Pizza Learning:

Random Acts of Pizza

Ricardo Barrera, Yang Linfang
Denis Vrdoljak, Max Yan

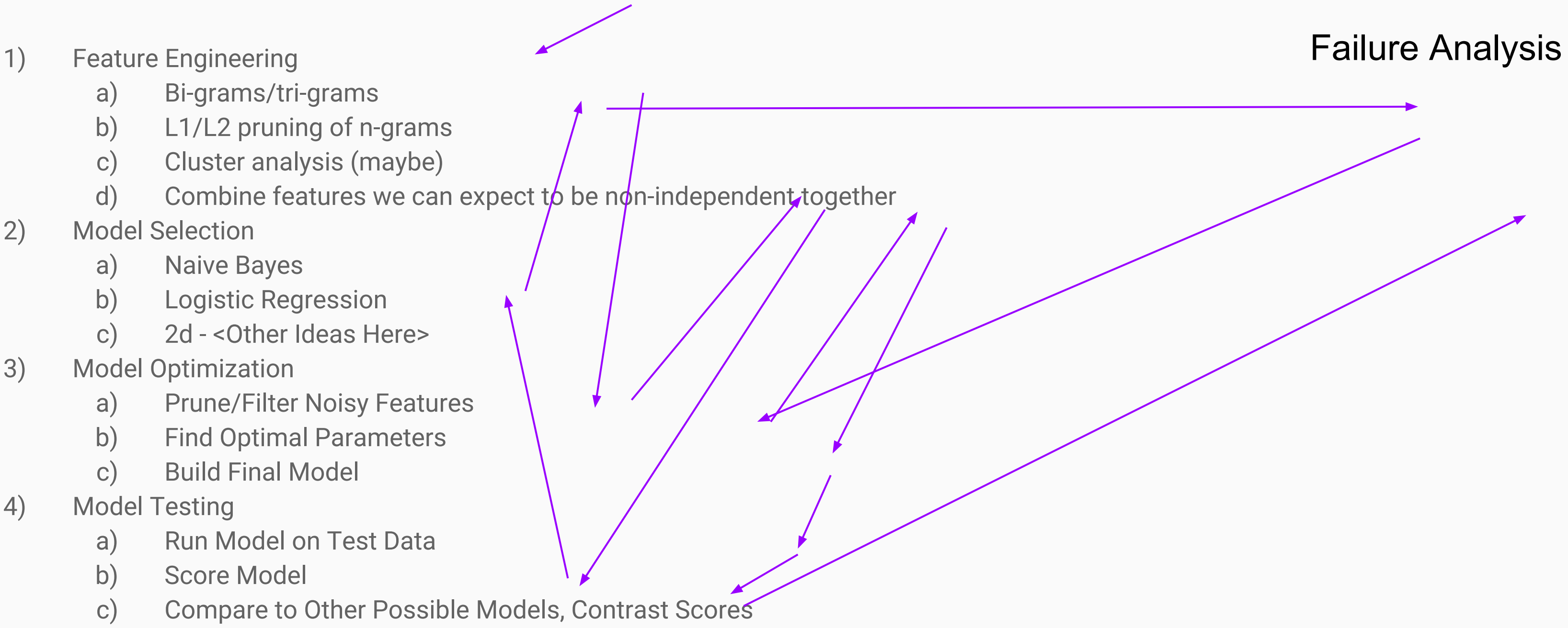
Overview

- Users request a pizza, and provide a short text of their request.
- aka: Random Act of Pizza, or RAOP
- Other users, if they feel so inclined, accept the RAOP and buy/send the requester a pizza (positive outcome).
- If no user responds to the RAOP, the user doesn’t get a pizza (negative outcome).
- The majority of cases result in a negative outcome.
- Various meta data is logged and is part of the initial dataset.

Example Entry

```json
{
 "giver_username_if_known": "N/A",
 "in_test_set": false,
 "number_of_downvotes_of_request_at_retrieval": 2,
 "number_of_upvotes_of_request_at_retrieval": 6,
 "post_was_edited": false,
 "request_id": "t3_w5491",
 "request_number_of_comments_at_retrieval": 7,
 "request_text": "I'm not in College, or a starving artist or anything like that. …
 "requester_account_age_in_days_at_request": 14.416875,
 "requester_account_age_in_days_at_retrieval": 531.9697222222222,
 "requester_days_since_first_post_on_raop_at_request": 0.0,
 "requester_days_since_first_post_on_raop_at_retrieval": 517.5111805555556,
 "requester_number_of_comments_at_request": 8,
 "requester_number_of_comments_at_retrieval": 93,
 "requester_number_of_comments_in_raop_at_request": 0,
 "requester_number_of_comments_in_raop_at_retrieval": 4,
 "requester_number_of_posts_at_request": 1,
 "requester_number_of_posts_at_retrieval": 6,
 "requester_number_of_posts_on_raop_at_request": 0,
 "requester_number_of_posts_on_raop_at_retrieval": 2,
 "requester_number_of_subreddits_at_request": 8,
 "requester_subreddits_at_request": [
  "AdviceAnimals",
  "WTF",
  "funny",
  "gaming",
  "movies",
  "technology",
  "todayilearned",
  "videos"
 ],
 "requester_upvotes_minus_downvotes_at_request": 32,
 "requester_upvotes_minus_downvotes_at_retrieval": 212,
 "requester_upvotes_plus_downvotes_at_request": 48,
 "requester_upvotes_plus_downvotes_at_retrieval": 610,
 "requester_user_flair": "shroom",
 "requester_username": "RitalinYourMemory",
 "unix_timestamp_of_request": 1341604684.0,
 "unix_timestamp_of_request_utc": 1341601084.0
}
```

Plan and Reality

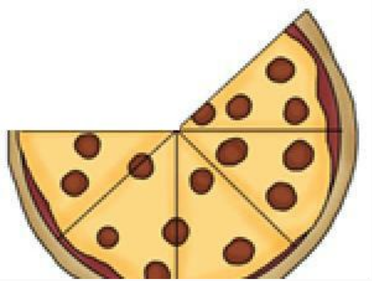


1) Feature Engineering
    a) Bi-grams/tri-grams
    b) L1/L2 pruning of n-grams
    c) Cluster analysis (maybe)
    d) Combine features we can expect to be non-independent together
2) Model Selection
    a) Naive Bayes
    b) Logistic Regression
    c) 2d - <Other Ideas Here>
3) Model Optimization
    a) Prune/Filter Noisy Features
    b) Find Optimal Parameters
    c) Build Final Model
4) Model Testing
    a) Run Model on Test Data
    b) Score Model
    c) Compare to Other Possible Models, Contrast Scores

Failure Analysis

63%

Baseline Accuracy



- **request_text_edit_aware**: request_text field after stripping out edits
- Tokenized into 2200 features
- Resulted in 63% accuracy

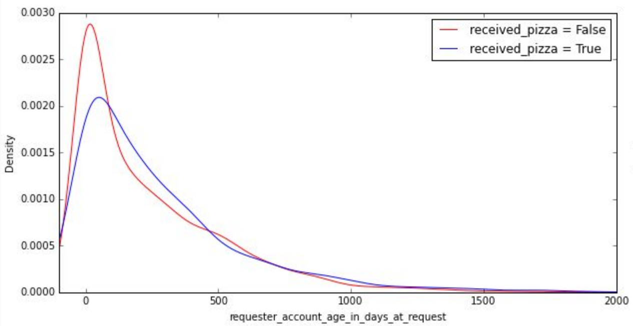
Feature Engineering

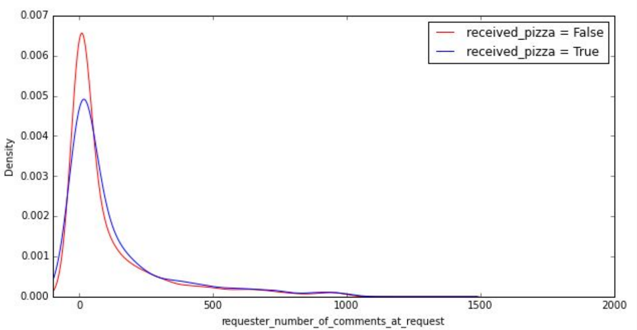
- Tokenized Title Text
- Tokenized Entry Text
- Length of Text
- Length of Title
- Time of day
- Hyperlinks
- Upvotes - Downvotes
- etc

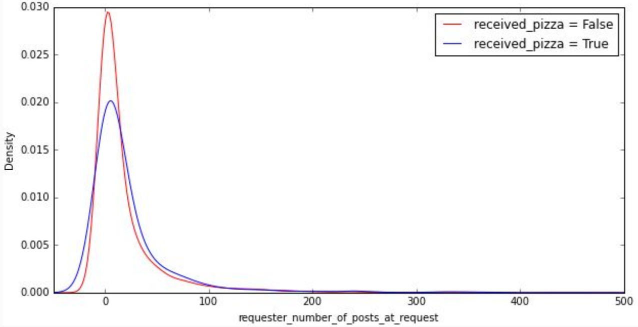
- requester_number_of_posts_on_raop_at_request
- requester_number_of_subreddits_at_request
- requester_subreddits_at_request
-

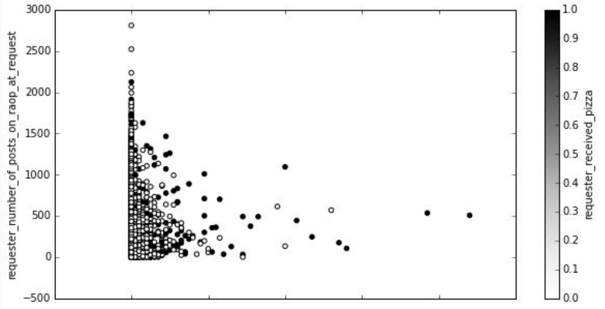
Feature Explorations

- hour_of_request
- requester_upvotes_minus_downvotes_at_request
- day_of_week
- day_of_month
- ...







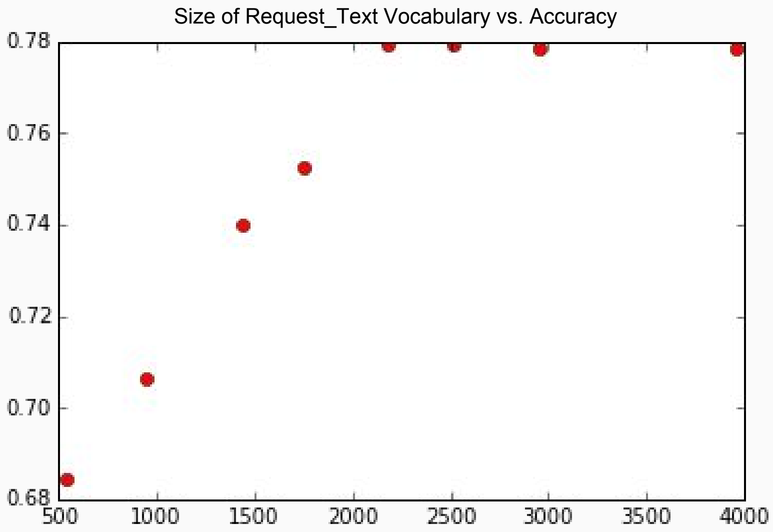


Feature Engineering - Textual Features

- Preprocessing
  - Lowercase Transformation
  - Removing Punctuations
  - Removing Digits

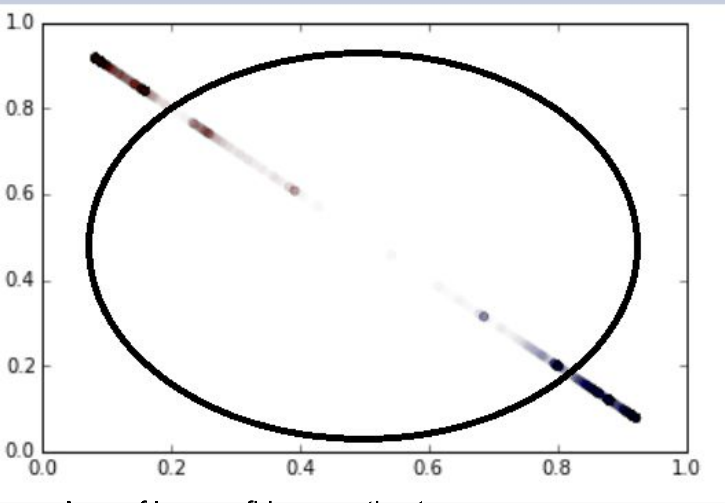
- Tokenization:
  - CountVectorizer
  - LemmaTokenizer

- L1 Pruning: 14,000 → 2,000

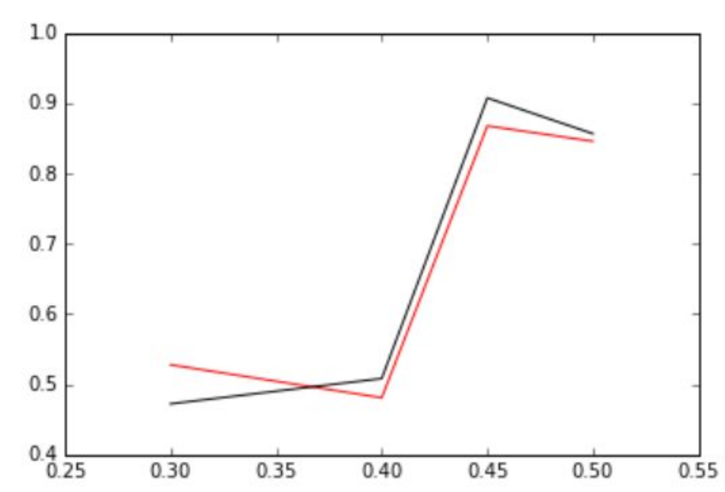


Failure Analysis

- Train a separate classifier on the “low-confidence” classifications, and funnel only low-confidence data to it



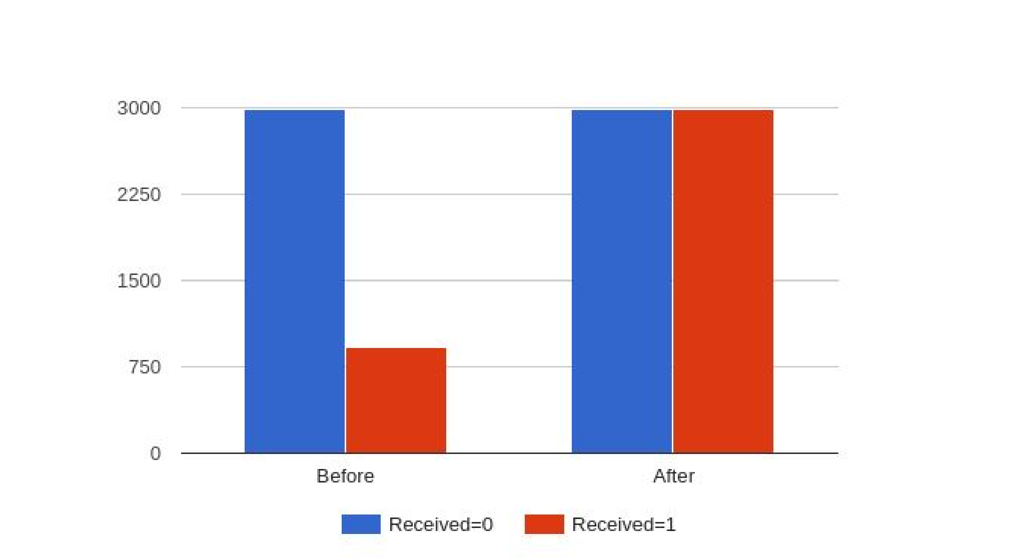
Area of low-confidence estimates



Threshold vs accuracy on low-confidence classifier

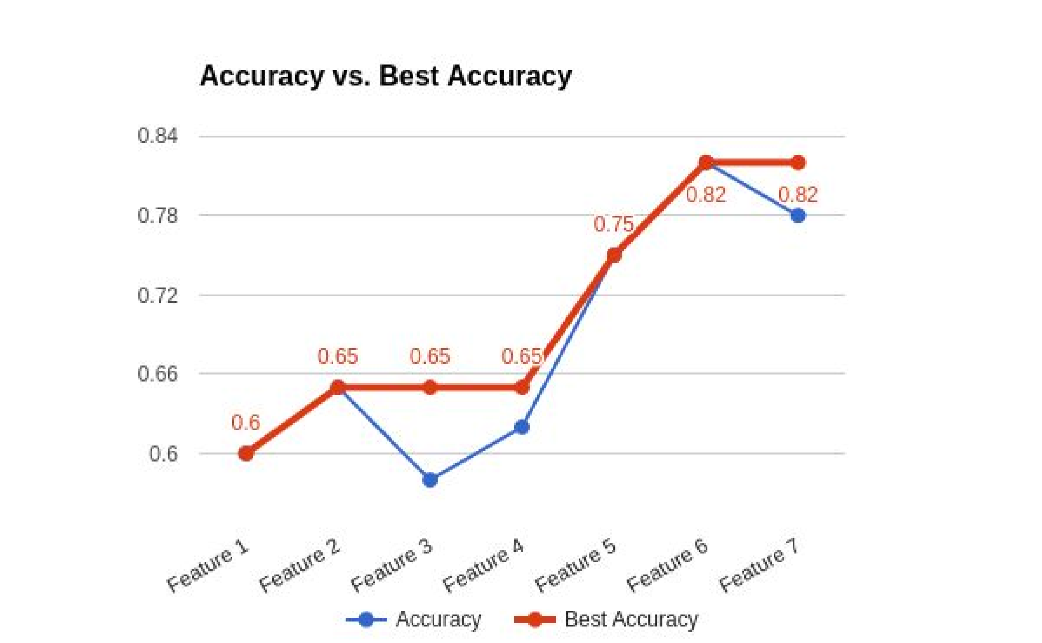
Upsampling vs. Downsampling

- 24% of positive requests
- Under-representation of positive cases
- Also tried downsampling negative cases but no clear difference



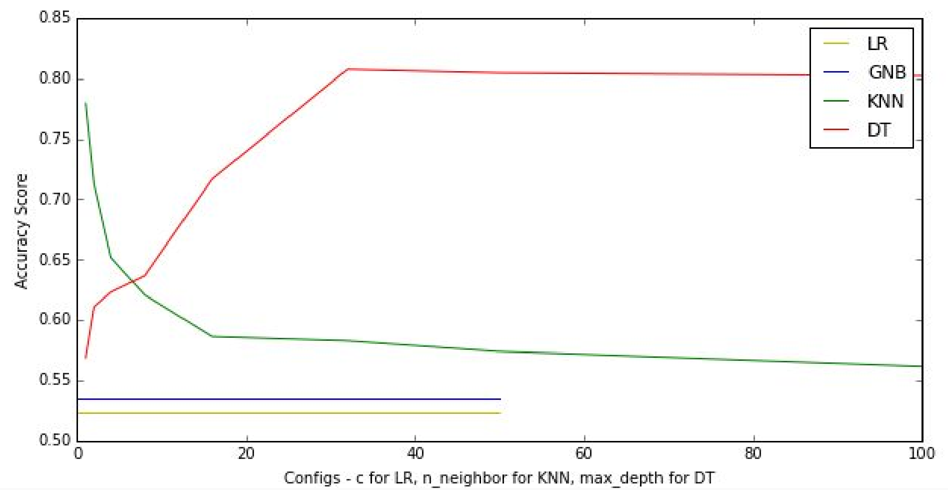
Iterative Feature Selection

- A module to evaluate numerical features sequentially
- Accepts new feature if it improves test accuracy
- Keep only existing features if otherwise

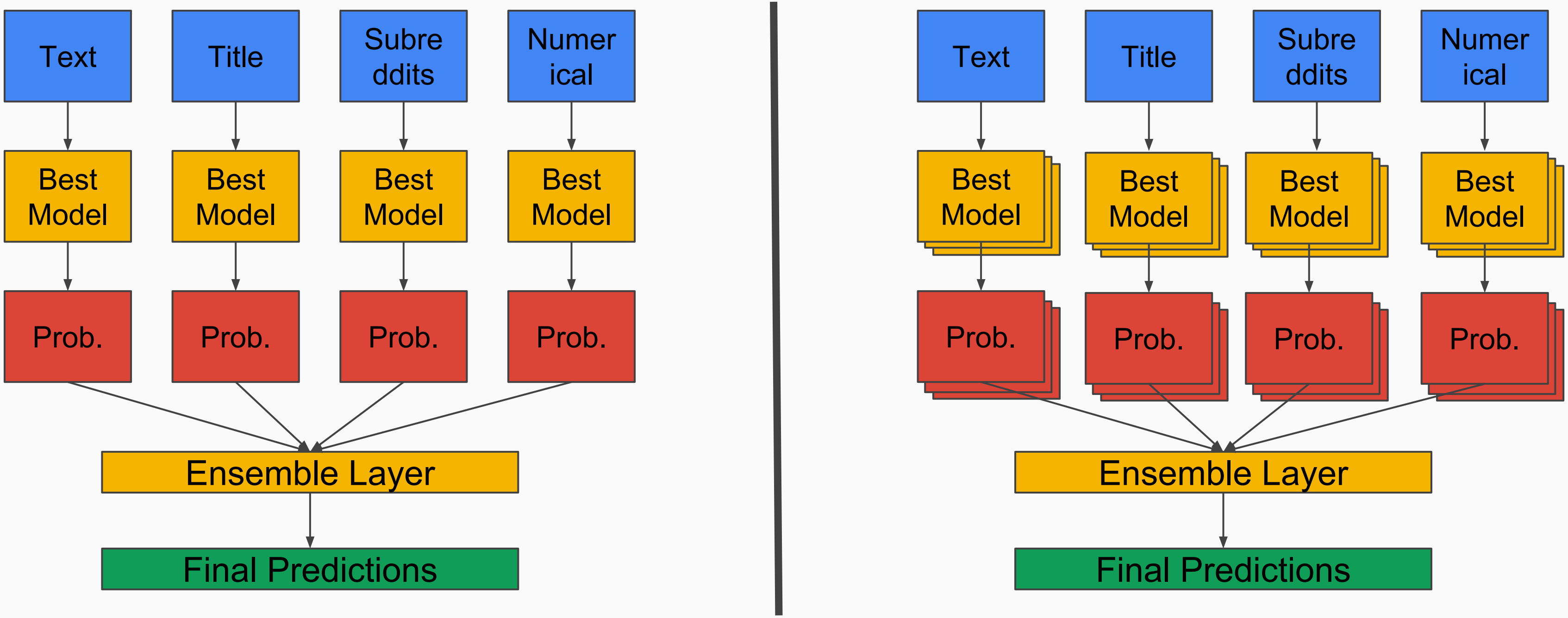


Model Selection

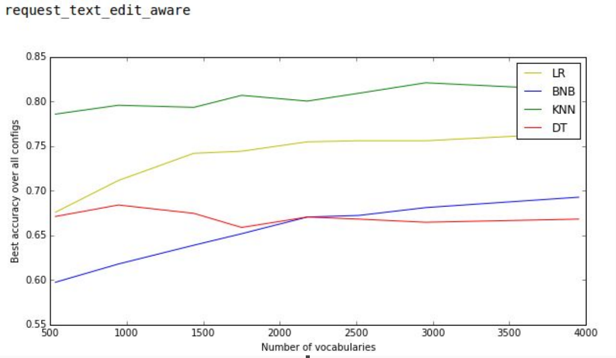
- LR, GNB (BNB), KNN, DT
- Iterate through a wide range of configs
- Include the feature if improves test accuracy
- Also select the best model and config



Two Types of Model Ensembling

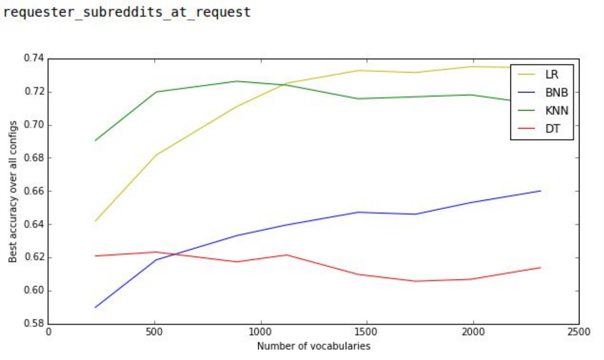


Example Ensembling



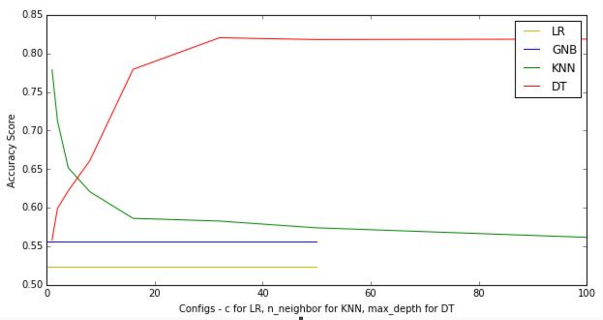
request_text_edit_aware

```
[[ 0.45117338,  0.54882662],
 [ 0.46182652,  0.53817348],
 [ 0.56005853,  0.43994147],
 [ 0.42813328,  0.57186672],
 [ 0.19566884,  0.80433116]]
```



requester_subreddits_at_request

```
[[ 0.77892174,  0.22107826],
 [ 0.72267351,  0.27732649],
 [ 0.11707534,  0.88292466],
 [ 0.90084274,  0.09915726],
 [ 0.22671418,  0.77328582]]
```



```
[[ 0.93408343,  0.06591657],
 [ 0.47370213,  0.52629787],
 [ 0.39715203,  0.60284797],
 [ 0.28550043,  0.71449957],
 [ 0.72490577,  0.27509423]]
```

93%

Final Accuracy

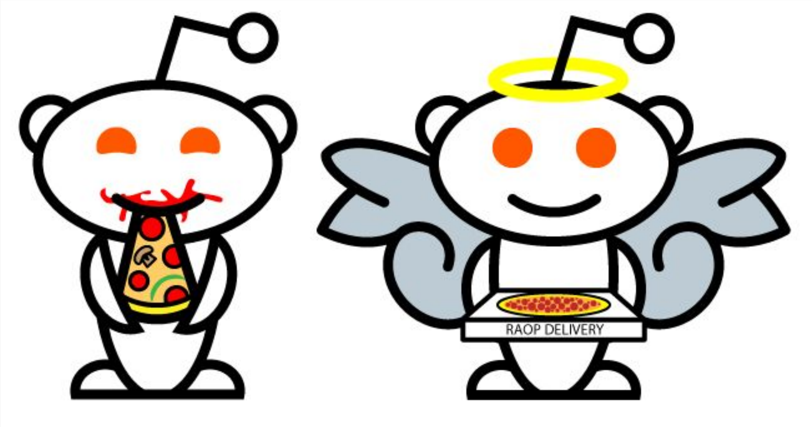
- L1 Pruning for tokenized text / title / subreddits
- Upsampling of positive cases
- Iteratively evaluate and select features
- Fit features individually and ensemble the predictions

Lessons Learned

- **Tossing in features haphazardly makes things worse!**
- **Find the model the works best for a specific feature type and then ensemble!**
- **Balance the classes!**
- **Reduce dimensionality!**

Also,

- GitHub sucks for PYNB co-authoring!
- Good coordination prevents integration errors.
- Translation to Industry Lesson: Having the coders of key sections on hand makes a HUGE difference in resolving bugs (vs. having separate integration teams).



Questions?