Project 3:
Web APIs & NLP

MATT BURKE

GA-DSI-11-DEN

April 24, 2020

AGENDA

- Data
  - Acquisition
  - Preprocessing
- Model
  - Approach
  - Selection
  - Evaluation
- Conclusions & Recommendations

image source

Problem statement

This is a binary classification problem. Scrape data from two subreddits using Pushshift's API. Then use Natural Language Processing (NLP) to train for a classifier model on which subreddit a given post came from.

Data

Data: Scraping

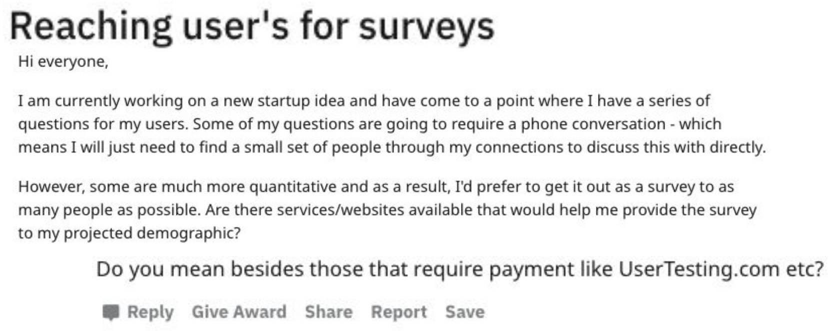
The Subreddits

- r/UXResearch
  - 10.1K members
  - *"A community for sharing and discussing UX research. The goal is to think about UX research broadly and consider studies from related/overlapping disciplines (e.g., market research, medical anthropology, public health, design research). Open to both academic and applied research.*
- r/userexperience
  - 58.6K members
  - *"User experience design is the process of enhancing user satisfaction by improving the usability, ease of use, and pleasure provided in the interaction between the user and the product. User experience design encompasses traditional human–computer interaction (HCI) design, and extends it by addressing all aspects of a product or service as perceived by users."*

Scraping with Pushshift API

```
def pull_text(post_type, subreddit, n_iter)
```

- for Loop based on Epoch time
- post_types
  - submission: 'title', 'selftext'
  - comment: 'body'


Reaching user's for surveys

Hi everyone,

I am currently working on a new startup idea and have come to a point where I have a series of questions for my users. Some of my questions are going to require a phone conversation - which means I will just need to find a small set of people through my connections to discuss this with directly.

However, some are much more quantitative and as a result, I'd prefer to get it out as a survey to as many people as possible. Are there services/websites available that would help me provide the survey to my projected demographic?

Do you mean besides those that require payment like UserTesting.com etc?

**pull_text function adapted from Tim Book*

Data: Preprocessing

- Step 1: basic prep
    - Merge comments dataframes; merge submissions dataframes
    - Remove special characters
    - Dropping nans and [removed]
- Step 2: treat submissions dataframe
    - Concatenate ‘title’ and ‘selftext’ into single feature ‘text’
        - More text data in a single feature
        - Same shape as comments dataframe
- Step 3: prepare dataframe for modeling
    - Concatenate comments and submissions dataframes
    - Remove special characters
    - Create classes for subreddits
        - 1: userexperience
        - 0: UXResearch

13,611

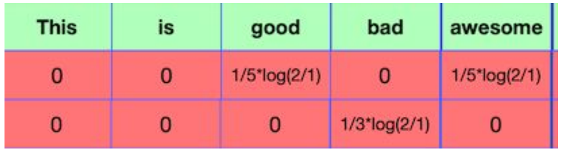
# of documents

Model Selection & Evaluation

Model: Approach

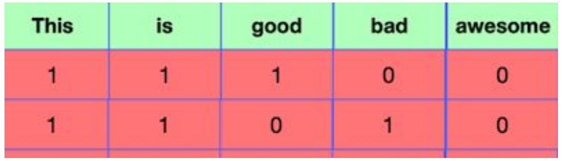
- Models
  - Logistic Regression, Multinomial Naive Bayes, Random Forest, AdaBoost, Voting Classifier
- Vectorizers

TfidfVectorizer



This	is	good	bad	awesome
0	0	1/5*log(2/1)	0	1/5*log(2/1)
0	0	0	1/3*log(2/1)	0

CountVectorizer

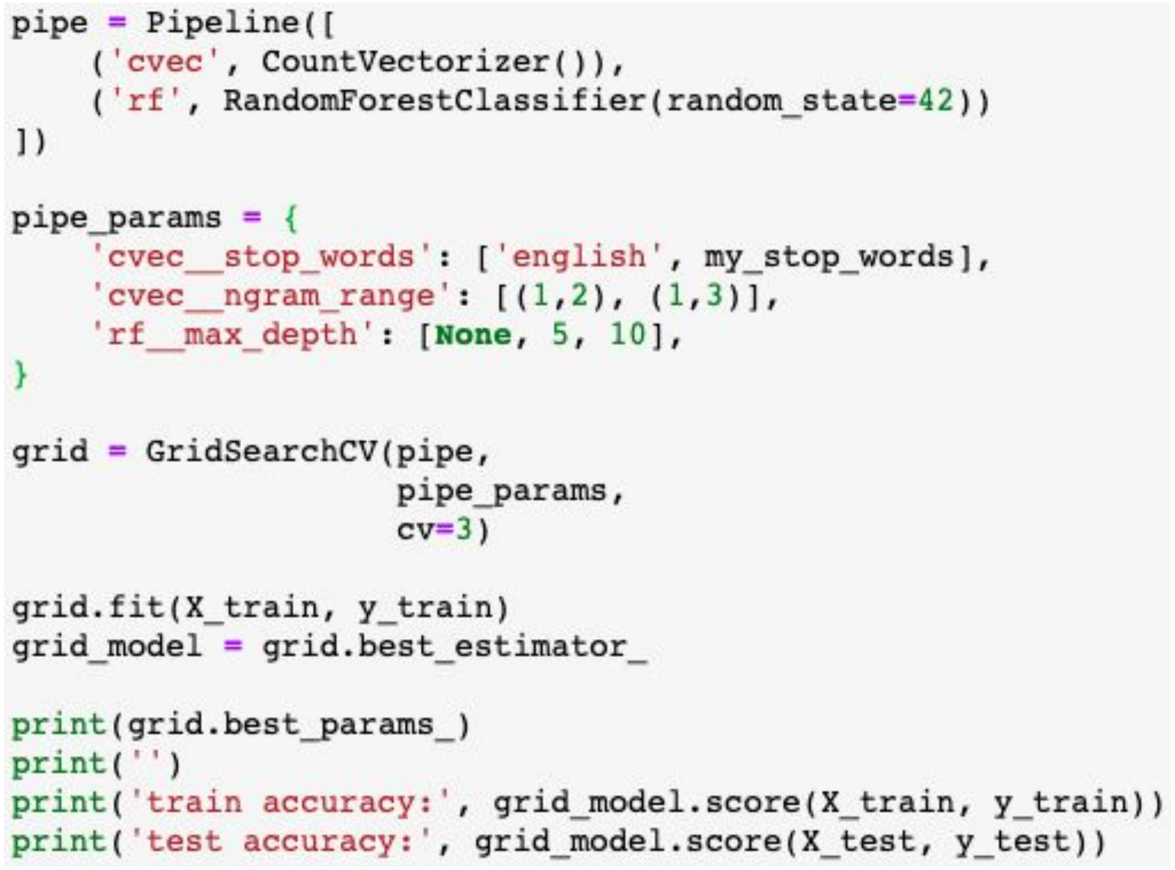


This	is	good	bad	awesome
1	1	1	0	0
1	1	0	1	0

image source

- Pipeline
  - Vectorizer
  - Model
- Parameters
  - *examples*: stopwords, n-gram range, max_features, penalties, alphas
- GridSearch
  - Pipeline + Parameters + 3-5 folds
- ***Fit, score, tune, repeat***

Model: Example

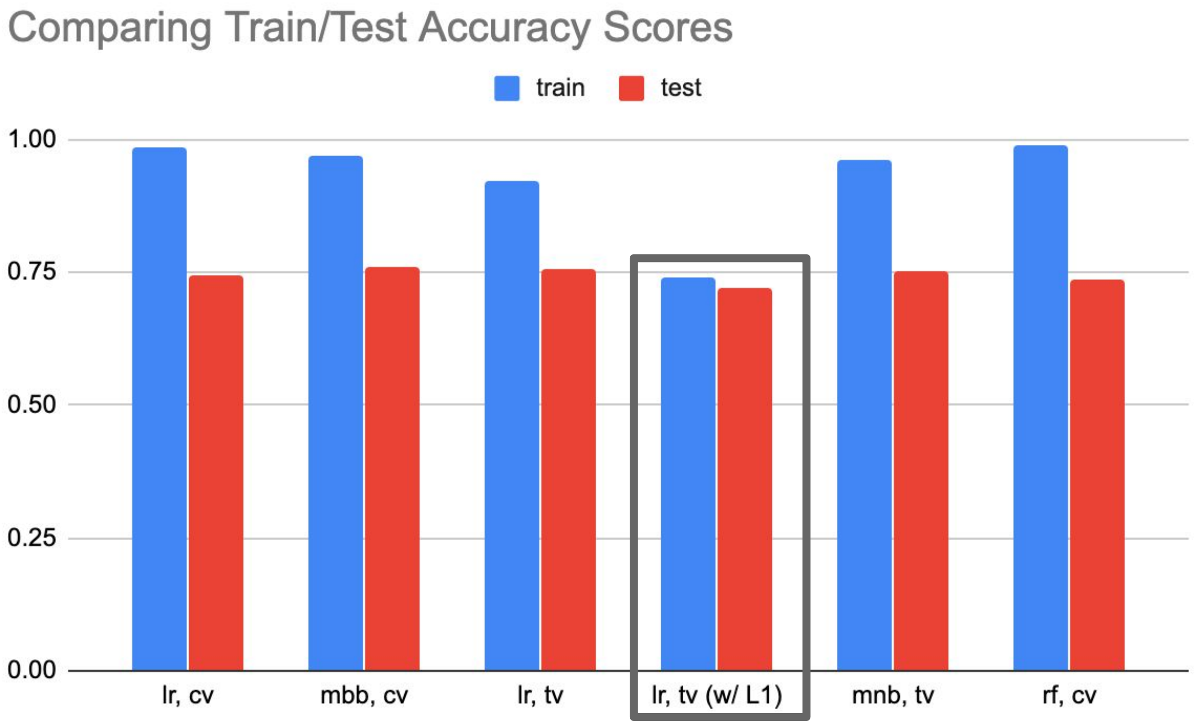


```python
pipe = Pipeline([
    ('cvec', CountVectorizer()),
    ('rf', RandomForestClassifier(random_state=42))
])

pipe_params = {
    'cvec__stop_words': ['english', my_stop_words],
    'cvec__ngram_range': [(1,2), (1,3)],
    'rf__max_depth': [None, 5, 10],
}

grid = GridSearchCV(pipe,
                    pipe_params,
                    cv=3)

grid.fit(X_train, y_train)
grid_model = grid.best_estimator_

print(grid.best_params_)
print('')
print('train accuracy:', grid_model.score(X_train, y_train))
print('test accuracy:', grid_model.score(X_test, y_test))
```

- Pipeline
  - Set vectorizer
  - Define model
- Parameters
  - for vectorizer as well as model

- GridSearch
  - Pipe + param combos
  - define folds
- Fit model on train data
- Define model
- Print
  - Best params
  - Train & test scores

Model: Evaluation



- Examples of train/test scores
- Problem of overfitting
- Average delta was 22 %age point *(excl. best performing model)*

best performing model

Best Performing Model

- Logistic Regression, TfidfVectorizer()
- Parameters
  - n-gram range: (1, 2)
  - stopwords: ‘english’
  - penalty: ‘l1’ (lasso)
- Accuracy scores
  - Train: 74%
  - Test: 72%
  - Compare to baseline accuracy: 53% positive class
- This is okay but not great...

Conclusions & Recommendations

Challenges

- Subreddit similarity
- Tradeoff between class balance and amount of data
  - UXResearch limited to ~6,600 posts
    - 5105 comments
    - 1492 submissions
- Computing power
  - Timeout issues
    - e.g., Logistic Regression model with *solver: saga* and comparing *penalties: elasticnet, l1, l2*
  - Fitting Random Forests with multiple hyperparameters
  - Made it challenging to iterate efficiently

Conclusions

- Surprisingly, “simpler” models such as Logistic Regression performed on par with if not better than more complex models such as Random Forest and VotingClassifier
- Computing power was the biggest constraint
- Though best performing model’s accuracy score was 19 percentage points higher than baseline accuracy, it is not yet optimized

Recommendations

- The data
    - Pull more data from r/userexperience and find sweet spot between tradeoff of class balance vs amount of data
    - Try stemming and/or lemmatizing before passing the dataset through vectorizer
    - Feature engineering (e.g., combining key phrases, manually adding weight to certain words)
- Parameters
    - Experiment with different stopwords
- Computing power
    - Renting additional computing power through a service such as AWS would allow us to fit more complex models more quickly, and iterate appropriately

AMA

(QUESTIONS)