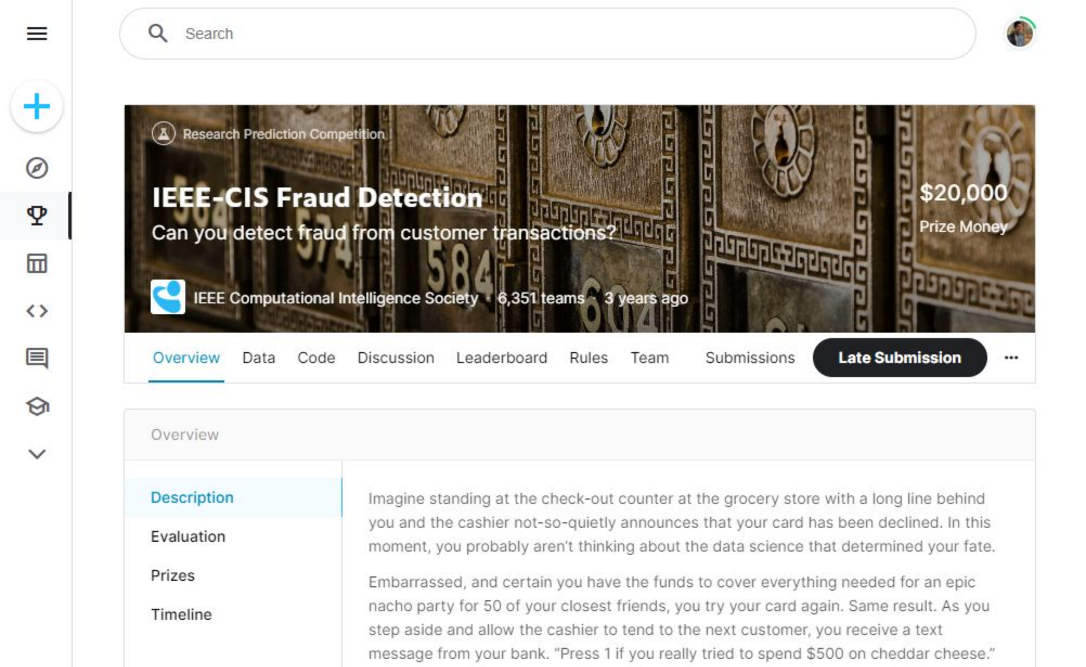
Detecting Credit Card Fraud

Final Project - W207 - Section 11 - Fall 2022

Naikaj Pandya, Chase Madson, Eric Danforth

The Competition: *IEEE-CIS Fraud Detection*



[Link to competition]

- Hosted by IEEE-CIS in 2019 to improve fraud prevention system

```
train.sample(5)
```

[10] ✓ 1.7s

	TransactionID	isFraud	TransactionDT	TransactionAmt
336447	3323447	0	8281233	77.95
443089	3430089	0	11273831	57.95
4917	2991917	0	170923	44.00
534610	3521610	0	14076224	226.00
9535	2996535	1	273968	100.00

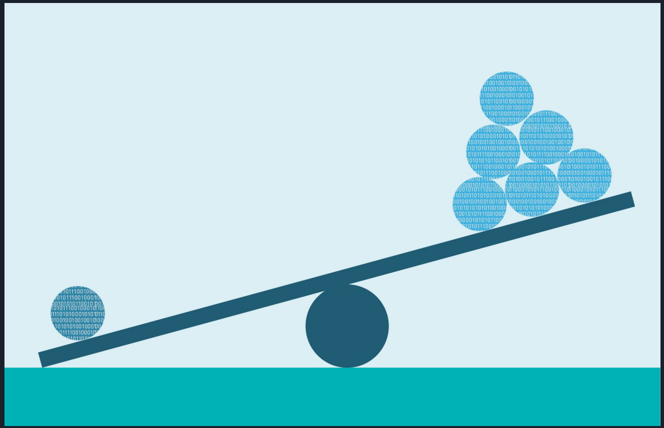
- 590K rows × 430 features for training
  - One record per transaction
  - Binary target: `isFraud == 1`
- **Our goal**: Build a good NN classifier

Available Features for Each Transaction

Group of Features	Features	Example
Basic Transaction Details	Timestamp, amount, payment card, email domains, addr1 &2	Card level details (visa, MC) Product CD (actual product/service) Txn amount, date, etc
Card-Level Counts	C1-C14	number of addresses are associated with the payment card
Card-Level Time Between (Unspecified) Events	D1-D15	days between prior transactions
Card-Level Matches	M1-M9	match, such as names on card and address, etc.
Various Vesta-Enriched Features	V1-V339	Vesta engineered rich features, including ranking, counting, and other entity relations.

Dealing with Imbalanced Data

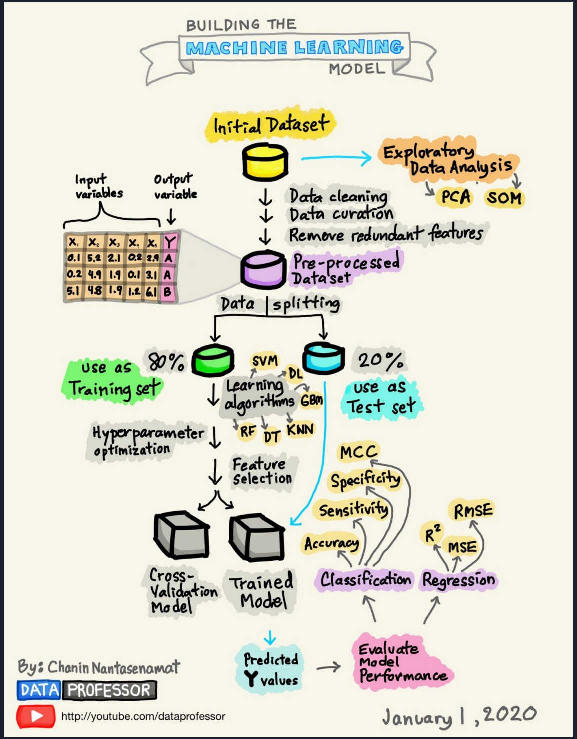
- Fraud detection training dataset is highly imbalanced.
- The training dataset consisted of 96.5% non-fraud transactions and 3.5% fraudulent transactions
- A simple-majority baseline would yield a 96.5% accurate model, but would not be too good at identifying fraud.
- Downsampling: Kept all frauds, and then down sampled to get the same number of random, non-fraud transactions



https://datascience.aero/predicting-improbable-part-1-imbalanced-data-problem/

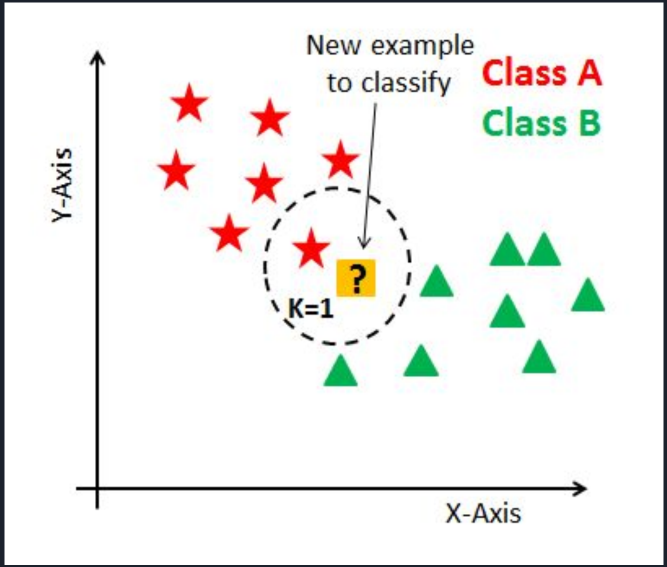
ML Model Design Process Overview

- The problem: to predicting whether a given transaction is fraudulent.
- EDA: Look at the feature space of more than 390 columns
- Baseline Model: KNN (5)
- Neural Network model:
  - Deeper EDA
  - Data processing, cleanup
  - Feature engineering (iterative)
  - Build & fit model



Baseline Model

- Simple majority baseline model would not work due to heavy data imbalance.
- We decided on using KNN model, since we theorized that fraudulent transactions could have patterns and clustering in certain features.
- 18 features to develop KNN model: TransactionDT, TransactionAmt, ProductCD, card1, and C1-C14
- Baseline model has validation AUC of 86.9% and a Kaggle score of 82.5%



https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn

KNN Baseline Score
Public: 86.9%
Private: 82.5%

Data Pre-Processing & Feature Selection

De-anonymization

Dimensionality reduction

- Limited PCA given time available and number of anonymized features
- Dropped identity columns in the transaction table that weren't providing much information
- Identity table fields such as 'browser' and 'OS' mapped down to family groups

Min-max normalization

Missing values replaced for numeric values

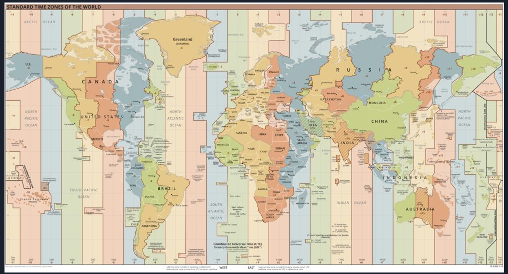
Categorical features one-hot encoded

https://www.goodwin.edu/enews/investigator-vs-detective/

Feature Engineering

- Time & location features - training data range
- Top level domains for recipient and purchaser emails
- Transaction in foreign currency - 3rd decimal place
- Create heuristic for User ID
- Filled identity table & De-anonymized features
    - Country - Most fraud originating from Russia and China
    - Timezone
- Foreign currency conversion

https://www.xe.com/currencycharts/?from=USD&to=EUR&view=5Y



https://en.wikipedia.org/wiki/File:World_Time_Zones_Map.png

Keras Model

- Final model used Keras Sequential API
- 4 Dense layers, each with Batch Normalization and Dropout applied
- Total parameters: 730,881
- We had a working functional model as well
- This gave us flexibility to conduct data preprocessing within the model
- We found that some Tensorflow features are hardware specific and not easily portable to accelerators

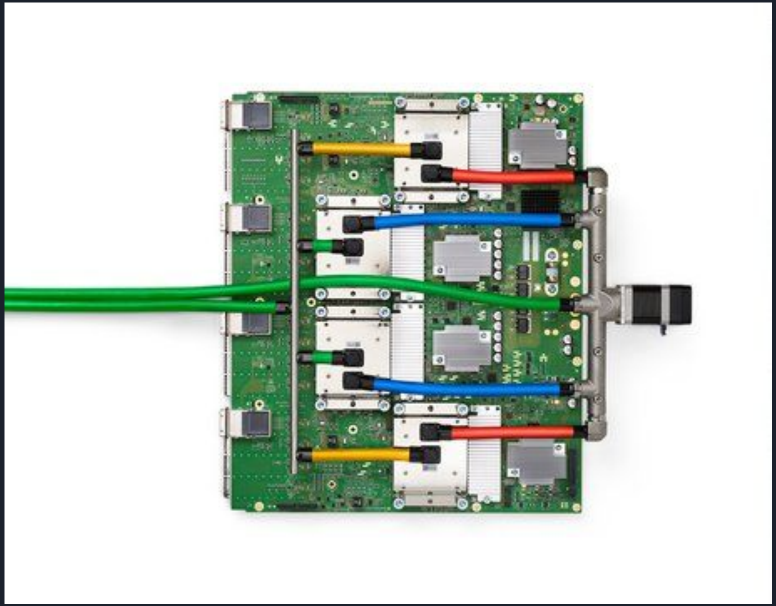
Layer (type)	Output Shape	Param #
dense_0 (Dense)	(None, 512)	496128
batch_norm_0 (BatchNormalization)	(None, 512)	2048
dropout_0 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
batch_norm_1 (BatchNormalization)	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
batch_norm_2 (BatchNormalization)	(None, 256)	1024
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
batch_norm_3 (BatchNormalization)	(None, 128)	512
dropout_3 (Dropout)	(None, 128)	0
dense (Dense)	(None, 1)	129

Total params: 730,881
Trainable params: 728,577
Non-trainable params: 2,304

https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html

Accelerators - TPUs

- We experimented with accelerators to speed up training time to allow more experimentation and larger models
- Training epochs were lightning fast even with thousands of features; however, model serialization proved to be a memory intensive operation
- Overall training time for a model of this size was not improved by using TPUs
- Performance metrics were degraded compared to the same model architecture on a CPU, possibly due to a reduction in decimal precision

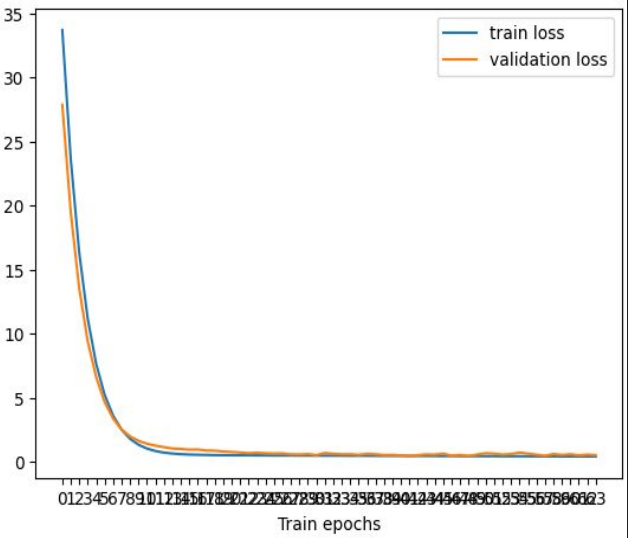


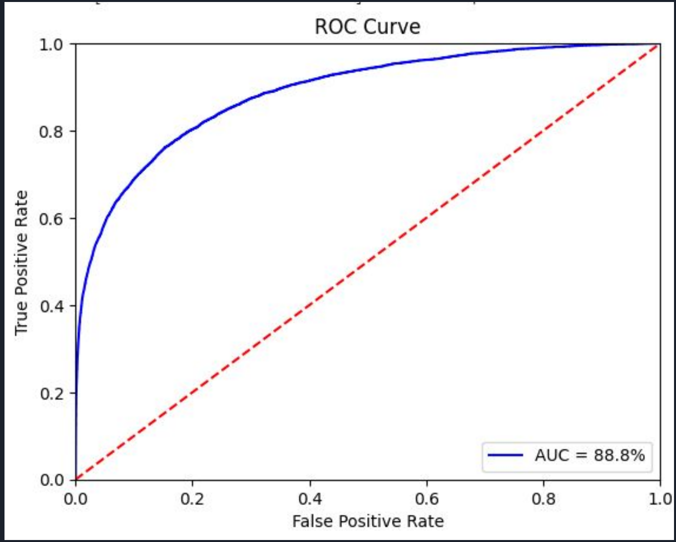
https://blog.google/technology/developers/io21-helpful-google/

Hyperparameter Tuning

Hyperparameter Type	Hyperparameter	Selection
Training and backpropagation	Loss Function and Validation Split	Binary cross-entropy w/ 10% split
	Optimizer and Learning Rate	Adam w/ L.R. = 0.0005
	Epochs and Batch Size	64 epochs in batches of 1024
Hidden Layers	Hidden Layers and Nodes	[512, 256, 256, 128]
	Activation Function	ReLU
Regularization	Model Parameters	L2 regularization w/ lambda = 0.05
	Hidden Layers	Batch normalization and dropout at a rate of 50%

Model Performance and Results



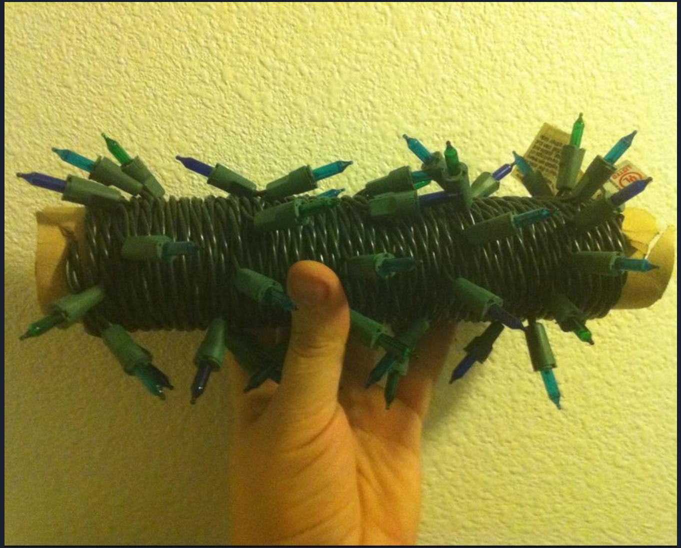


Best Kaggle Score
Public: 87.3%
Private: 84.9%

KNN Baseline Score
Public: 86.9%
Private: 82.5%

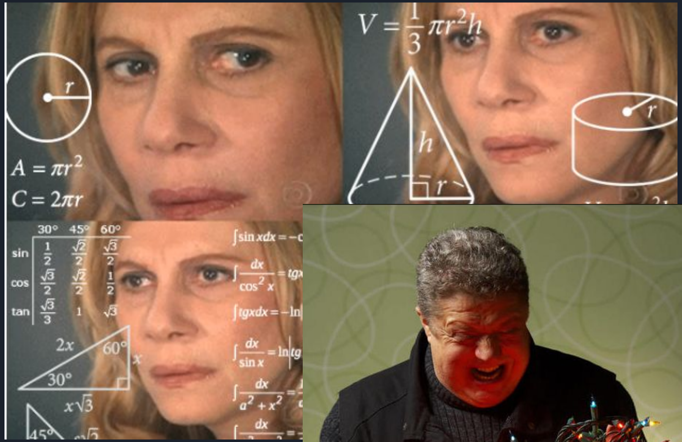
Data Generating Process for our Labels

What we expected



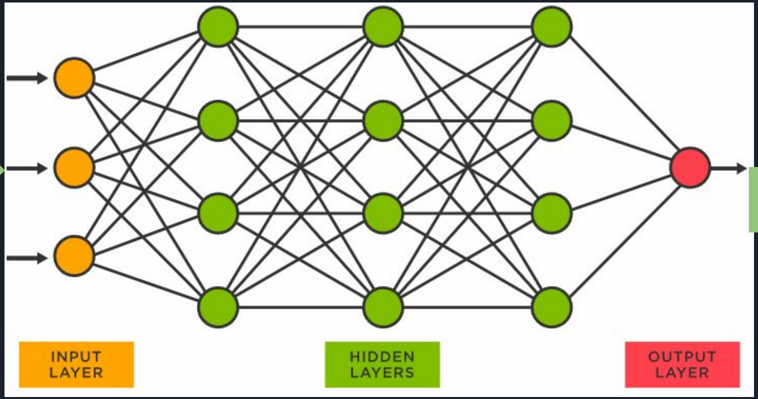
https://stressexterminators.wordpress.com/2015/02/10/stress-hacking-christmas-lights-organization/

What we got



Train New Model on Latest Transaction by Card

card1	addr1	date_account_opened	P_emaildomain	isFraud	txn_timestamp
~~7585~~	~~433.0~~	~~2018-04-14~~	~~yahoo.com~~	~~1~~	~~2018-08-19 15:22:46-07:00~~
~~7585~~	~~433.0~~	~~2018-04-14~~	~~yahoo.com~~	~~1~~	~~2018-08-28 02:12:01-07:00~~
~~7585~~	~~433.0~~	~~2018-04-14~~	~~yahoo.com~~	~~1~~	~~2018-09-02 16:06:44-07:00~~
~~7585~~	~~433.0~~	~~2018-04-14~~	~~yahoo.com~~	~~1~~	~~2018-09-06 18:48:05-07:00~~
~~7585~~	~~433.0~~	~~2018-04-14~~	~~yahoo.com~~	~~1~~	~~2018-10-21 14:54:18-07:00~~
7585	433.0	2018-04-14	yahoo.com	1	2018-11-25 01:00:25-08:00



Unshrink

Kaggle Score

Public: 85.9%

Private: 84.3%

Val. AUC: 85.6%

https://www.tibco.com/reference-center/what-is-a-neural-network

A Very Simple Ensembling Approach

Best Model

<u>Kaggle Score</u>
Public: 87.3%
Private: 84.9%

Using Latest Transaction

<u>Kaggle Score</u>
Public: 85.9%
Private: 84.3%

Simple Average of the Two

<u>Kaggle Score</u>
Public: 88.4%
Private: 86.6%

Things we Tried, or Would Try in Future

Trial and Error

- ❖ PCA dim. reduction of the 339 Vesta Features
- ❖ Impute median on missing values
- ❖ Log transform heavy-skewed features

Looking ahead

- ❑ More feature engineering
- ❑ Use just the “top 20” Vesta Features
- ❑ Use more data from the identity table provided
- ❑ Better approximation to identify genuine fraud and/or an ID for card
- ❑ Smarter missing value imputation
- ❑ Categorical embeddings - cat2vec

Questions?

https://developer.confluent.io/tutorials/credit-card-activity/confluent.html

Thank You!