Ad Fraud – download prediction

Chris Lowe
Nazih Kalo
Markus Wehr

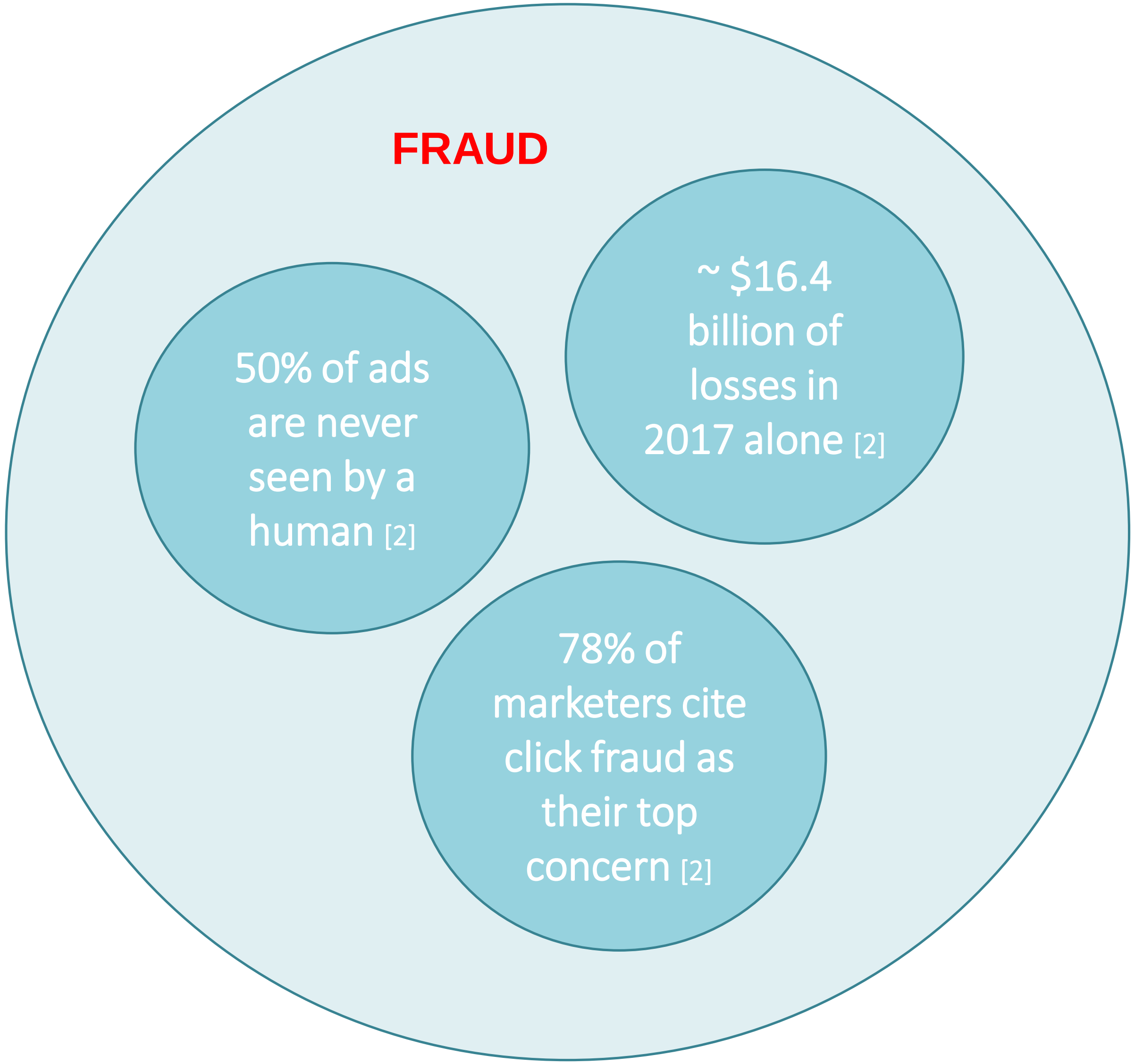
Outline

1. Business Problem
2. Dataset & Processing
3. Exploratory Data Analysis
4. Feature Engineering
5. Modeling
6. Conclusion & Future Work

1. Business Problem

Business Problem

- Advertisers pay websites/app providers by click/per download
- About **$280 billion** digital ad spending globally per year (2018) and growing [1]



Sources:
[1] https://www.emarketer.com/content/global-digital-ad-spending-2019
[2] https://medium.com/@aprofita_co/add-fraud-know-your-enemy-or-how-to-recognize-prevent-being-hacked-fc8caf19b1f2

Business Problem (cont.)

[1]

- TalkingData is China's largest independent big data platform
- Covers ~ 70% of active mobile devices nationwide
- Handle about 3 billion clicks per day, of which ~ 90% are potentially fraudulent

[2]

1. Fake Installs

- **Botnets:** Bots designed to impersonate user behavior
- **App Install Farms:** Low paid workers install apps through mobile ads

2. Fake Clicks

- Click Bots
- Click Farms
- Ghost Websites

3. Fake Impressions

- Hidden Adds
- Invisible Pixels
- Auto-Impression

Sources:
[1] https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection
[2] https://medium.com/@aprofita_co/add-fraud-know-your-enemy-or-how-to-recognize-prevent-being-hacked-fc8caf19b1f2

2. Dataset & Processing

Dataset & Processing

#	Feature	Description	Type	# categories	# complete rows
1	is_attributed	The target that is to be predicted, indicating if the app was downloaded	binary	2	184,903,890
2	ip	Ip address of click	categorical	277,396	184,903,890
3	app	App id for marketing	categorical	706	184,903,890
4	device	Type id of user mobile phone (e.g., iphone 7, huawei mate 7, etc.)	categorical	3475	184,903,890
5	os	Operating system version id of user mobile phone	categorical	800	184,903,890
6	channel	Channel id of mobile ad publisher	categorical	202	184,903,890
7	click_time	timestamp of click (UTC)	datetime	-	184,903,890

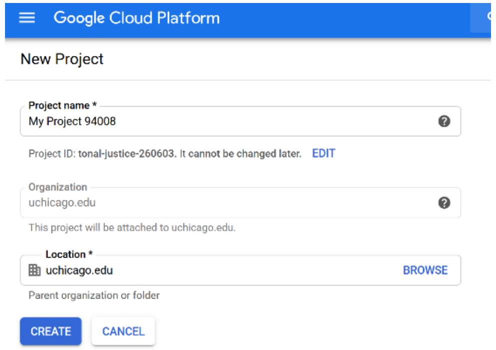
- Data collected over 4 days
- CSV (Train ~ 7.3GB | Test ~ 2.6GB)
- Changed datatypes to lower memory types (unint8, unint16, unint32)

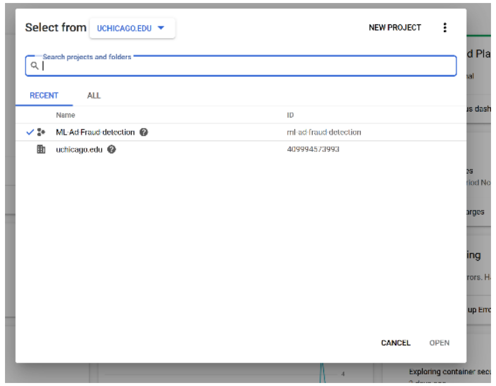
Too large for local processing!

Dataset & Processing (cont.)

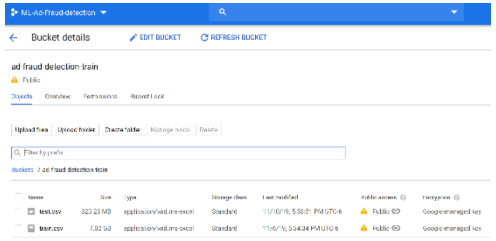
Using Google Cloud

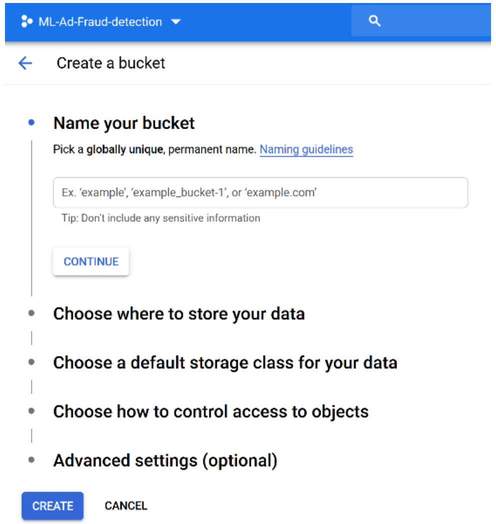
1. Creating a new project in the cloud



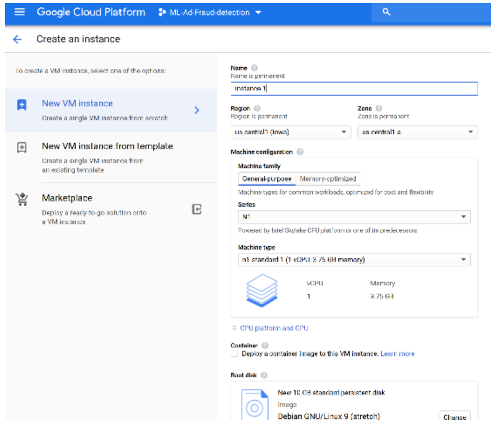


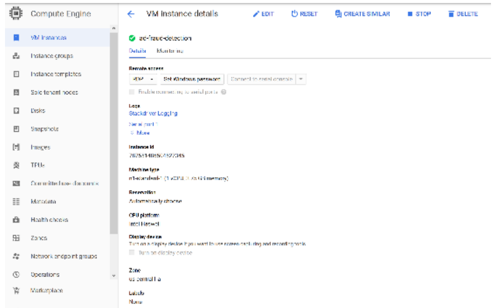
2. Creating bucket and loading data



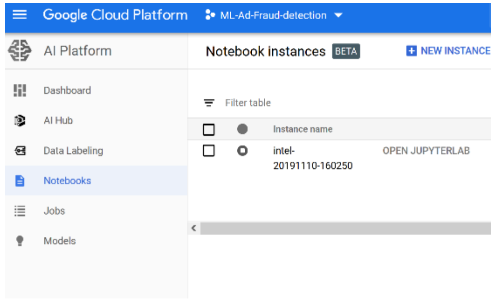


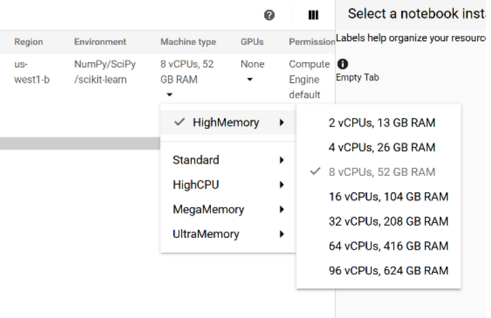
3. Creating virtual machine





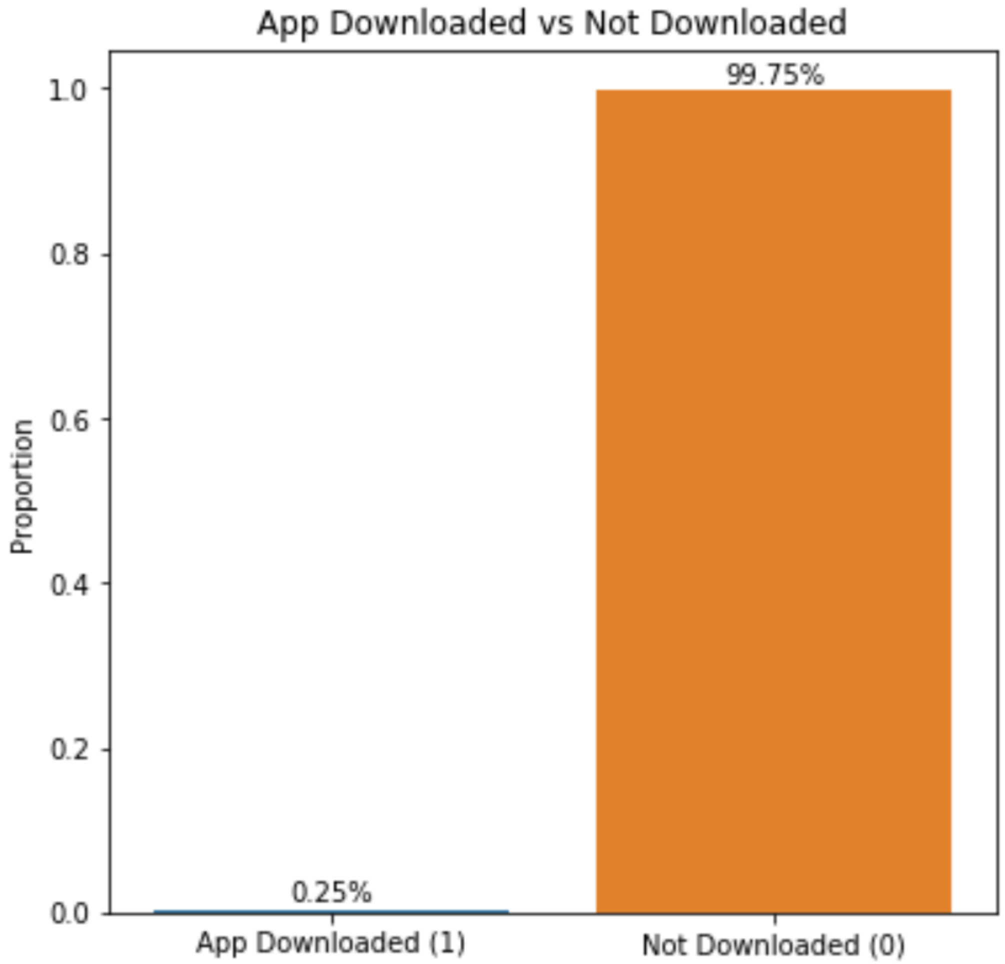
4. Creating notebook + selecting machine type





3. Exploratory Data Analysis

Exploratory Data Analysis



is_attributed:

- Target variable is highly imbalanced
- Only ~ 0.25% of clicks result in an actual download

Need to balance dataset!

Under-sampling

+ Reduce size of dataset
+ Computationally less expensive

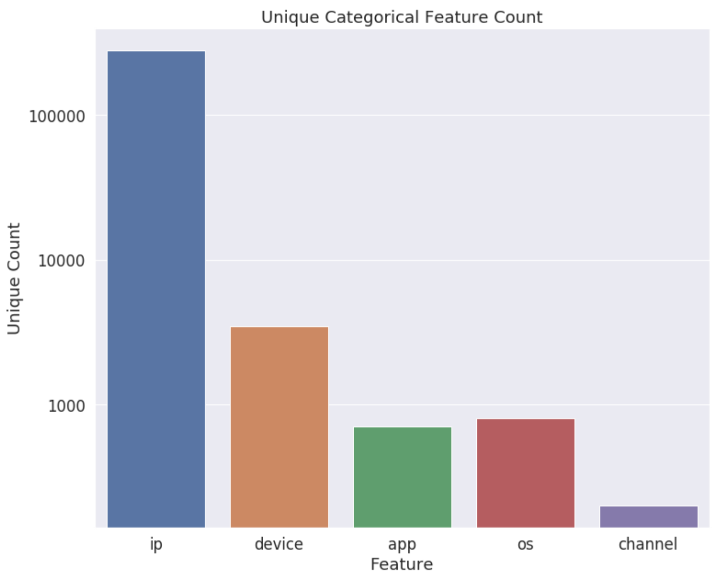
- Lose a lot of data

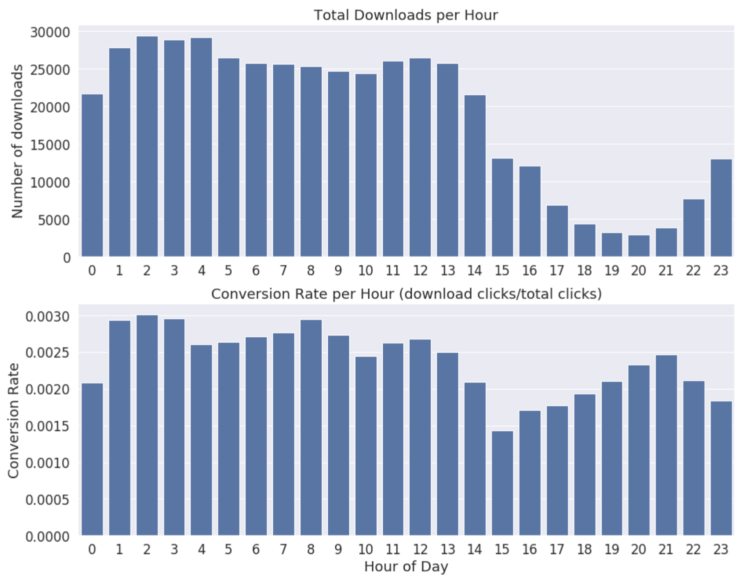
Over-sampling (SMOTE)

+ Do not loose data

- Increases size of dataset
- Synthetic datapoints
- Computationally very expensive

Exploratory Data Analysis (cont.)

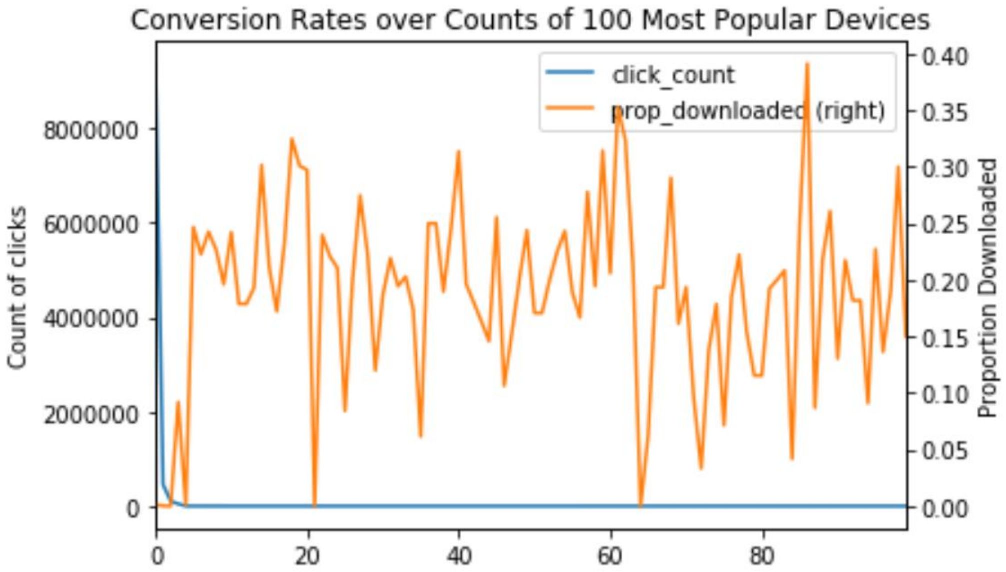


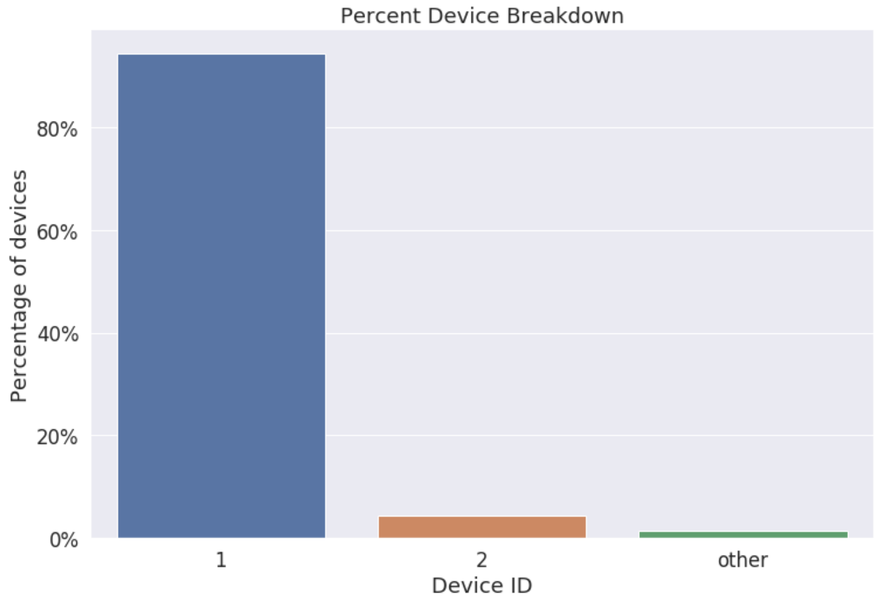


- By far most categories for ip (~ 2x device and ~ 4-5x app/os)
- Least categories for channel
- Grouping of most categories not possible (only IDs)

- Conversion rate = $\dfrac{downloads_t}{total\ clicks_t}$
- Total # of downloads goes down in the evening
- Conversion rate is relatively stable distributed (almost uniform)

Exploratory Data Analysis (cont.)



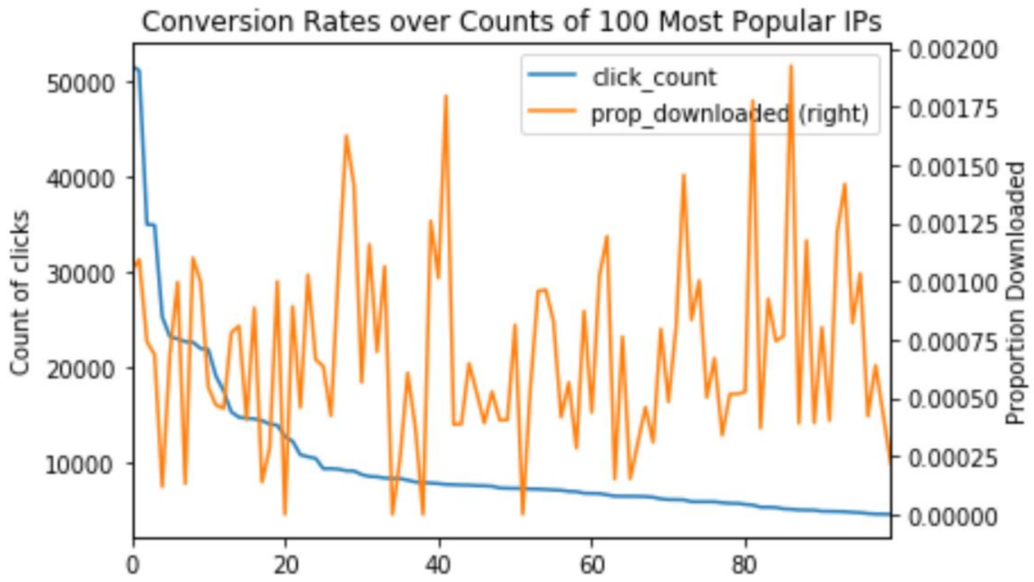


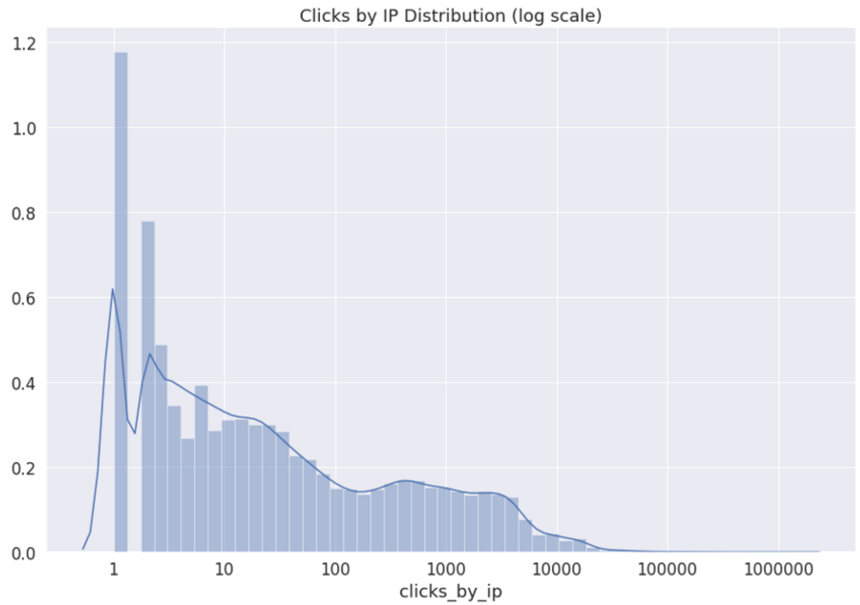
device:

- Conversion rate for 100 most popular devices (by click count) is similarly distributed, except for device 1
- Device 1 has the most click and the lowest conversion rate, therefore, most fraudulent clicks are within that group

Group devices other than 1 or 2 into group 'other'!

Exploratory Data Analysis (cont.)

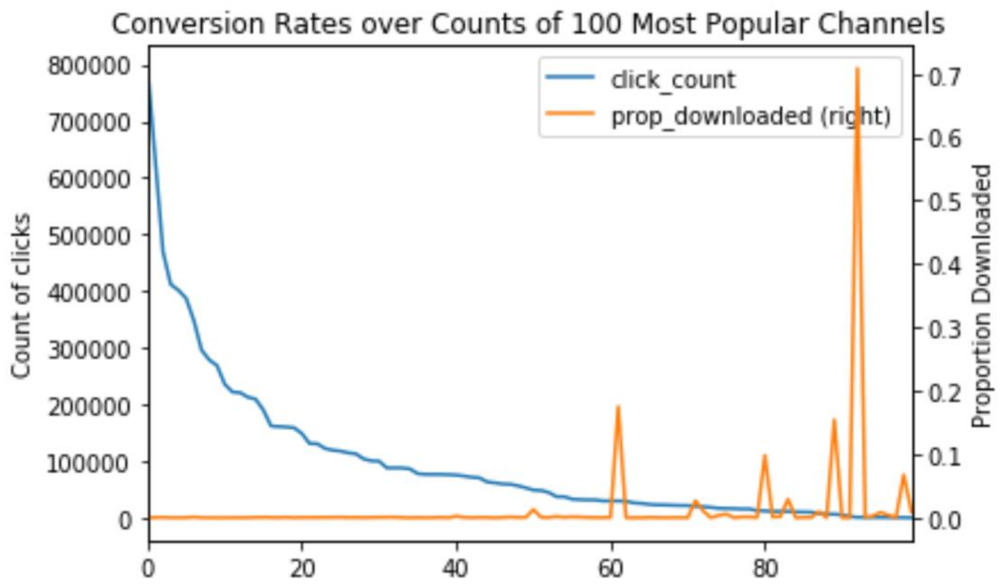


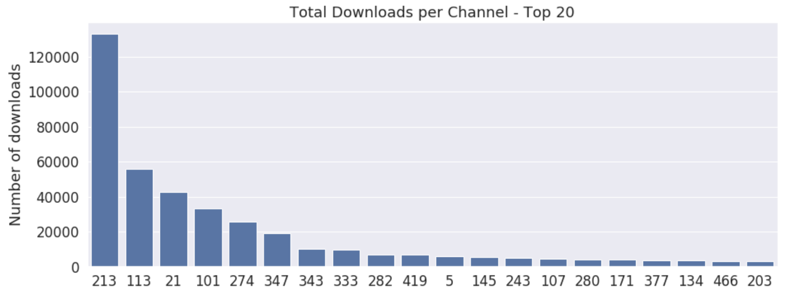


ip:

- Conversion rate is not dependent on total number of clicks per ip, according to the 100 most popular ips (by click)
- About 10,000 different ips with a very high number of clicks, while the rest is relatively insignificant

Exploratory Data Analysis (cont.)

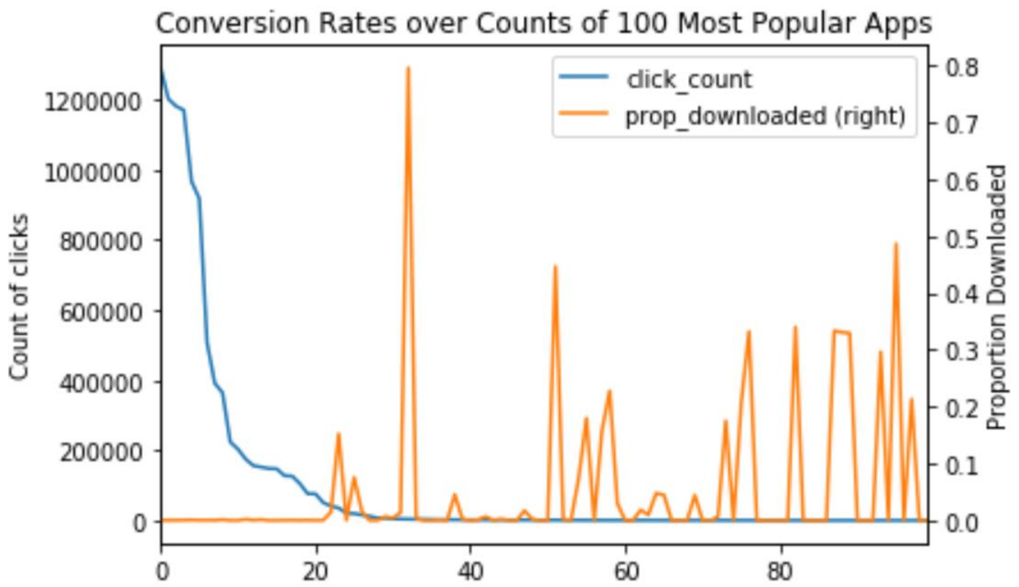


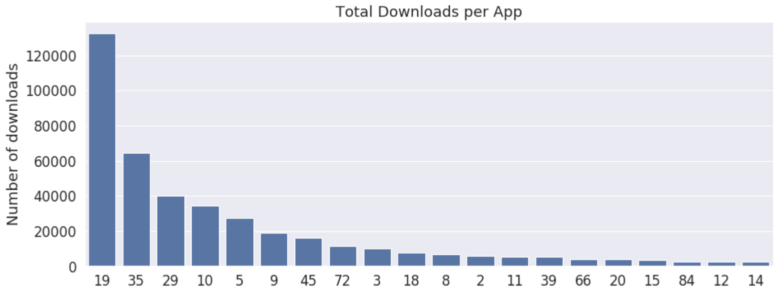


channel:

- Conversion rate is significantly lower for channels with a high absolute number of clicks
- For top 20 channels (by number of clicks), conversion rates differ a lot
- Channels with lower conversion rates and a high absolute number of clicks might be a sign for potential fraud

Exploratory Data Analysis (cont.)

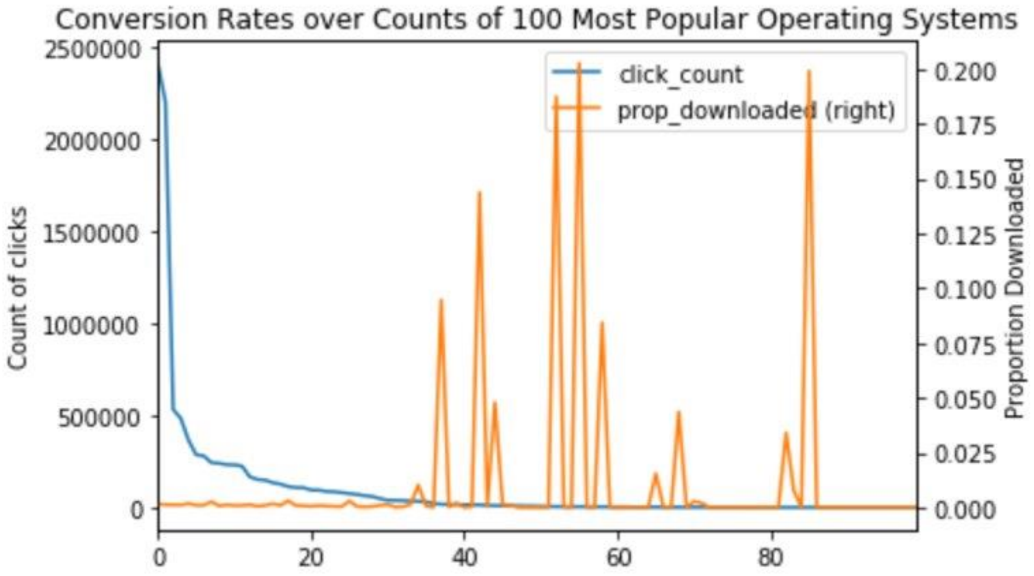


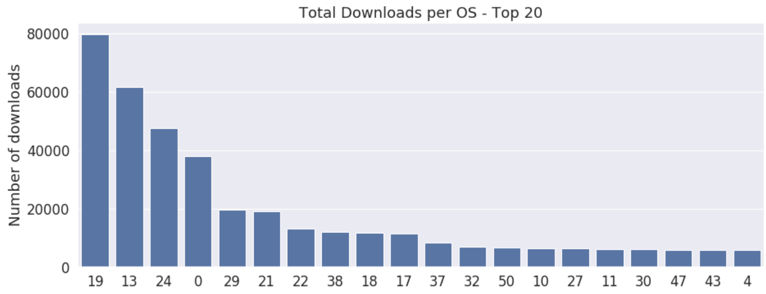


app:

- Conversion rate is significantly lower for top 20 apps (by click count)
- For top 20 apps (by number of clicks), conversion rates differ a lot
- Apps with lower conversion rates and a high absolute number of clicks might be a sign for potential fraud

Exploratory Data Analysis (cont.)





os:

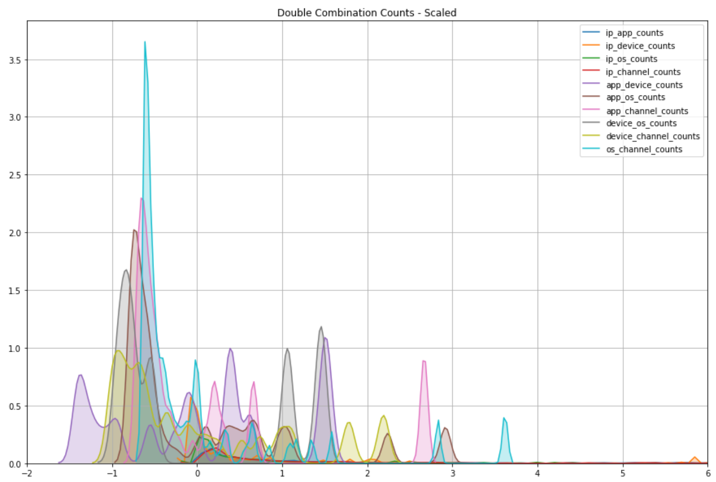
- Conversion rate is significantly lower for top ~ 30 operating systems (by click count)
- For top 20 os (by number of clicks), conversion rates differ a lot
- Os with lower conversion rates and a high absolute number of clicks might be a sign for potential fraud

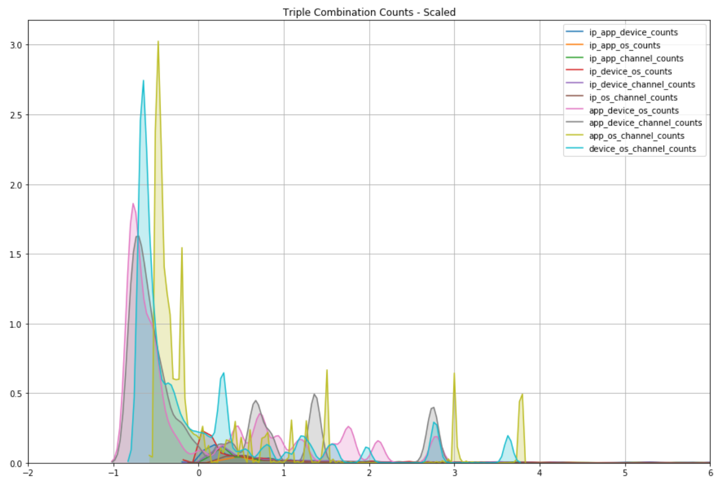
4. Feature Engineering

Feature Engineering

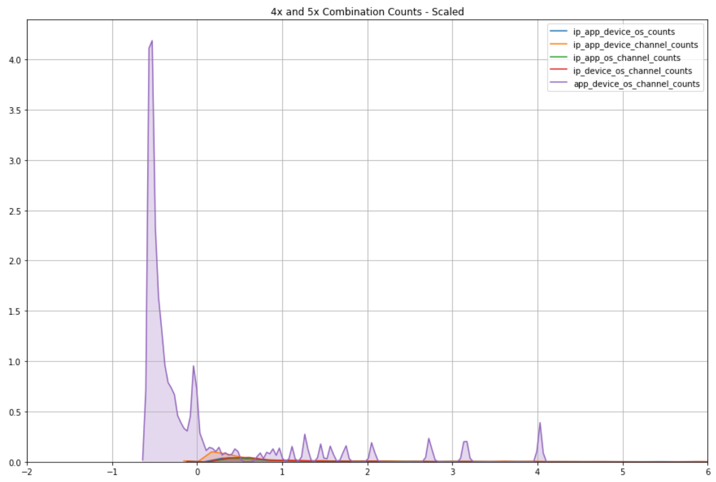
Combination counts

- Each 'click' contains data about the 'clicker' – IP address, type of device, type of OS, etc.
- To better understand the 'clicker' attribute interaction, we decided to use value counts along with combinatorics to identify any potentially recurring 'clickers.'





- These features were extracted by using a series of for loops to create the correct combinatorics sequence, followed by a pandas.Series.value_counts() function to count the number of times a combination occurred.
- After scaling, this feature provides a 'weight' for how often the same combination occurs



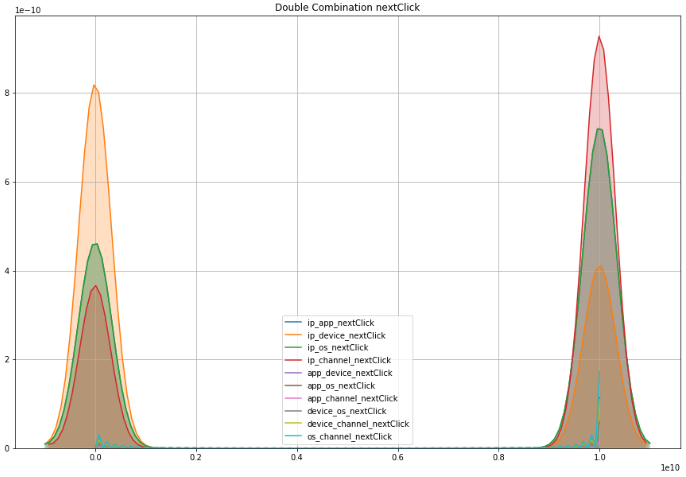
- Kernel density estimation plots of the various combinatorics

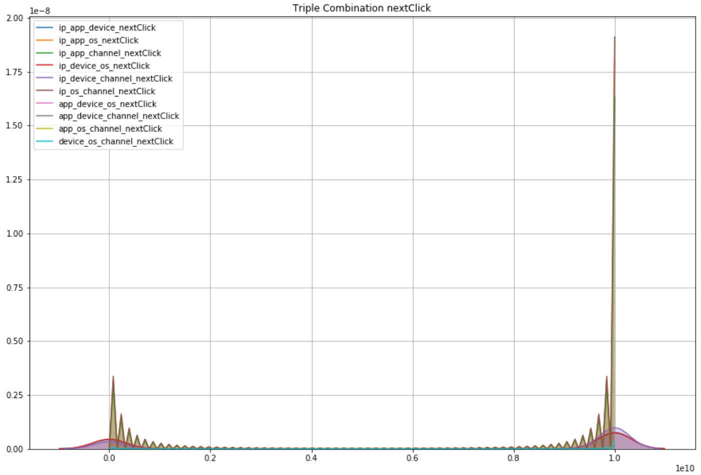
Feature Engineering (cont.)

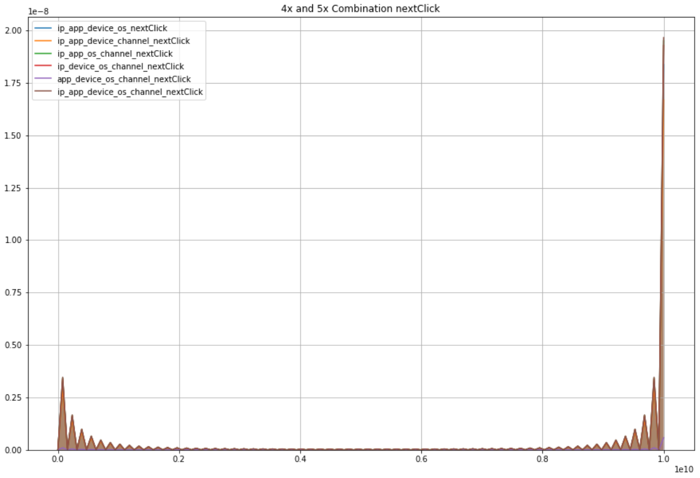
Time to next click

- Time to next click looks at the timing interval between different combinations of clickers.
- For example, if a specific device and IP combination clicked on an ad, how long before that same device and IP combination clicked again?

- Using combinatorics and a series of for loops, this process was carried out for double, triple, quadruple, and quintuple combinations.
- The NaN values were replaced with a very high value – 1e10 to ‘filter’ out click patterns that did not recur
- This feature searches for timed patterns – programs that potentially click an ad repeatedly every given step of time.



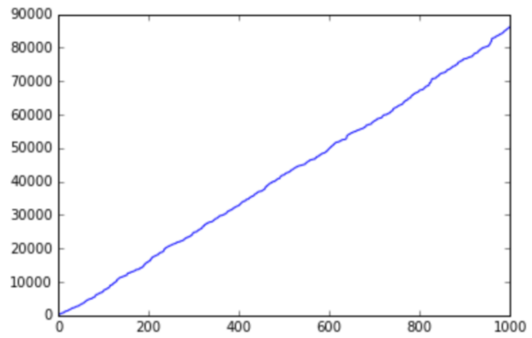


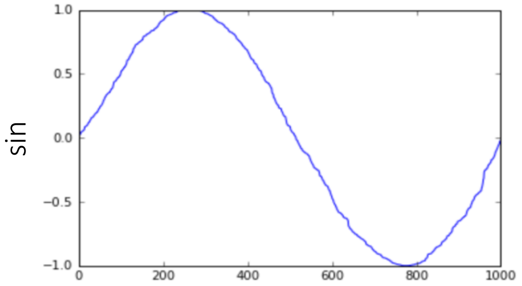


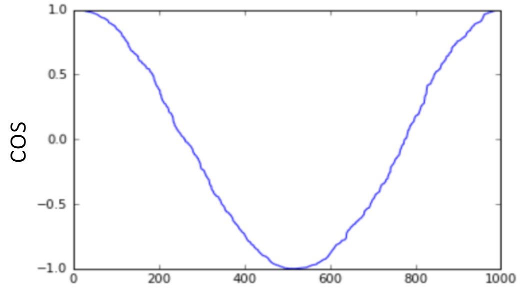
- Kernel density estimation plots of the various combinatorics

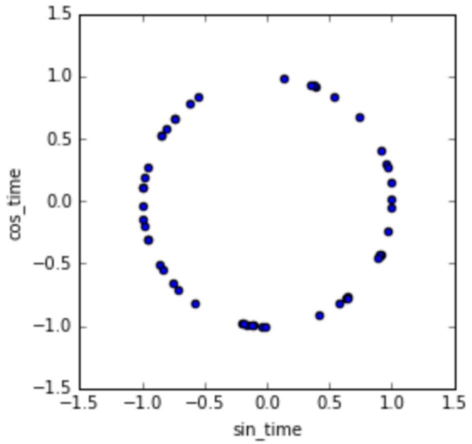
Feature Engineering (cont.)

Encoding cyclical continuous features – Day and hour









- Some features are cyclical (e.g., **days, hours** etc.)
- Without transformation, cyclical nature is not conveyed

Encode as cyclical feature!

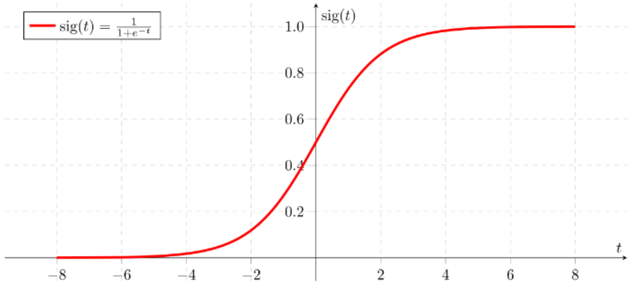
- Deriving a sine transform and cosine transform for days and hours respectively (2 new columns each)
- Both transformations needed to avoid side effects

- Using the two features together, all times can be distinguished from each other
- Difference corresponds to expected difference in time

5. Modeling

Modeling

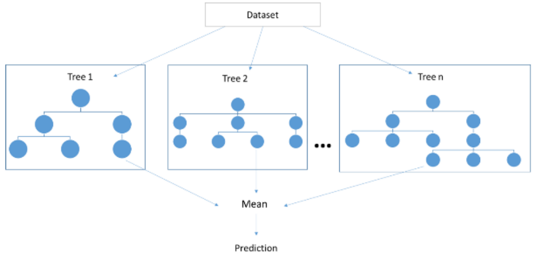
Binary Logistic Regression



Advantages
- Efficient & easy
- Highly interpretable
- Multicollinearity somewhat handled with L2 (Ridge)

Disadvantages
- No large feature spaces
- Does not handle many categorical features well
- transformation for non-linear relations needed

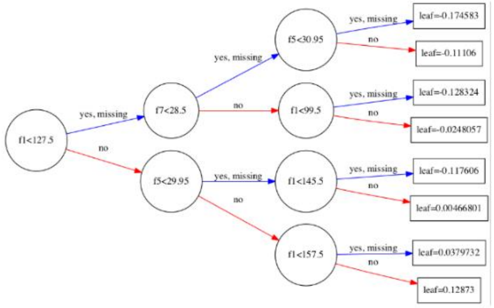
Random Forest



Advantages
- Performs well with noisy data
- Reduces overfitting in DTs
- Handles continuous and categorical features
- Handles missing values
- Robust to outliers

Disadvantages
- More complex and computationally expensive
- Greedy (prone to overfitting)

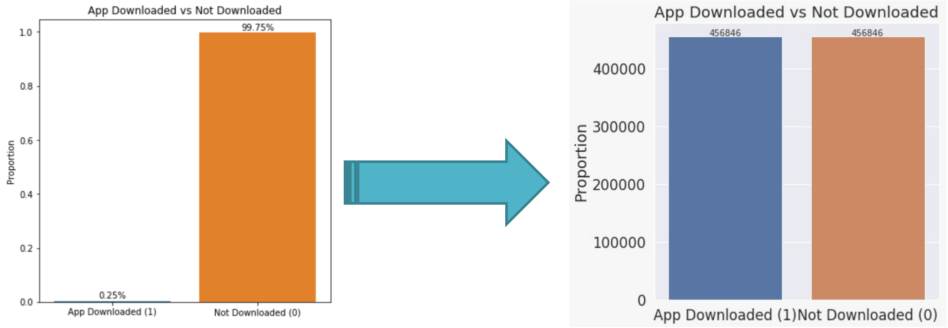
XGBoost



Advantages
- Improves weak learners
- Performs well on imbalanced data
- Built in Regularization
- Parallel processing makes it faster
- Handles missing values
- Removes splits that are not above threshold gain

Disadvantages
- Easily overfits with noisy data
- Hard to tune

Modeling (cont.)



Baseline Model Performance

Baseline RF X_train Prediction:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	129112931
1	0.82	0.33	0.47	319792
avg / total	1.00	1.00	1.00	129432723

Baseline RF X_test Prediction:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	55334113
1	0.81	0.33	0.47	137054
avg / total	1.00	1.00	1.00	55471167

Baseline LR X_train Prediction:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	129112931
1	0.09	0.00	0.00	319792
accuracy			1.00	129432723
macro avg	0.55	0.50	0.50	129432723
weighted avg	1.00	1.00	1.00	129432723

Baseline LR X_test Prediction:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	55334113
1	0.11	0.00	0.00	137054
accuracy			1.00	55471167
macro avg	0.56	0.50	0.50	55471167
weighted avg	1.00	1.00	1.00	55471167

Random Forest Baseline ROC_AUC Reports:

Baseline RF X_train Prediction:
0.6665695015675492
Baseline RF X_test Prediction:
0.6655235617319419

Logistic Regression Baseline ROC_AUC Reports:

Baseline LR X_train Prediction:
0.5007386608117844
Baseline LR X_test Prediction:
0.5009016531375848

- Overall baseline performance has difficulty with precision and recall of the 'is_attributed' prediction. This appears to be due to an anomaly detection task.
- Random Forest performs noticeably better than Logistic Regression.

Model Performance after Random Undersampling

Post-RUS RF X_train Prediction:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	319792
1	1.00	0.98	0.99	319792
accuracy			0.99	639584
macro avg	0.99	0.99	0.99	639584
weighted avg	0.99	0.99	0.99	639584

Post-RUS RF X_test Prediction:

	precision	recall	f1-score	support
0	0.89	0.96	0.92	137054
1	0.95	0.88	0.91	137054
accuracy			0.92	274108
macro avg	0.92	0.92	0.92	274108
weighted avg	0.92	0.92	0.92	274108

Post-RUS LR X_train Prediction:

	precision	recall	f1-score	support
0	0.75	0.76	0.75	319792
1	0.75	0.74	0.75	319792
accuracy			0.75	639584
macro avg	0.75	0.75	0.75	639584
weighted avg	0.75	0.75	0.75	639584

Post-RUS LR X_test Prediction:

	precision	recall	f1-score	support
0	0.75	0.75	0.75	137054
1	0.75	0.75	0.75	137054
accuracy			0.75	274108
macro avg	0.75	0.75	0.75	274108
weighted avg	0.75	0.75	0.75	274108

Random Forest Post-RUS ROC_AUC Reports:

Post-RUS RF X_train Prediction:
0.9912318006704357
Post-RUS RF X_test Prediction:
0.9158433901965649

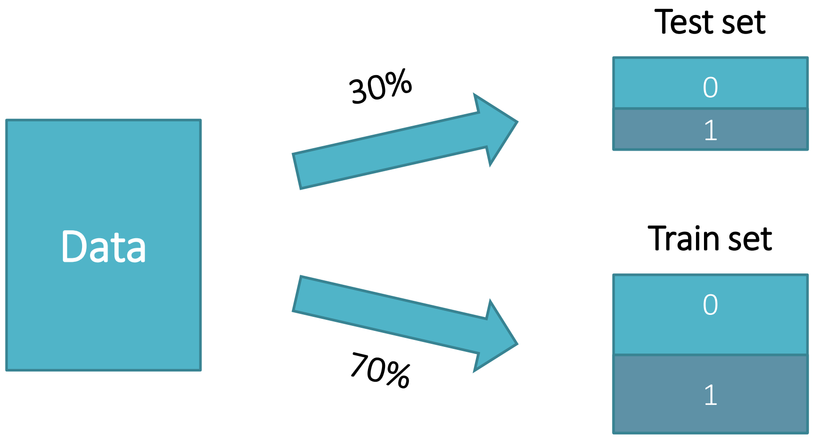
Logistic Regression Post ROC_AUC Reports:

Post-RUS LR X_train Prediction:
0.7502095111822685
Post-RUS LR X_test Prediction:
0.7502918557648809

- Random Undersampling drastically improves both the Random Forest and Logistic Regression models, while also improving the size of the dataset so it is more manageable.
- Oversampling (SMOTE) was also tried and worked well on a subsample, however it is a bad choice for such a large dataset.

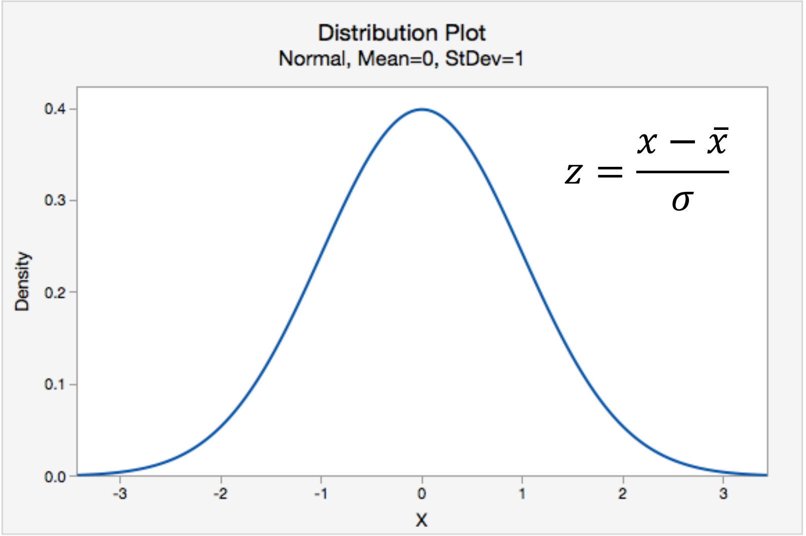
Modeling (cont.)

Stratified train and test sampling



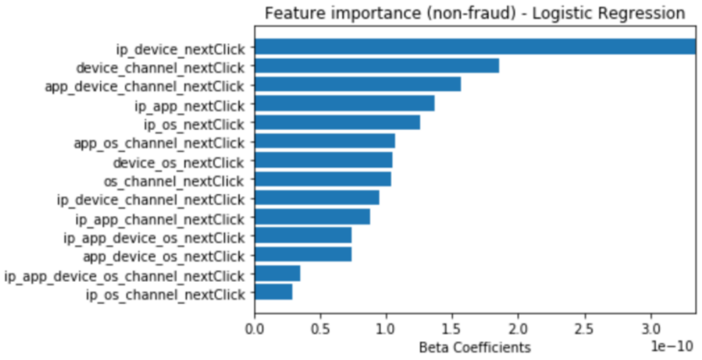
- Stratified sampling by **is_attributed** (target variable) with a 70/30 train-test split
- 50/50 distribution of downloads and non-downloads in train and test set

Scaling



- Standardized count features by removing the mean and scaling to unit variance
- Zero mean and unit variance
- Necessary especially for distance-based models

Modeling (cont.)



	feature	importance	cum_sum	cum_perc
32	ip_device_nextClick	3.33977e-10	3.33977e-10	20.1991
39	device_channel_nextClick	1.859e-10	5.19876e-10	31.4425
48	app_device_channel_nextClick	1.56798e-10	6.76674e-10	40.9257
31	ip_app_nextClick	1.36976e-10	8.1365e-10	49.2101
33	ip_os_nextClick	1.25708e-10	9.39359e-10	56.813

	feature	importance	cum_sum	cum_perc
38	device_os_nextClick	0.189665	0.189665	18.9665
36	app_os_nextClick	0.171941	0.361606	36.1606
47	app_device_os_nextClick	0.153709	0.515315	51.5315
37	app_channel_nextClick	0.1101	0.625415	62.5415
35	app_device_nextClick	0.108508	0.733923	73.3923

Dropped 14 features

- Based on Beta coefficients and Information Gain, Next Click features appear to have the most impact on the model prediction
- Ranked the features by IG and Beta Coefficients respectively and calculated cumulative percentage
- Dropped all features that are **not within the first 95%** for both Random Forest and Logit
- Additionally, the time of download was dropped previously to prevent leakage, although this might have had a high IG/Coefficient

Modeling (cont.)

Binary Logistic Regression

```
Classification report: Train set
              precision    recall  f1-score   support

    No Fraud       0.77      0.83      0.80    319792
       Fraud       0.81      0.75      0.78    319792

   micro avg       0.79      0.79      0.79    639584
   macro avg       0.79      0.79      0.79    639584
weighted avg       0.79      0.79      0.79    639584
 ##########################################################
Classification report: Test set
              precision    recall  f1-score   support

    No Fraud       0.77      0.83      0.80    137054
       Fraud       0.81      0.75      0.78    137054

   micro avg       0.79      0.79      0.79    274108
   macro avg       0.79      0.79      0.79    274108
weighted avg       0.79      0.79      0.79    274108
 ##########################################################
Avg. f1 score: cross-validation set
 0.78
```

- Results are very balanced between train, test, and cross-validation set
- Precision higher than recall, so there is more emphasis on capturing all downloads in this model

Random Forest

```
Classification report: Train set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    319792
       Fraud       1.00      1.00      1.00    319792

   micro avg       1.00      1.00      1.00    639584
   macro avg       1.00      1.00      1.00    639584
weighted avg       1.00      1.00      1.00    639584
 ##########################################################
Classification report: Test set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    137054
       Fraud       1.00      1.00      1.00    137054

   micro avg       1.00      1.00      1.00    274108
   macro avg       1.00      1.00      1.00    274108
weighted avg       1.00      1.00      1.00    274108
 ##########################################################
Avg. f1 score: cross-validation set
 1.0
```

- The model is clearly overfit
- Balancing the data previously actually introduced a bias to the model towards predicting more downloads than there actually are [1]

XGBoost

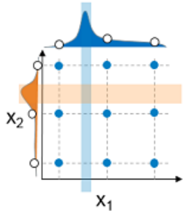
```
Classification report: Train set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    319792
       Fraud       1.00      1.00      1.00    319792

   micro avg       1.00      1.00      1.00    639584
   macro avg       1.00      1.00      1.00    639584
weighted avg       1.00      1.00      1.00    639584
 ##########################################################
Classification report: Test set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    137054
       Fraud       1.00      1.00      1.00    137054

   micro avg       1.00      1.00      1.00    274108
   macro avg       1.00      1.00      1.00    274108
weighted avg       1.00      1.00      1.00    274108
 ##########################################################
Avg. f1 score: cross-validation set
 1.0
```

- XGBoost seems to be overfit as well
- Again, undersampling seems to have introduced a bias to the models

Source: [1] Dal Pozzolo et al. (2015): Calibrating Probability with Undersampling for Unbalanced Classification

[1] Plus compared L1 (Lasso) vs L2 (Ridge) regularization.

Modeling (cont.)



Standard Grid Search

Model	# of parameters	# of fits
Logistic Regression	1 [1]	30
Random Forest	4	243
XGBoost	3	54

Binary Logistic Regression

```
Classification report: Train set
              precision    recall  f1-score   support

    No Fraud       0.78      0.82      0.80    319792
       Fraud       0.81      0.77      0.79    319792

    accuracy                           0.80    639584
   macro avg       0.80      0.80      0.80    639584
weighted avg       0.80      0.80      0.80    639584
#########################################################
Classification report: Test set
              precision    recall  f1-score   support

    No Fraud       0.78      0.82      0.80    137054
       Fraud       0.81      0.77      0.79    137054

    accuracy                           0.80    274108
   macro avg       0.80      0.80      0.80    274108
weighted avg       0.80      0.80      0.80    274108
#########################################################
Avg. f1 score: cross-validation set
 0.79
```

Random Forest

```
Classification report: Train set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    319792
       Fraud       1.00      1.00      1.00    319792

    accuracy                           1.00    639584
   macro avg       1.00      1.00      1.00    639584
weighted avg       1.00      1.00      1.00    639584
#########################################################
Classification report: Test set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    137054
       Fraud       1.00      1.00      1.00    137054

    accuracy                           1.00    274108
   macro avg       1.00      1.00      1.00    274108
weighted avg       1.00      1.00      1.00    274108
#########################################################
```

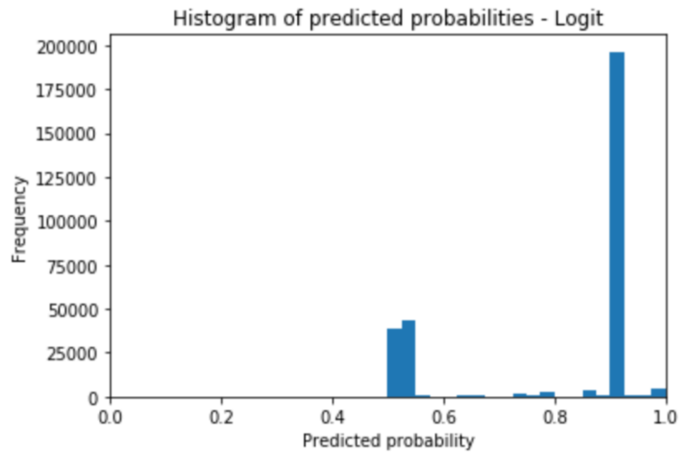
XGBoost

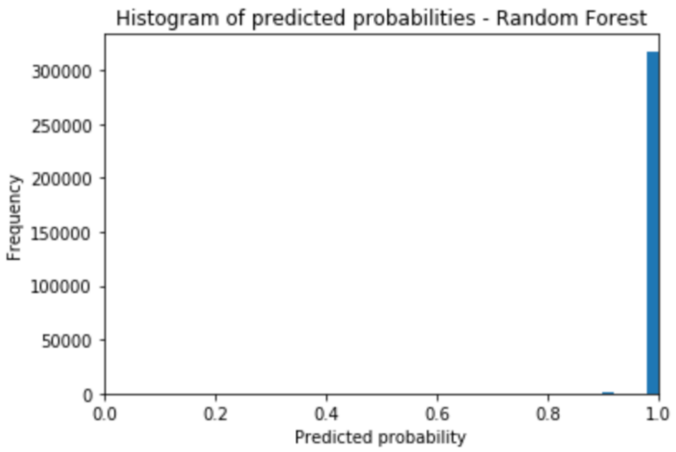
```
Classification report: Train set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    319792
       Fraud       1.00      1.00      1.00    319792

    accuracy                           1.00    639584
   macro avg       1.00      1.00      1.00    639584
weighted avg       1.00      1.00      1.00    639584
#########################################################
Classification report: Test set
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    137054
       Fraud       1.00      1.00      1.00    137054

    accuracy                           1.00    274108
   macro avg       1.00      1.00      1.00    274108
weighted avg       1.00      1.00      1.00    274108
#########################################################
Avg. f1 score: cross-validation set
 1.0
```

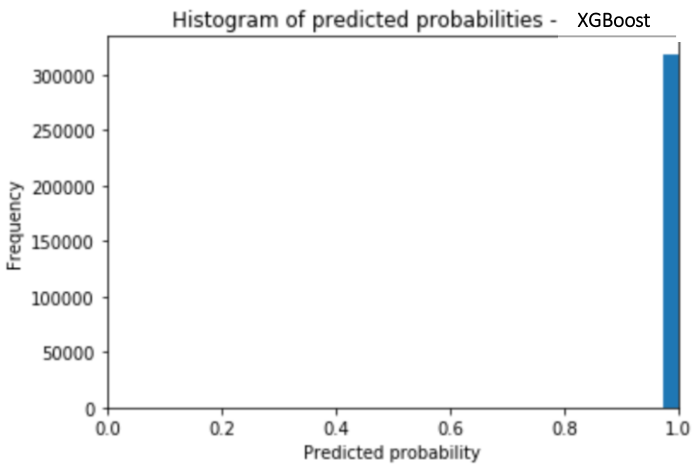
- Model predictions for logistic regression improved by ~1% based on F1-score
- Regularization did not change results for Random Forest and XGBoost
- Using XGBoost as model for final predictions because of its better generalizability on imbalanced dataset (expecting Holdout set to be highly imbalanced)

Modeling (cont.)

- Since we are expecting the hold out set to be similarly imbalanced as the training set, our models will most likely deliver biased results
- Among others, Dal Pozzolo et al. (2015) propose a solution by changing the threshold post downsampling and modeling (e.g., by using Bayesian approaches)
- $p(y|x, real) \neq p(y|x, undersampled)$, therefore, threshold needs to be adjusted to real probability distribution again
- For this project, we are looking at the probabilities predicted (for class 1) and set the thresholds accordingly when applying the model on the Hold Out set (naïve approach)







Source: [1] Dal Pozzolo et al. (2015): Calibrating Probability with Undersampling for Unbalanced Classification

6. Conclusion

Conclusion & Future Work

Conclusion:

- Anomaly detection problems require several methods to correctly classify binary or multiclass problems.
- Feature Engineering was important to help improve the performance of the model based on AUC. In situations with very large datasets, appropriate feature selection is important to consider in data processing.
- Sampling is a good solution for helping to deal with very unbalanced datasets to remove majority/minority bias. However, sampling also leads to apriori and aposterior probability differences due real data vs. manufactured data that affects test set model performance.

Future work:

- For future work, we would like to investigate more sophisticated models of thresholding to counteract the overfitting that was produced during random undersampling. There are several cases applying Bayes' theorem to reduce the Sample Selection Bias.

Thank you!

- Questions?