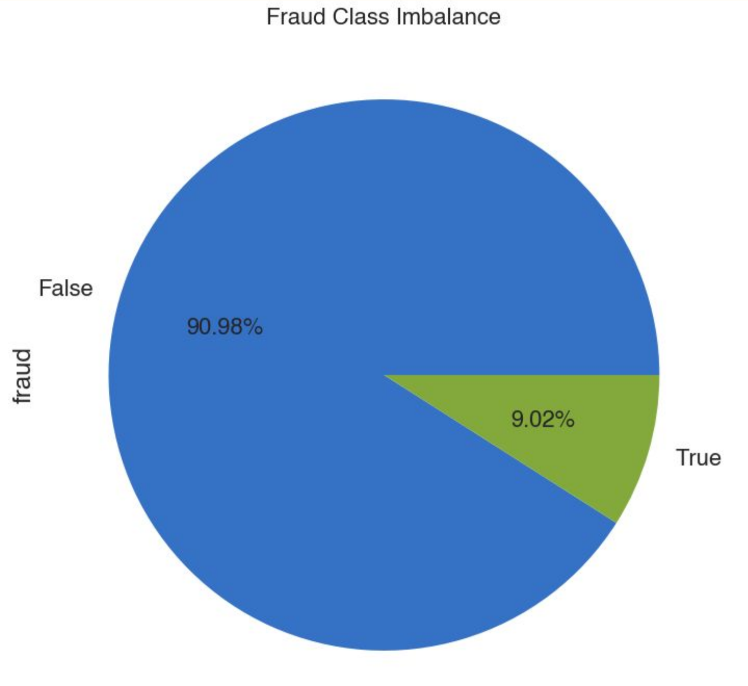
Detecting Fraud

Jiexi, Serhan, Takeshi, Chelsea

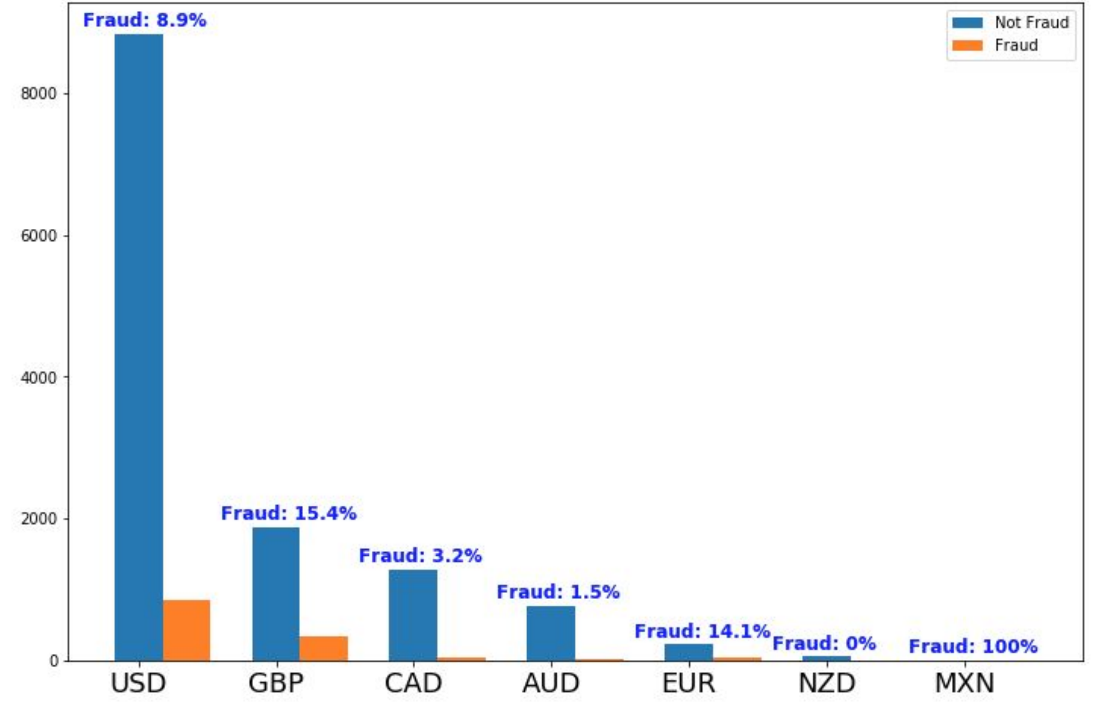
Dataset

- 14,337 rows, 44 features
    - **Numerical**: num_order, num_payouts
    - **Categorical**: country, currency
    - **Textual**: email_domain, description

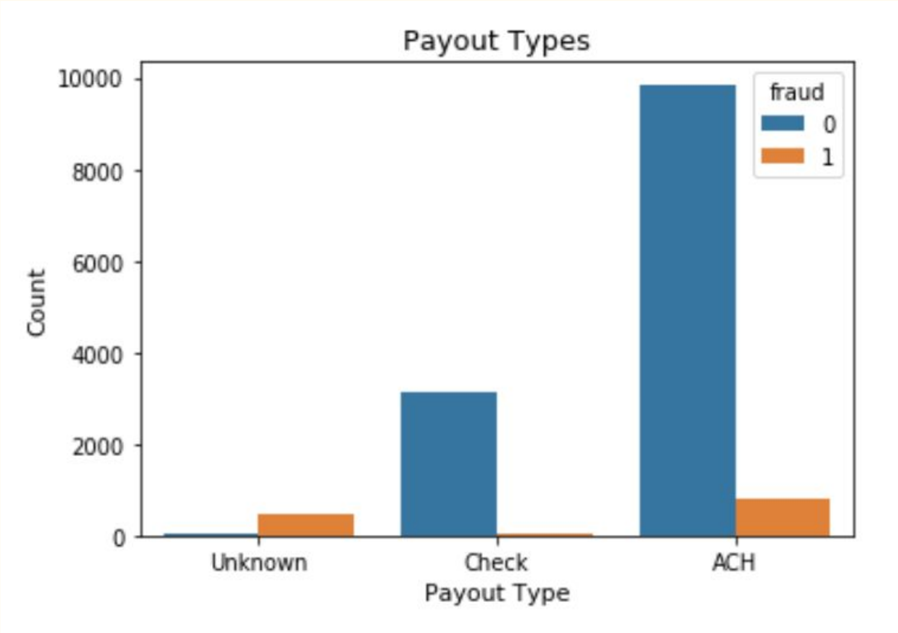
EDA



Currency



EDA


Payout Types
10000
8000
6000
4000
2000
0
Count
fraud
0
1
Unknown
Check
ACH
Payout Type


Sale Duration
7
6
5
4
3
2
1
0
log(sale_duration)
Not Fraud
Fraud
Is Fraud

Approach/Pipeline

- OneHotEncoder for categorical features
- Standardized numerical features
- Balanced training data by downsampling
- Built models on a selection of categorical & numerical features
- Removed leakage columns: num_order, num_payouts, sale_duration2
- Focused on optimizing recall score & roc-auc
    - Wanted to minimize false negatives

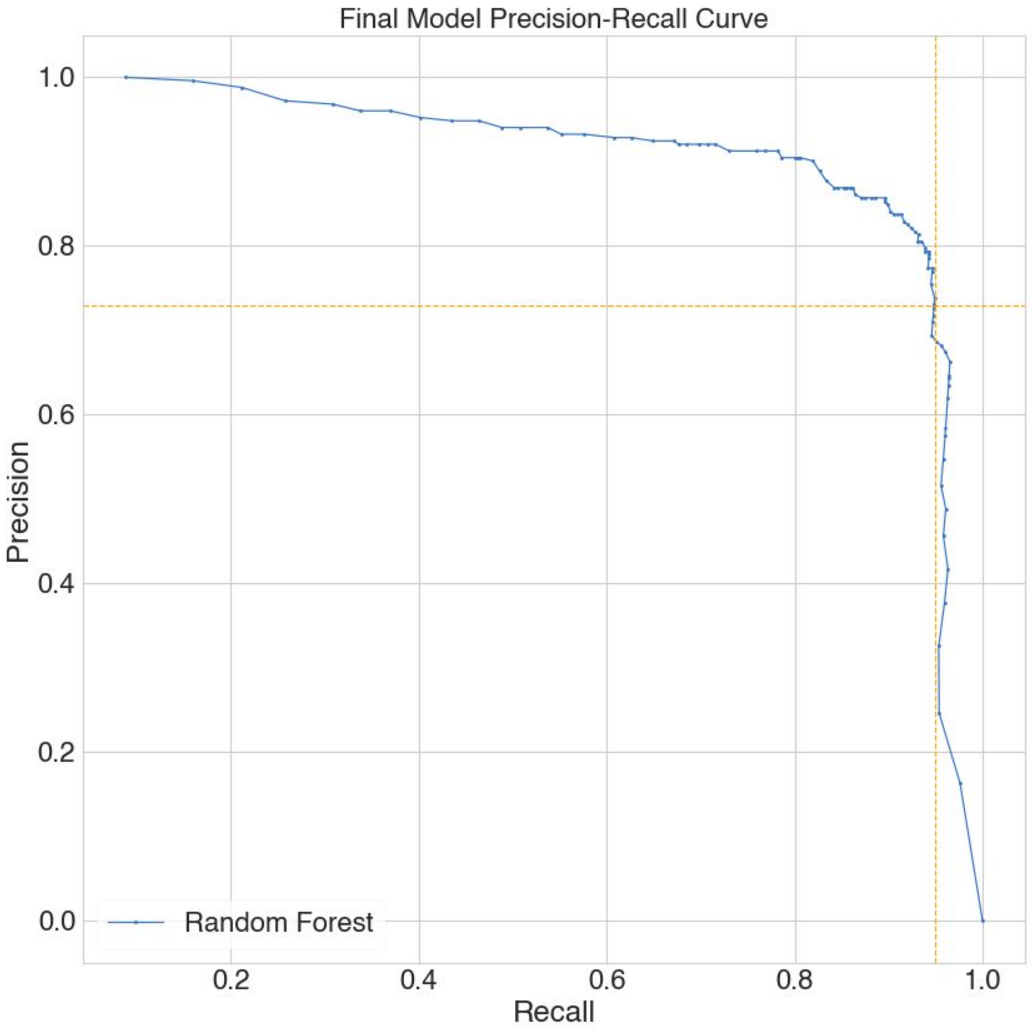
Model Comparisons: 5-Fold CV Scores

Model Type	Model Parameters/ Details	# Features	ROC AUC Score	Recall Score	Precision Score	Remarks
Logistic Regression	max_iter = 1000	48	0.9584	0.7726	0.8553	
SVC	probability=True		0.9788	0.8752	0.8966	
XGBoost	Default params		0.9848	0.8509	0.9189	
Gradient Boost			0.9841	0.854	0.9157	
kNN			0.9785	0.8615	0.8178	
Decision Tree			0.9809	0.9909	0.8792	
Random Forest			***0.9981***	***0.9919***	***0.9595***	***Best Scores***
Random Forest		37	0.9979	0.9919	0.9563	Removing features didn't help
Random Forest	"Tuned" on recall	48	0.9981	0.9914	0.9608	Tuning improved ROC AUC slightly, but decreased recall

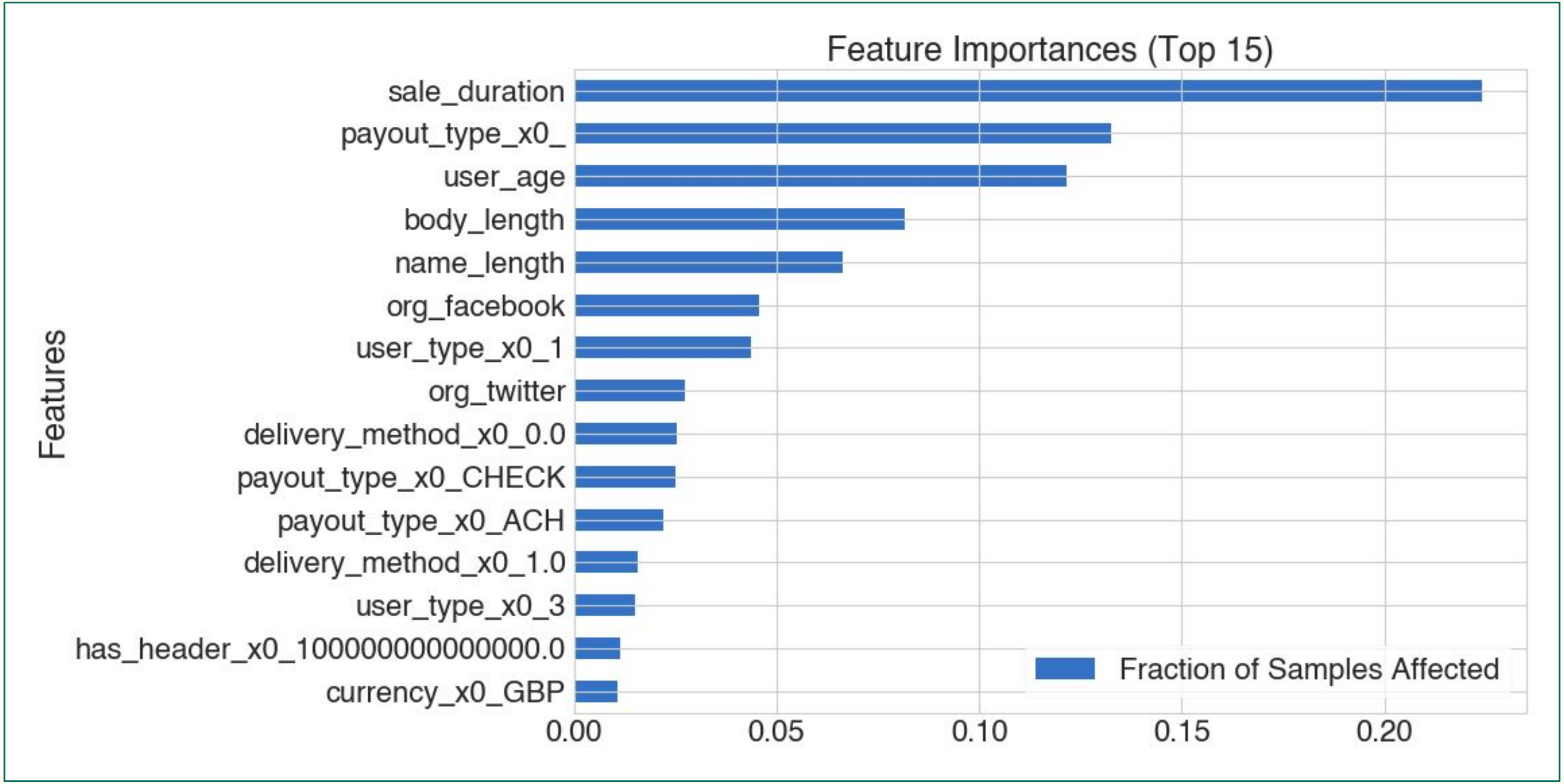
Final Model

- Random Forest
- Default parameters
  - Tuning did not improve recall
- 48 features (after OHE)
- Test ROC AUC Score: 0.9834

- Threshold: 0.2
- Test Recall Score: 0.92
- Test Precision Score: 0.72



Final Model Feature Importances



Business Actions

- Low Risk: < 0.2
  - events can carry on business as usual.
- Medium Risk: < 0.5
  - request additional information for verification, if we don't tell the customer we suspect them to be fraudulent, they won't lose trust in us.
- High Risk: $>= 0.5$
  - request additional information and seriously monitor these events.

Web App

- Built a web app using Flask that displays the results of our model on live data being pulled in real time

ec2-52-88-59-216.us-west-2.compute.amazonaws.com

AWS

#	Name	Probability	Risk
0	Decarbonising London, Energy Hubs and Boris's District Heating Manual.	0.26	medium
1	Marketing Coworking Day	0.0	low
2	MTV Filming Xandernation: Best Day Ever (Ft. KirillWasHere)	0.0	low
3	Juicing 101 with Black Girls Juice and Friends	0.01	low
4	Maggie's Place OH Magic of Motherhood	0.01	low
5	Marsh Social Group: Eloise Butler Wildflower Garden Tour and Mill Valley Kitchen	0.03	low
6	Mad-Hatter's May-Hem Masquerade Ball!	0.01	low
7	PRIDE Victorious After Party!	0.01	low
8	Simple Steps for Starting Your Business May 2013	0.0	low
9	TSH	0.58	high

Conclusions/Next Steps

- Overall, our detection task wasn’t too difficult, most of the fraudulent data had some obvious issues with it: like not specifying how they wanted to be paid out or being a newly created user.
- Left out a lot of Natural Language - type features
- Enhance interactions on the web app
- Give it a reserved IP Address

Any Questions?

Section to Dump Plots/Images

Appendix

5-Fold CV Scores Before Finding Data Leakage

Model Type	Model Parameters/ Details	# Features	ROC AUC Score	Recall Score	Remarks
Logistic Regression	max_iter = 1000		0.9654	0.7789	
XGBoost			0.9873	0.8526	
Gradient Boost			0.9875	0.8581	
kNN			0.9793	0.8822	
Decision Tree			0.9789	0.988	
Random Forest	Default params		**0.9976**	***0.9935***	***Best Score***
Random Forest	"Tuned" on recall		0.9971	0.99	Tuning didn’t help...
SVC	probability=True	51	0.9813	0.8802	