TalkingData AdTracking Fraud Detection Challenge

Notebook here:
https://github.com/lstmemery/talking_data_presentation
(https://github.com/lstmemery/talking_data_presentation)

Who Am I

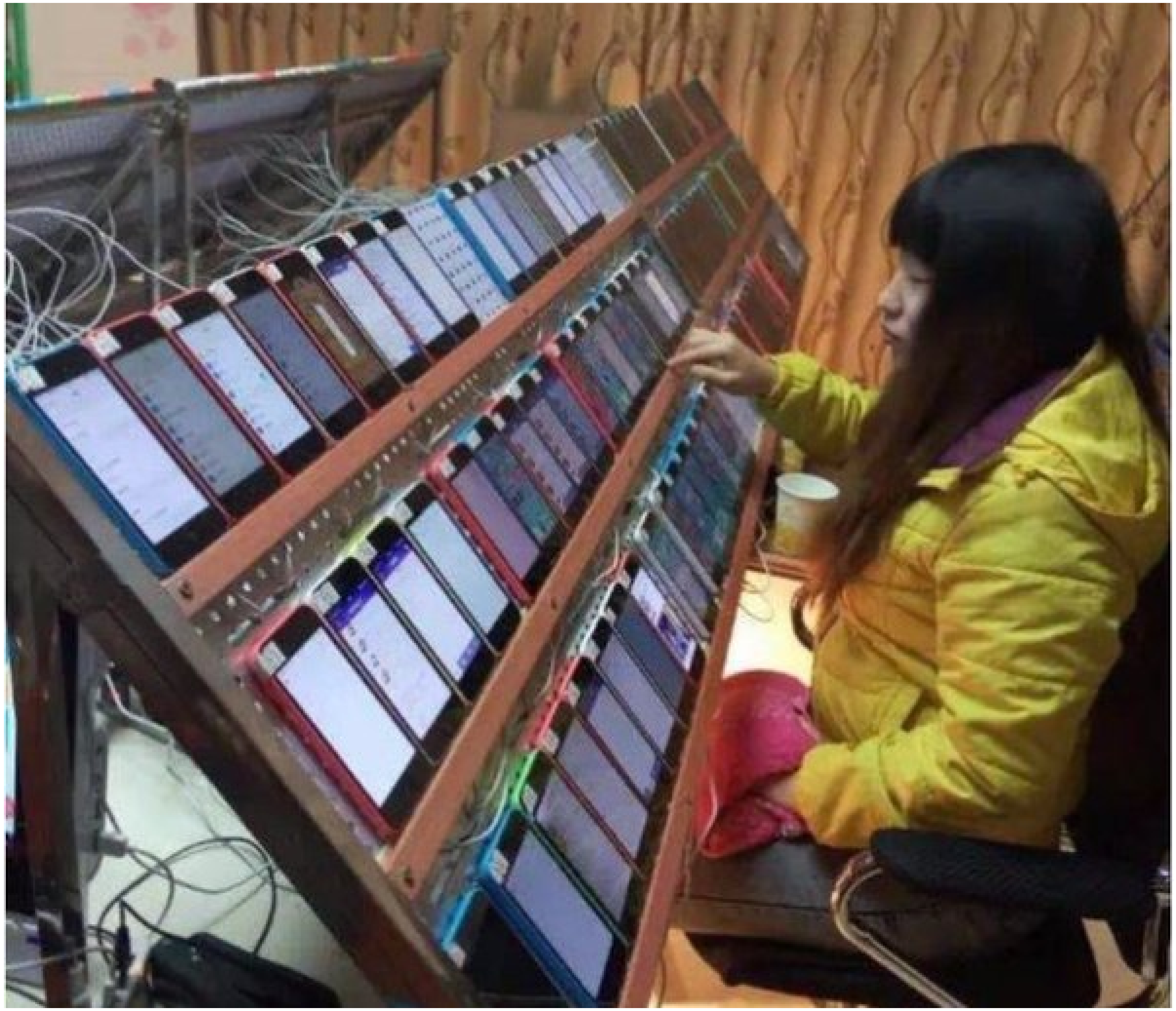
- My name is Matt @lstmemery (https://github.com/lstmemery)
- I'm a data scientist at Imbellus
- My company builds simulations to test problem solving ability
- I use the telemetry from the simulations to build models for prediction
- I first got interested in data science through this group

What is TalkingData?

- TalkingData, China’s largest independent big data service platform
- 3 Billion clicks per day

The Competition

- Find the fradulent click based on IP, App, Device etc.



The Data

- 5 categorical columns (IP, App, Device (iPhone 6+), OS, Channel (Ad Publisher))
- 1 datetime column (click_time)
- 0.25 **Billion** Rows representing 3 days of logs
- Predict a binary "is_attributed"

In [6]:

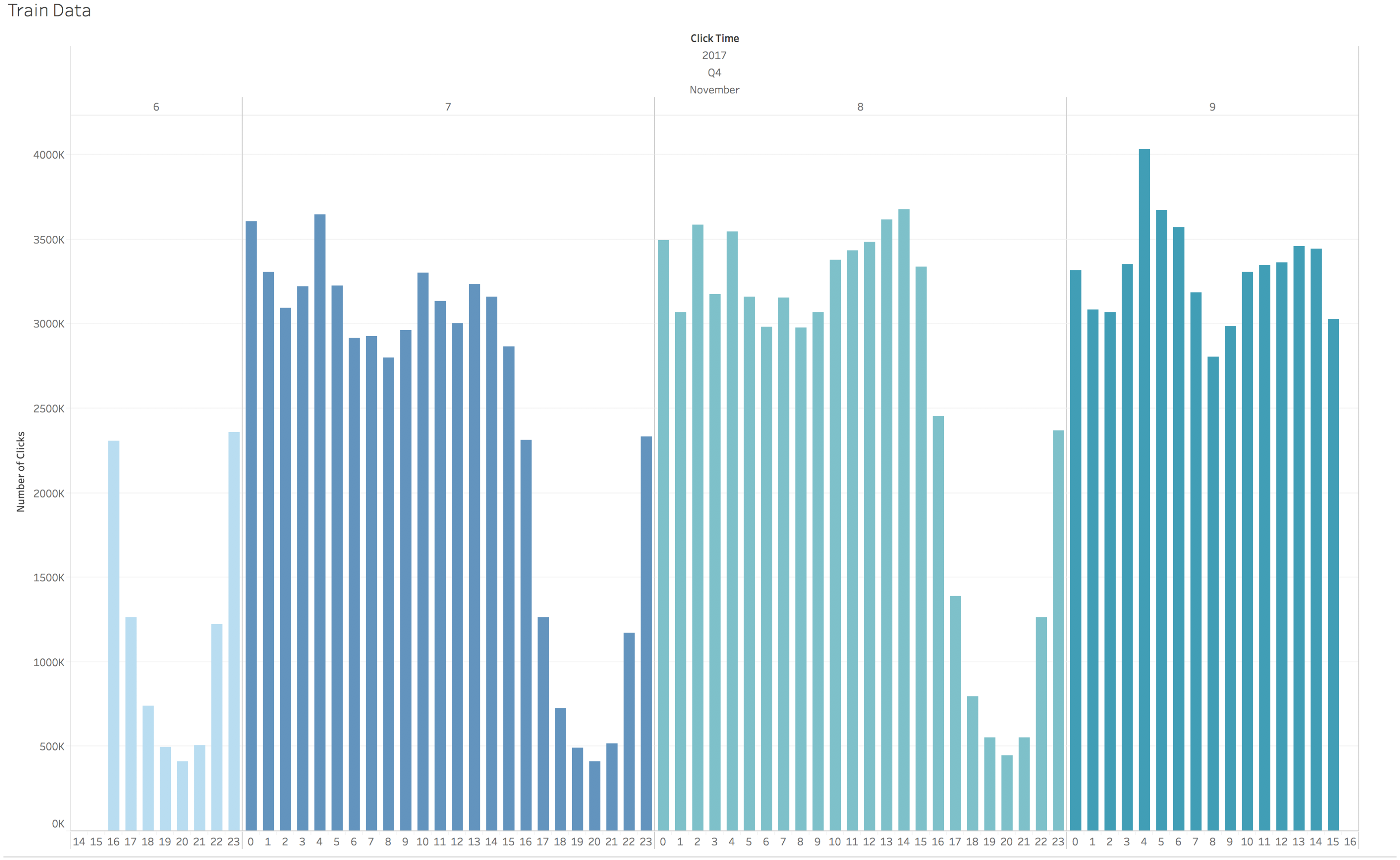
```python
import pandas as pd

df = pd.read_csv("data/train_sample.csv")\
    .drop(columns=["attributed_time"])
df.sample(3)
```

Out[6]:

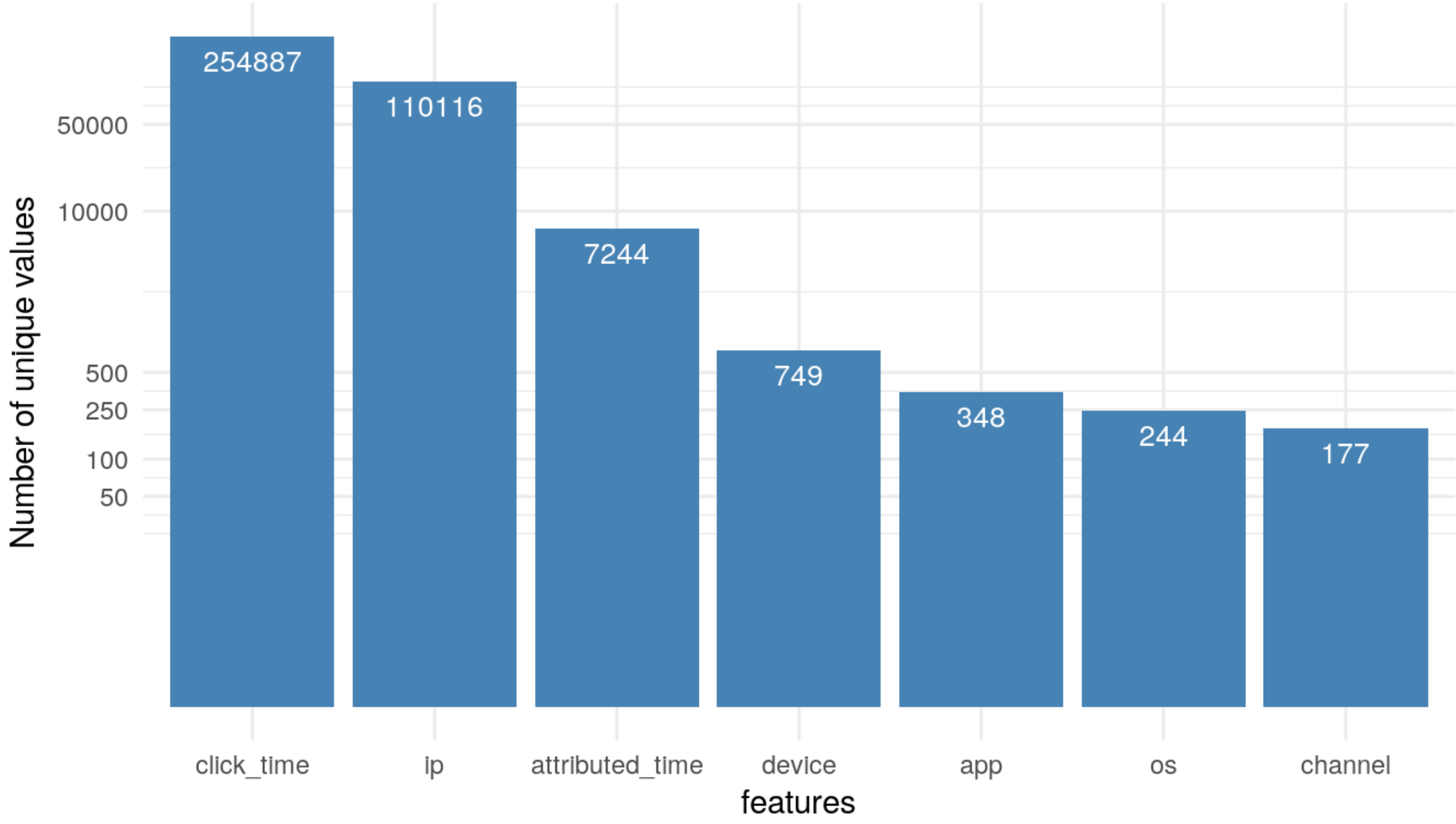
	ip	app	device	os	channel	click_time	is_attributed
53918	97670	18	1	19	121	2017-11-08 03:34:42	0
19802	48240	64	1	19	459	2017-11-07 08:39:20	0
69472	48219	22	1	13	116	2017-11-08 07:33:55	0

Train vs Test

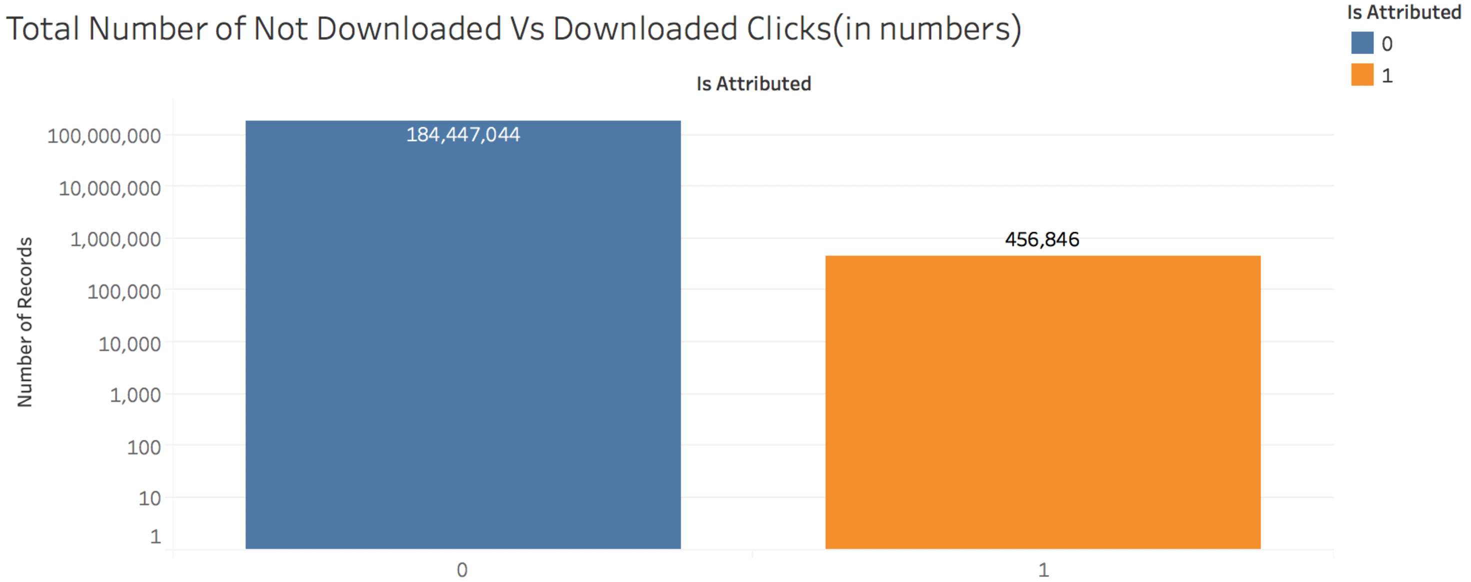
Training data: November 6th to 9th Test data: Some of the 10th

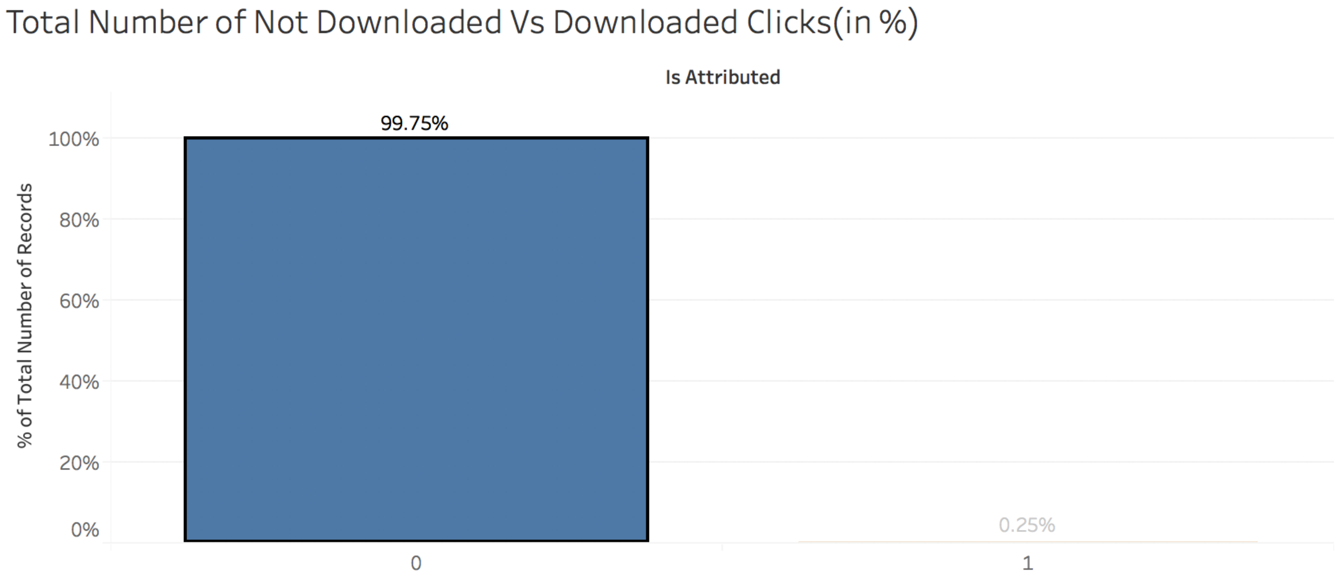


Unique Values



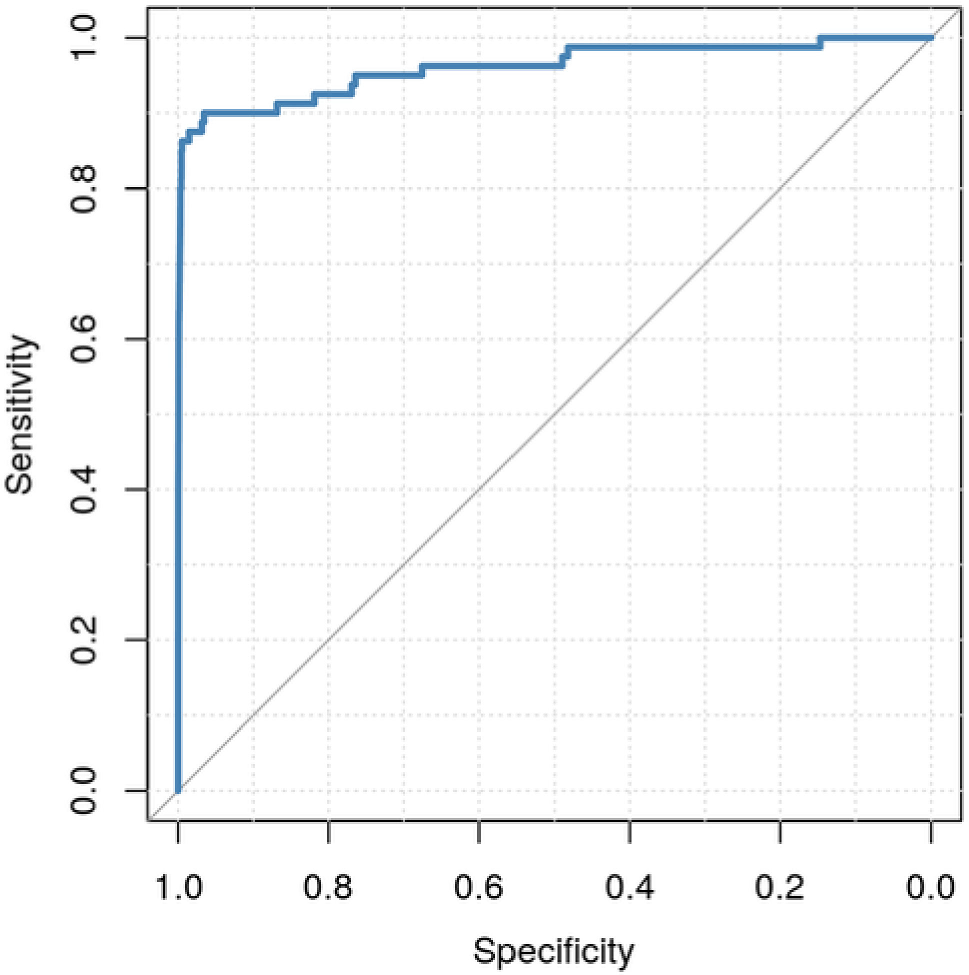
This is a VERY imbalanced dataset





Scoring

- ROC-AUC
- Top Score: 0.9843223
- 2nd Best: 0.9841256



What Is BIG DATA?

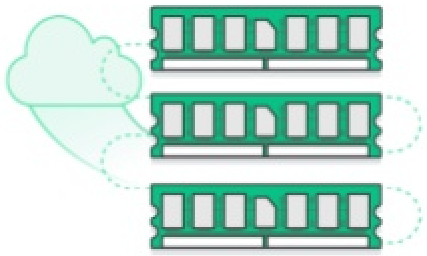
- If it's a pain in the ass to do basic operations like counting in RAM, it's big data
- This is dependent on your computer and libraries you use

> *pandas rule of thumb: have 5 to 10 times as much RAM as the size of your dataset -- Wes McKinney, creator of Pandas*

You Can Always Rent a Bigger Instance!

Amazon EC2 X1 Instances

Designed for SAP HANA



Specifications

- Powered by four Intel® Xeon® E7 8880 v3 (Haswell) processors (64 cores / 128 vCPUs)
- Up to 2TB of DDR4 RAM per instance
- High memory bandwidth and larger L3 caches
- Up to 20 Gbps of network bandwidth
- Up to 10 Gpbs of dedicated bandwidth to Amazon EBS

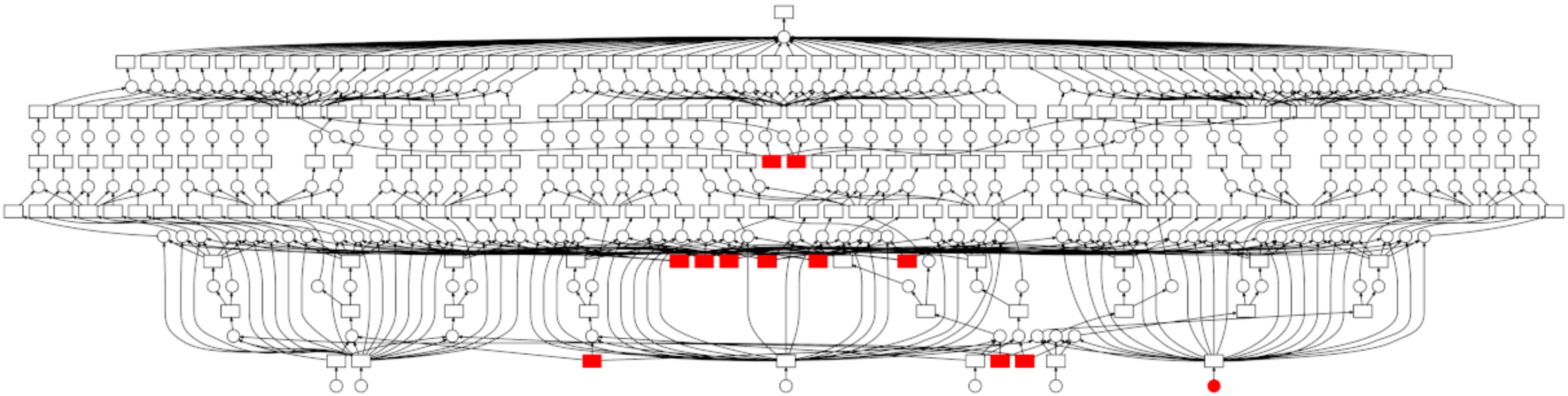
"X1 instances offer more memory than any other SAP-certified cloud instance available today"

https://global.sap.com/community/ebook/2014-09-02-hana-hardware/enEN/iaas.html#

But it'll cost you

Dask (and Parallel computing)

- Dask offers multithreaded, multiprocessing and cluster computing options for numpy, pandas and scikit-learn
- Basic idea: take a really big array, chop it into managable pieces and add it to a collection
- The underlying data structure is a bag (think a set with repeats)



In [1]:
```python
# pip install "dask[dataframe]"

import dask.dataframe as dd

df = dd.read_csv("data/train_sample.csv")
print(type(df))
apps_by_ip = df.groupby("ip")["app"].nunique().compute()
print(type(apps_by_ip))
apps_by_ip.sample(10)
```

```
<class 'dask.dataframe.core.DataFrame'>
<class 'pandas.core.series.Series'>
```

Out[1]:
```
ip
113155    1
13920     1
18507     1
71491     1
16453     6
14903     3
199848    1
162711    1
29086     2
195934    1
Name: app, dtype: int64
```

Use BigQuery

- BigQuery is a big data service that acts like SQL
- This is Google service. You pay for storage (0.023 USD per GB, per month) and queries (first Terabyte each month is free)
- There's a real nice pandas API

In [2]:

```
# pip install pandas-gbq

import pandas as pd

query = """
SELECT
  ip,
  COUNT(DISTINCT(app)) AS app_nunique
FROM
  [tactile-bindery-675:talkingdata.train_sample]
GROUP BY
  ip
LIMIT
  1000
"""
ip_app_count = pd.read_gbq(query, project_id="tactile-bindery-675")
ip_app_count.sample(5)
```

Out[2]:

	ip	app_nunique
382	183199	1
746	25118	3
421	124047	3
700	194308	4
255	41232	10

Undersample

- So Much Data + Data Imbalance = A Good Candidate for Undersampling
- Undersampling means removing examples of the majority class until some point
- Different under/oversampling algorithms can be found in the imbalanced-learn (http://contrib.scikit-learn.org/imbalanced-learn/stable/index.html) package

In [33]:

```
from imblearn.under_sampling import RandomUnderSampler
import numpy as np

df = pd.read_csv("data/train_sample.csv",
                 parse_dates=["click_time"])
print(df.shape)
df["click_time"] = df["click_time"].astype(np.int64) #turn the datetime to a uni
x timestamp
features = df[["ip", "device", "os", "channel", "click_time"]]
target = df["is_attributed"]

undersampler = RandomUnderSampler(random_state=0)
undersampled_features, undersampled_target = undersampler.fit_sample(features, t
arget)
undersampled_features.shape
```

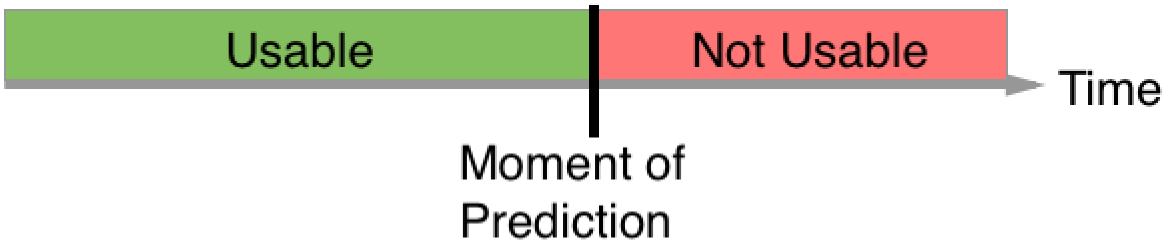
```
(100000, 8)
```

Out[33]:

```
(454, 5)
```

A Note about Data Leakage

- Many competitors (including the winners) leaked distribution information to their models
- Data leakage is when the model is exposed to validation/test set information
- Data leakge results in overconfident models
- A really common example in this case is pre-computing grouped features in BigQuery using train and test set
- It's possible to do this in a Kaggle competition but would be impossible to do in production
- **RULE OF THUMB: If you are doing anything that couldn't be done one row at a time, it's a potential source of data leakage**



Validation Strategies

- Validation is when you leave some training data in reserve to test locally
- The computation cost of running an algorithm over the whole dataset means that validation was critical

- Generally, there were two approaches:

    1. Ignore the time series characteristic of the data and use cross validation
    2. Set aside certain periods of time of the last day of the data

Evaluate your validation strategy by seeing how well your validation score predicts your test score

First Prize (0.9843223)

- Undersampled to equality, throwing out 99.8% of the data
- Bagged five different predictors with five different undersampled datasets
- Did all of the basic feature engineering summary statistics
- Final model was 7 bagged LightGBM models and a bagged neural net
- Validated on the final day of data

Categorical Feature Embedding

- Take any two categorical features (say ip and app id)
- Find all the app ids for a given ip and concatenate them to together as a sentence
- Run your favorite topic model over the feature "sentence"
- Team used non-negative matrix factorization, latent Dirichlet allocation and latent semantic analysis
- Limit 5 topics per embedding

$$5 * 4 \text{ categorical combinations} * 5 \text{ topics per embedding}$$
$$* 3 \text{ types of embedding} = 300 \text{ new features}$$

Score change: 0.9821 to 0.9828

In [ ]:

```
apps_of_ip = {}
for sample in data_samples:
  apps_of_ip.setdefault(sample['ip'], []).append(str(sample['app']))
ips = list(apps_of_ip.keys())
apps_as_sentence = [' '.join(apps_of_ip[ip]) for ip in ips]
apps_as_matrix = CountTokenizer().fit_transform(apps_as_sentence)
topics_of_ips = LDA(n_components=5).fit_transform(apps_as_matrix)
```

Second Prize

- Subsampled for training, final model was trained on all data
- Created 100s of features but none of them were particularly interesting
- Ensembled a ton of LightGBM models using weights inferred from the public leaderboard

Third Prize

- The only one of the top that used RNNs

In [ ]:

```
def build_model(self):
    categorial_inp = Input(shape=(len(self.categorical),))
    cat_embeds = []
    for idx, col in enumerate(self.categorical):
        x = Lambda(lambda x: x[:, idx,None])(categorial_inp)
        x = Embedding(self.categorical_num[col][0], self.categorical_num[col][1
],input_length=1)(x)
        cat_embeds.append(x)
    embeds = concatenate(cat_embeds, axis=2)
    embeds = GaussianDropout(0.2)(embeds)
    continous_inp = Input(shape=(len(self.continous),))
    cx = Reshape([1,len(self.continous)])(continous_inp)
    x = concatenate([embeds, cx], axis=2)
    x = CuDNNGRU(128)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.20)(x)
    x = Dense(64)(x)
    x = PReLU()(x)
    x = BatchNormalization()(x)
    x = Dropout(0.20)(x)
    x = Dense(32)(x)
    x = PReLU()(x)
    x = BatchNormalization()(x)
    x = Dropout(0.05)(x)
```

Fourth Prize

- Used Weight of Evidence Encoding for high cardinality categories

$$WoE = \ln \frac{\%non - events}{\%events}$$

So if IP x had 4 non-events and 2 events:

$$WoE_x = \ln \frac{\frac{4}{6}}{\frac{2}{6}} = \ln \frac{0.667}{0.333} = \ln 2 = 0.693$$

They said this didn't work well, but it may be worth exploring in other competitions

Sixth Prize

- Used a Keras implementation of libFM
- libFM is a matrix factorization library that generalizes to more than two factors https://github.com/jfpuget/LibFM_in_Keras/blob/master/keras_blog.ipynb (https://github.com/jfpuget/LibFM_in_Keras/blob/master/keras_blog.ipynb)

Questions?