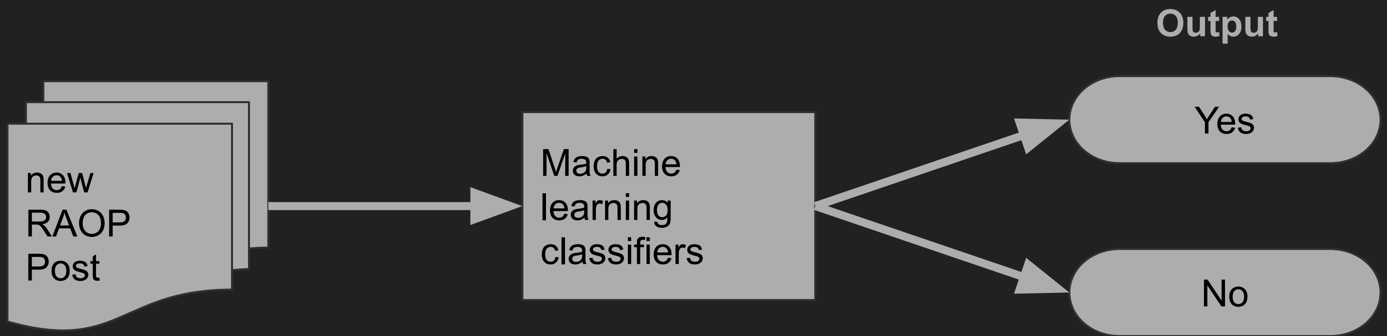
Random Acts of Pizza

Kanika Mahajan, Carolina Arriaga

W207 Applied Machine Learning

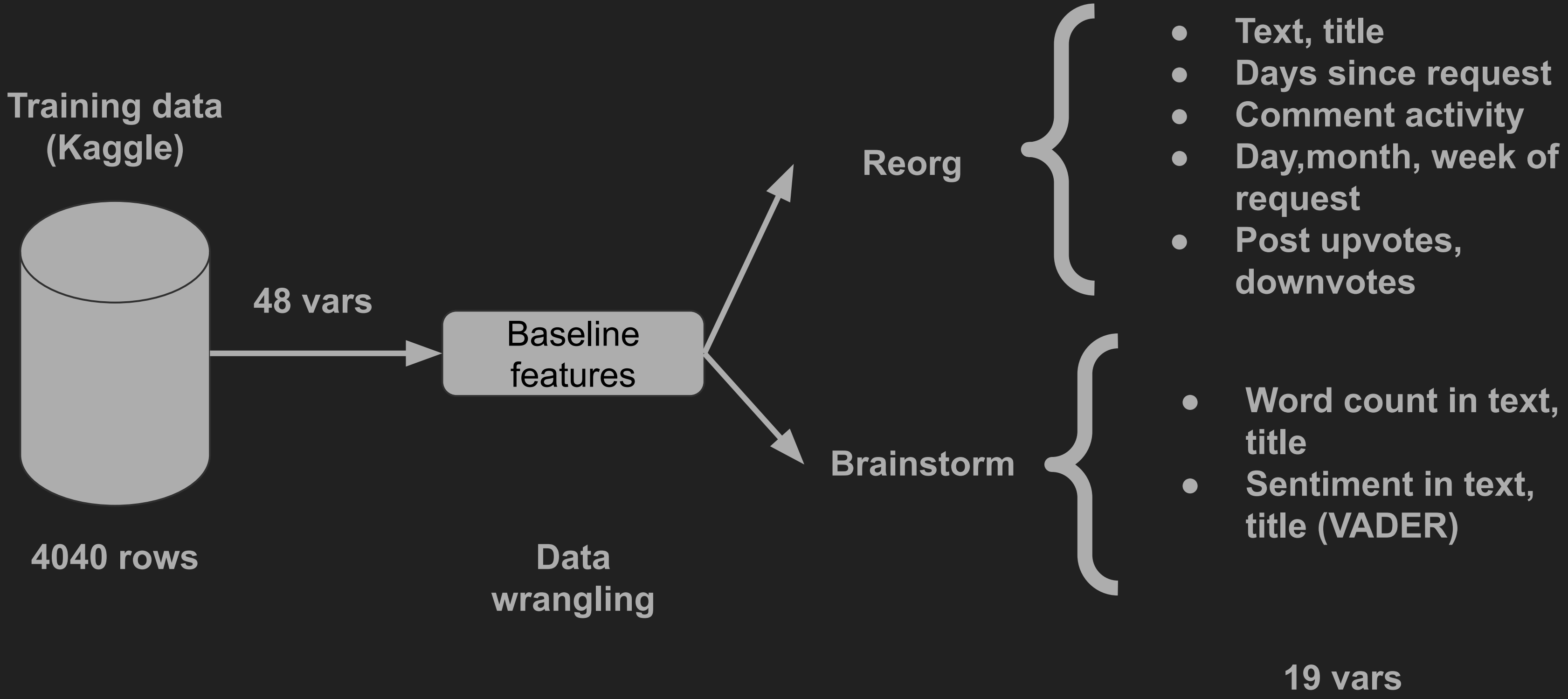
Problem description

In the Random Acts of Pizza challenge (Kaggle competition) we are asked to **predict** if a message posted on Reddit group RAOP will get a pizza or not.



Accuracy >> random (0.5) & ≈ 1

Data description



Training data (Kaggle)

48 vars

Baseline features

4040 rows

Data wrangling

Reorg

- Text, title
- Days since request
- Comment activity
- Day,month, week of request
- Post upvotes, downvotes

Brainstorm

- Word count in text, title
- Sentiment in text, title (VADER)

19 vars

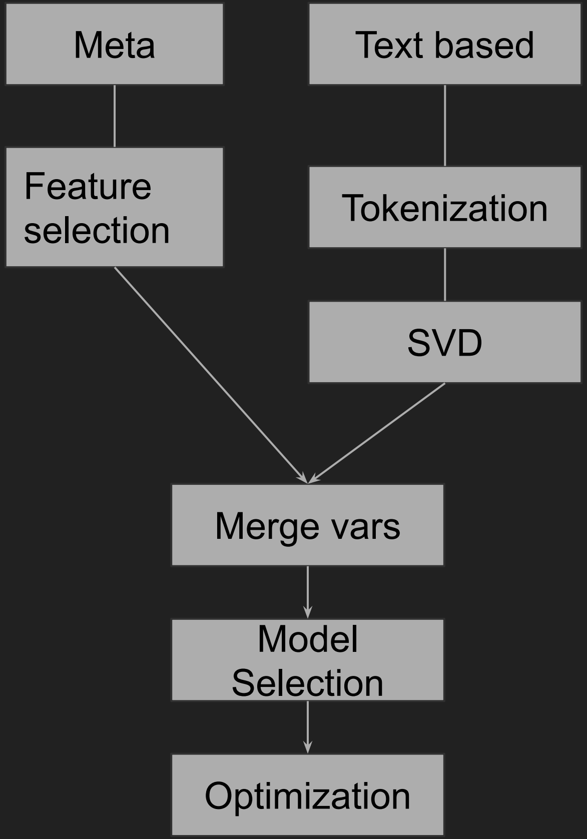
Challenges

- The initial data was unbalanced.
  - 24% of the posts received pizza
  - Classifiers might be biased towards predicting “False”.
- We had a small data set (4,040 rows)
  - Test data not available (Kaggle competition)
  - Split in three ways -> even smaller dataset (75-15-10)
- Team objective:

  get accuracy > 90%

Our Approach

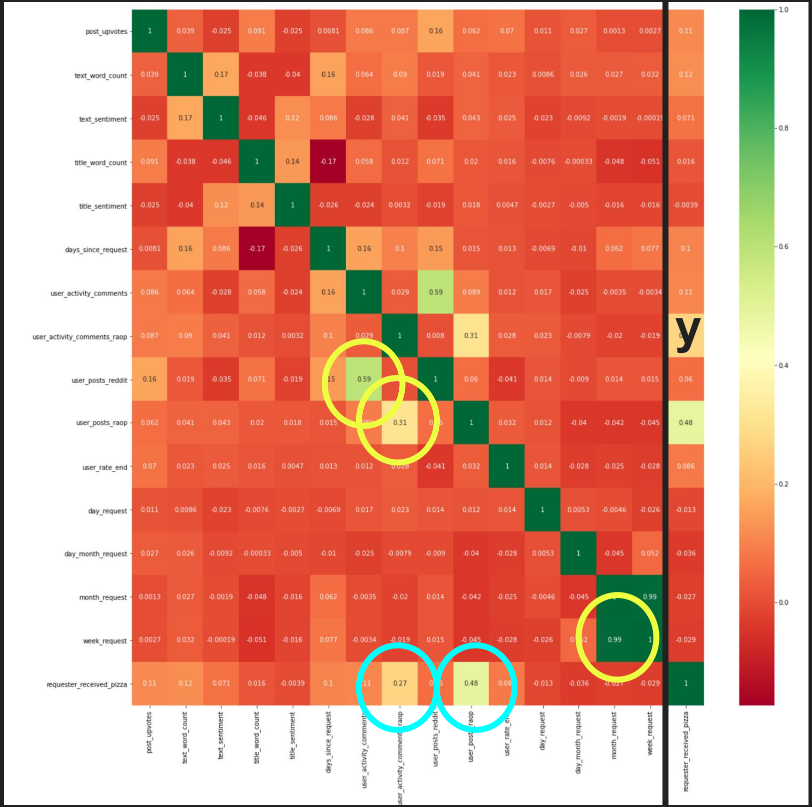
- Feature engineering
  - Meta - related to user or our understanding of language (e.g. post upvotes, activity)
  - Text based features - word counts, sentiment, TFIDF
  - Matrix correlation
  - Dimensionality reduction - SVD
- Model selection
  - Logistic regression
  - Random Forests (RF, Ada, XGB)
  - Multilayer Neural Network
  - Dense Neural Network
- Hyperparameter optimization



Feature Selection - Meta

Finding features that are independent from each other to avoid multicollinearity.

- Highly correlated covariates:
  - Week - Month
  - Comments - Posts on Reddit
  - Comments raop - Posts on raop
- Removed week of request, posts on reddit, comments on raop.
- High correlation between output variable posts on raop (blue) - good variables.



	post_upvotes	text_word_count	text_sentiment	title_word_count	title_sentiment	days_since_request	user_activity_comments	user_activity_comments_raop	user_posts_reddit	user_posts_raop	user_rate_end	day_request	day_month_request	month_request	week_request	requester_received_pizza
post_upvotes	1	0.039	-0.025	0.091	-0.025	0.0081	0.086	0.087	0.16	0.062	0.07	0.011	0.027	0.0013	0.0027	0.11
text_word_count	0.039	1	0.17	-0.038	-0.04	0.16	0.064	0.09	0.019	0.041	0.023	0.0086	0.026	0.027	0.032	0.12
text_sentiment	-0.025	0.17	1	-0.046	0.12	0.086	-0.028	0.041	-0.035	0.043	0.025	-0.023	-0.0092	-0.0019	-0.00019	0.071
title_word_count	0.091	-0.038	-0.046	1	0.14	-0.17	0.058	0.012	0.071	0.02	0.016	-0.0076	-0.00033	-0.048	-0.051	0.016
title_sentiment	-0.025	-0.04	0.12	0.14	1	-0.026	-0.024	0.0032	-0.019	0.018	0.0047	-0.0027	-0.005	-0.016	-0.016	-0.0039
days_since_request	0.0081	0.16	0.086	-0.17	-0.026	1	0.16	0.1	0.15	0.015	0.013	-0.0069	-0.01	0.062	0.077	0.1
user_activity_comments	0.086	0.064	-0.028	0.058	-0.024	0.16	1	0.029	0.59	0.089	0.012	0.017	-0.025	-0.0035	-0.0034	0.11
user_activity_comments_raop	0.087	0.09	0.041	0.012	0.0032	0.1	0.029	1	0.008	0.31	0.028	0.023	-0.0079	-0.02	-0.019	[illegible]
user_posts_reddit	0.16	0.019	-0.035	0.071	-0.019	0.15	0.59	[illegible]	1	0.06	-0.041	0.014	-0.009	0.014	0.015	0.06
user_posts_raop	0.062	0.041	0.043	0.02	0.018	0.015	0.089	0.31	0.06	1	0.032	0.012	-0.04	-0.042	-0.045	0.48
user_rate_end	0.07	0.023	0.025	0.016	0.0047	0.013	0.012	0.028	-0.041	0.032	1	0.014	-0.028	-0.025	-0.028	0.086
day_request	0.011	0.0086	-0.023	-0.0076	-0.0027	-0.0069	0.017	0.023	0.014	0.012	0.014	1	0.0053	-0.0046	-0.026	-0.013
day_month_request	0.027	0.026	-0.0092	-0.00033	-0.005	-0.01	-0.025	-0.0079	-0.009	-0.04	-0.028	0.0053	1	-0.045	0.052	-0.036
month_request	0.0013	0.027	-0.0019	-0.048	-0.016	0.062	-0.0035	-0.02	0.014	-0.042	-0.025	-0.0046	-0.045	1	0.99	-0.027
week_request	0.0027	0.032	-0.00019	-0.051	-0.016	0.077	-0.0034	-0.019	0.015	-0.045	-0.028	-0.026	0.052	0.99	1	-0.029
requester_received_pizza	0.11	0.12	0.071	0.016	-0.0039	0.1	0.11	0.27	0.06	0.48	0.086	-0.013	-0.036	-0.027	-0.029	1

Correlation matrix

Feature engineering - TFIDF + SVD

We tried several approaches:

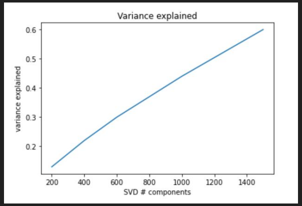
- Only tf-idf, word count (using token pattern)
- Better processor:
  - Removes special characters
  - Lemmatization and stemming
  - Remove stop words
- Used ngrams
  - Range (1-4)
- SVD

Tokenization

Up to 300k ngrams

SVD

Tested with: 200, 400, 600, 1000, 1500



Model Eval

Optimize Baseline Logistic regression

Random forest (XGB, Ada, RF)

Model Overview - Baseline

Logistic Regression

- Multiple componentes = 1500
- L2 Regularization

Trees - (DT, RF, Ada)

- Estimators=20
- Depth=4

7-NN

- Tried [1,3,5,7,9,11]

Accuracy (Dev set)

Model	Accuracy
Logit	84%
DT	75%
RF	82%
Ada	83%
7-NN	72%

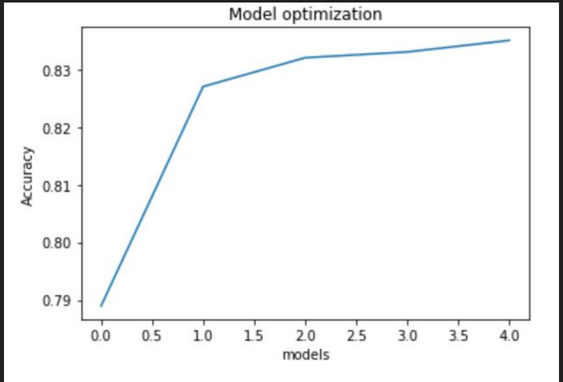
Model Overview - Optimization

Logistic Regression

- **SVD componentes** = [**200**, 400, 600, 1000, 1500]
- Multiple smoothing values **C**=[0.01,0.1,**1**,10,100]
- L2 Regularization

Trees - (**XGB**, Ada, RF)

- Multiple estimators [15, 30, 80, 100, 120, **250**]
- Multiple depths [2,3,4,5,6,**12**]



Accuracy (Dev set)

Logit	XGB
84.5%	83%

1% improvement

Model Overview - New models

Multilayer Neural Network

- Used MLPClassifier with dimension (216,216,216) - our best result of components.

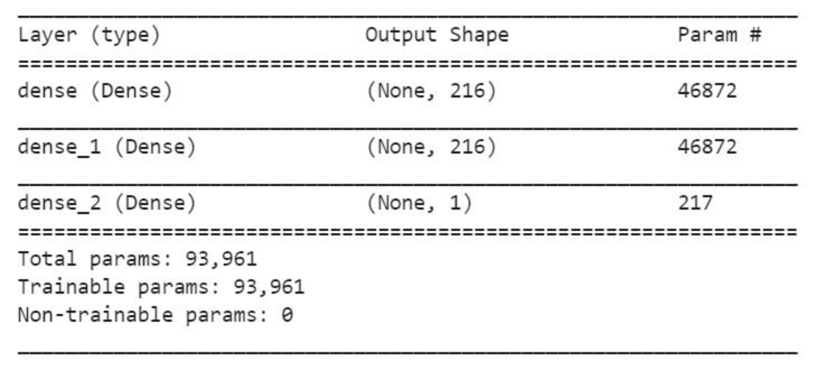
Dense Neural Network

- Included 3 sigmoid layers
  - 2 of size 216
  - Third layer of size 1
  - Sigmoid

Accuracy (Dev set)

MLNN	DNN
82.17%	92.17%

+7% improvement



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 216)               46872
_________________________________________________________________
dense_1 (Dense)              (None, 216)               46872
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 217
=================================================================
Total params: 93,961
Trainable params: 93,961
Non-trainable params: 0
_________________________________________________________________
```

Model Overview - New model optimization

Dense Neural Network

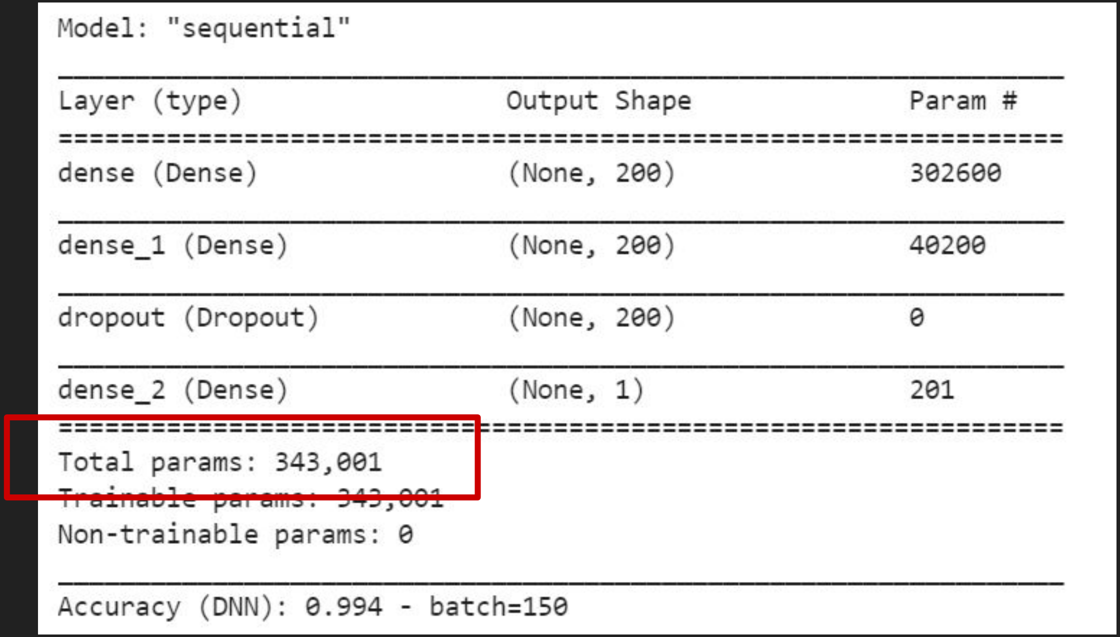
- Considered 1512 SVD (~60% variance)
- Stochastic Gradient Descent- batch size = 1
- Mini_batches = [20,60,100,150]
- 2 Layers
- Units = [200 & 500] each layer
- Epochs = 100

Dev set

Model	Accuracy
Stochastic GD	86%
Mini batch (20)	97%
Mini batch (60)	98%
Mini batch (150)	99%

+7% improvement

DNN - Best model



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 200)               302600
_________________________________________________________________
dense_1 (Dense)              (None, 200)               40200
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 201
=================================================================
Total params: 343,001
Trainable params: 343,001
Non-trainable params: 0
_________________________________________________________________
Accuracy (DNN): 0.994 - batch=150
```

Increased parameters

~ 3.6x

Increased accuracy to 99%

Added Noise (dropout=0.5)

Mini batch GD

Optimized parameters - Best models

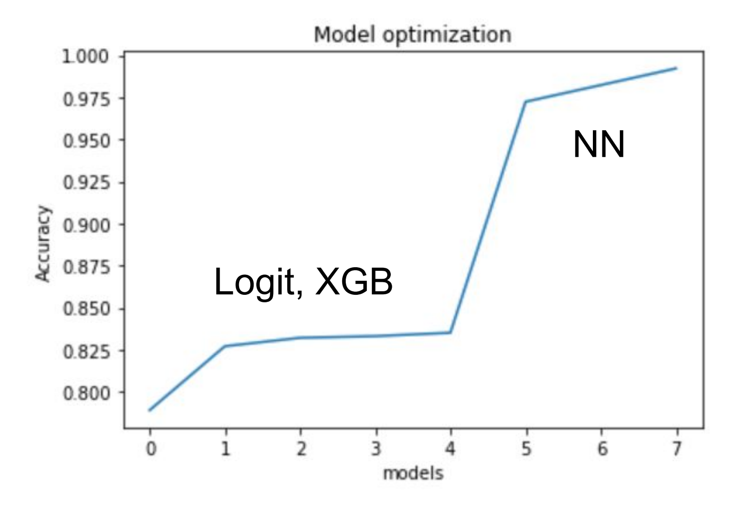
Model	Best hyperparameters	Accuracy
Logit	C = 1, L2, SVD=1512	85%

XGB	Depth=4
	Estimators=120
	SVD=1512
		84%

DNN	SVD=1512 (input)
	Sequential
	L1= 200 comps & Sigmoid
	L2= 200 comps & Sigmoid
	Dropout = 0.5
	L3= 1 comps & Sigmoid
	
	Batch size=150
	Epochs = 100
	
	Optimizer=Adam
		99%

Results

We tested our optimized models with the validation data.



Accuracy

Model	Dev	Test
Logit	86%	82%
XGB	84%	83%
DNN mini batch (150)	98%	99%

What worked

- Removing covariates with high correlation.
- SVD components ~ hundreds.
- Logistic regression was slightly better than forests, also faster.
- Dense Neural Network predicted the best results.

What didn’t work

- Baseline model
  - KNN model, accuracy was much lower compared to RF and Logistic Regression.
- Multilayer Neural Network
  - Didn’t do better than trees or logit.
  - Didn’t explore further.
- Dense Neural Network
  - We tried different activation layers (relu, softmax) - sigmoid was the best

Lessons learned

- We could have created an ensemble
  - Include multiple model predictions
- Some models were running well in a computer but due to computing power not so well in other.
  - We could have created a container
- Compiling takes a long time
  - Need to find ways to optimize computing time.