Restaurant Review Sentiment Analysis

Anastasios Grigoriou,
Shota Ebikawa

Presentation Overview

1. Introduction
2. Data Gathering
3. Previously Proposed Solution
4. Current Solution
5. Demo
6. Results & Analysis
7. Conclusions

Introduction

1. Introduction to problem
    - Yelp star rating alone does not describe customers' sentiment regarding the restaurant
    - Yelp cannot filter reviews by category
        - Service
        - Food quality

2. Solution

    - Build a restaurant review service:
        - that outputs customer's sentiment towards the restaurant
        - that categorizes review by service or food quality
            - Search Engine algorithms
            - Natural Language Processing (NLP) algorithms
            - Supervised Machine Learning algorithms

Data Gathering

- One of the most crucial (and difficult) aspects of the project!
- Original training dataset gathered (~1500 reviews, sentiment label, polarity range, etc)
  - <u>Concerns:</u>
  - Small dataset (< 1500 reviews in total), short reviews
  - Positive label skewed (over 70% majority classifier)
- New training data from Kaggle dataset (Yelp 2013)
  - (+200 MB of data, +200,000 user reviews)
  - provided FULL review text, star rating, votes attributes (cool, useful, etc)
- Yelp API to gather testing data
  - Only able to gather 3 latest reviews, resulted in positively skewed dataset
  - Only provides small excerpt of review text
- Zomato API
  - 5 latest reviews with more characters in text review
  - Able to make API calls to get asc and desc order for restaurant reviews (resulted in balanced dataset)

Data Gathering (continued)

- Restaurant Inspection Violation Reviews (~ 67,980 text reviews)
    - Dataset describing arrays of health violation codes (good hygienic practice, preventing contamination ny hands; etc.)
    - 54 different inspection codes
        - 24 inspection codes correlated to service
        - 30 inspection codes correlated to food quality

Previously Proposed Solution

- Read the dataset by implementing the Python library Pandas
- Initialize X: contains column with text reviews
- Initialize Y: contains column with sentiment labels (positive, neutral, negative)
- Preprocessed X with preprocessing function that we built from scratch
    - Implemented Nltk libraries to execute
        - Stopword removal
        - Tokenization
        - Lemmatization
- Implemented TfidfVectorizer for feature extraction
    - Term Frequency-Inverse Document Frequency (TF-IDF) function in sklearn
        - Utilized unigram/bigram and max_df (built-in parameter) for its features
- Split X and Y into training dataset (X_train, Y_train) and testing dataset (X_test, Y_test)

Previously Proposed Solution

- Implemented Multinomial Naive Bayes Classifier
  - Fit the classifier into X_train and Y_train
- Predicted the sentiment labels of data points in X_test
- Compared the predicted results with Y_test

Shortcomings

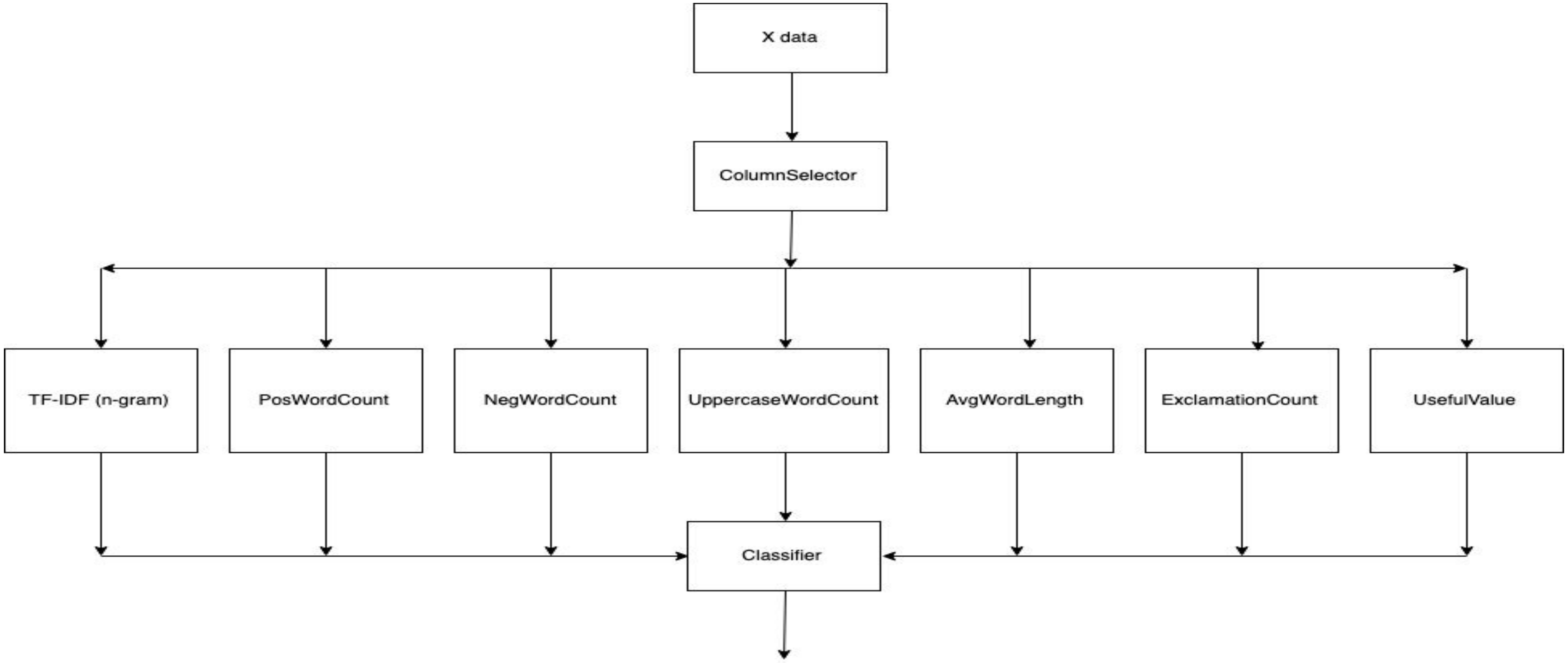
- Accuracy: 75%
- Skewed Dataset
    - Majority classifier of over 70%
        - Positive label overruled negative label and neutral label
        - Negative label and neutral label contains less than 30% of the dataset
        - Produces biased sentiment result
- Unable to implement features outside of built-in parameters of TfidfVectorizer
        - Only maximum df and unigram/bigram
        - Cannot add custom features
- Runtime of the program is ridiculously slow
    - All the procedure ran synchronously

Current Solution

Sentiment Analysis

- Read in training dataset, convert to DataFrame format with all separate columns
  - drop votes column, add 3 columns 'cool', 'useful', 'funny' from original 'votes' column
- Separate data by 1, 3, 5 star rating
  - **assumption**: 1 star review has 'negative' sentiment, 5 star review has 'positive' sentiment
  - **encoder**: 1 → 'negative', 3 → 'neutral, 5 → 'positive'
- **X**: Get random samples of rows for each star rating (N = 5000, ⅓ split for each class). Take only 'text', 'useful' columns
- **Y**: Same random sampling strategy. Use only 'rating' column (sentiment label)

Feature Pipeline



Sentiment Analysis

- **Feature Pipeline**:
  - ColumnSelector: for each feature, select the column used from original X data (ex: ‘text’ for tfidf, ‘column’ for useful value extractor) for fit and transform later on
  - FeatureUnion: feature union each individual feature pipeline in order to run features in parallel, combine at end
- **Feature Engineering:**
  - TfidfVectorizer (unigrams/bigrams, default pre-processing, ‘english’ stop word removal)
  - PositiveWordCountExtractor (counts # of words for given review text in pos_word.txt file list)
  - NegativeWordCountExtractor (counts # of words for given review text in pos_word.txt file list)
  - UppercaseWordCountExtractor (counts # of fully uppercased words in given review text)
  - AverageWordLengthExtractor (supplement neutral label class)
  - ExclamationPointCountExtractor (correlates to stronger sentiment in review text)
  - UsefulValueExtractor

Sentiment Analysis

- **Classifiers:**
  - Multinomial Naive Bayes (baseline performance)
  - Support Vector Machine
  - Random Forest Classifier
  - Logistic Regression
- **Prediction:**
  - For each classifier: Split into X_train, X_test, Y_train, Y_test (test_size=0.25)
  - Fit pipeline to X_train and Y_train data
  - Make prediction for pipeline on X_test data
  - Compute accuracy, confusion matrix, and classification report (precision, recall, f1-score)
  - Print 10 results (review text, predicted output sentiment label)
- **Predict Zomato Reviews:**
  - Read in zomato restaurant review data, convert data into DataFrame with ‘text’ column
  - Choose best performing (most accurate) classifier/pipeline and predict sentiment label for review text
  - Print 10 results (review text, predicted output sentiment label)

Category Classification

- Read in Restaurant Inspection Violation Reviews dataset
- Label Encoding
    - Converted restaurant violation code regarding service as 0
    - Converted restaurant violation code regarding food quality as 1
- Initialized two dataframe containing random samples of text reviews (10,000 datapoints each)
    - First one contains text reviews with label of 0 (service)
    - Second one contains text reviews with label of 1 (food quality)
- X: concatenation of two dataframes:
    - With columns of text reviews
- Y: concatenation of two dataframes:
    - With columns of category labels

Category Classification

- Feature Extraction
  - TfidfVectorizer
- Preprocessing
  - TfidfVectorizer’s built-in stopword removal parameter
- Features
  - Ngram range
  - Maximum df
  - Minimum df
- Classifiers
  - Linear Support Vector Machine
  - Multinomial Naive Bayes
  - Random Forest Classifer
  - Logistic Regression

Category Classification

- Initilialized GridSearchCV
  - Interface from sklearn that allows the program to do hyper paramater optimization
- Combine GridSearchCV in Pipeline together to run procedures in parallel
- Hyperparameters to optimize:
  - Ngram range: unigram, bigram, trigram
  - Maximum DF: 0, 0.25, 0.5, 0.75, 1
  - Minimum DF: 1, 2, 3, 4, 5
- Split X and Y into train data and test data
- Fit GridSearch to the training data
- return GridSearchCV with the best performing classifier and the hyperparameter values
- Compute accuracy, confusion matrix, and classification report (precision, recall, f1-score)

Category Classification

- Predict Zomato Reviews
    - Read the zomato restaurant review data
        - Convert them into DataFrame with column “text”
    - Predict 10 text reviews from the dataset using the best performing GridSearch
        - Outputs the text review and it’s category

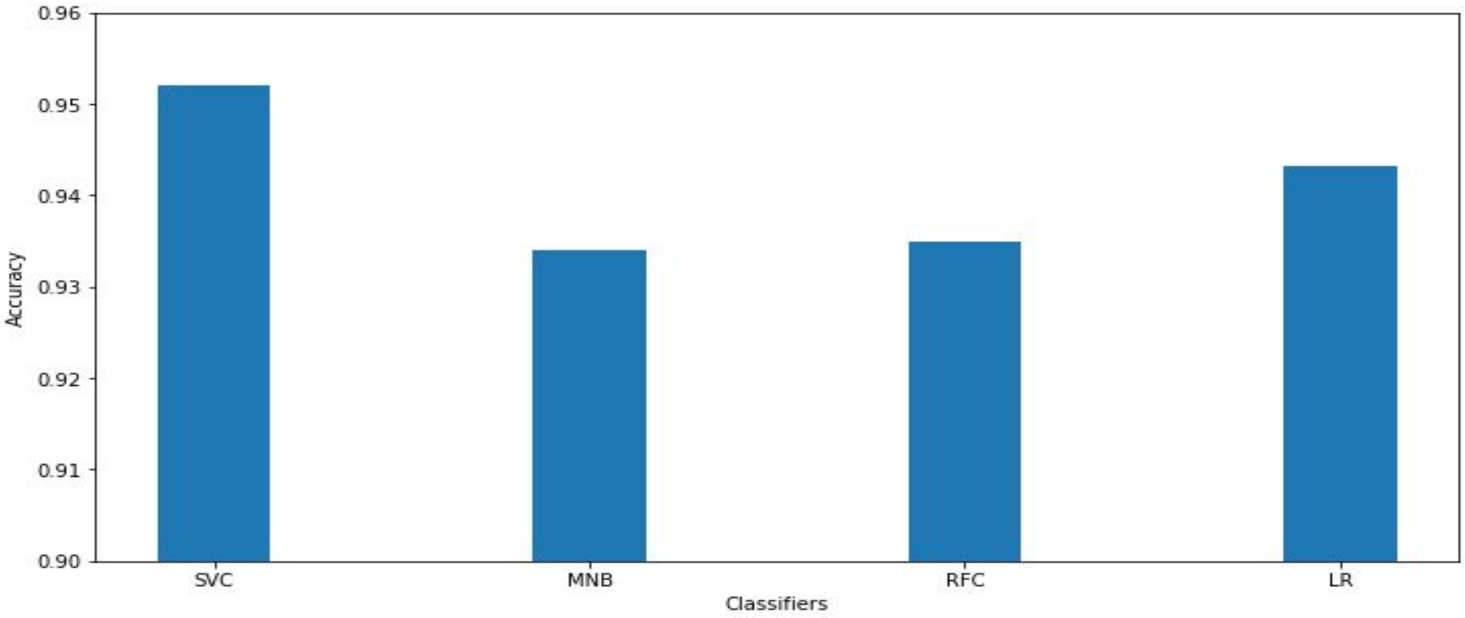
Category Classification (Another Approach)

- Implement unsupervised learning algorithm (soft clustering)
  - Categorize text reviews that :
    - belongs in either a category of service or food quality
    - belongs in both categories of service and food quality
- Tried implementing Fuzzy C Means function from skfuzzy library
  - Unable to add them into GridSearchCV or Pipeline
    - Poor documentation of the library
- Dropped this approach and sticked to the original plan

Demo

Results & Analysis

Category Classifier Results



Category Classifier Results

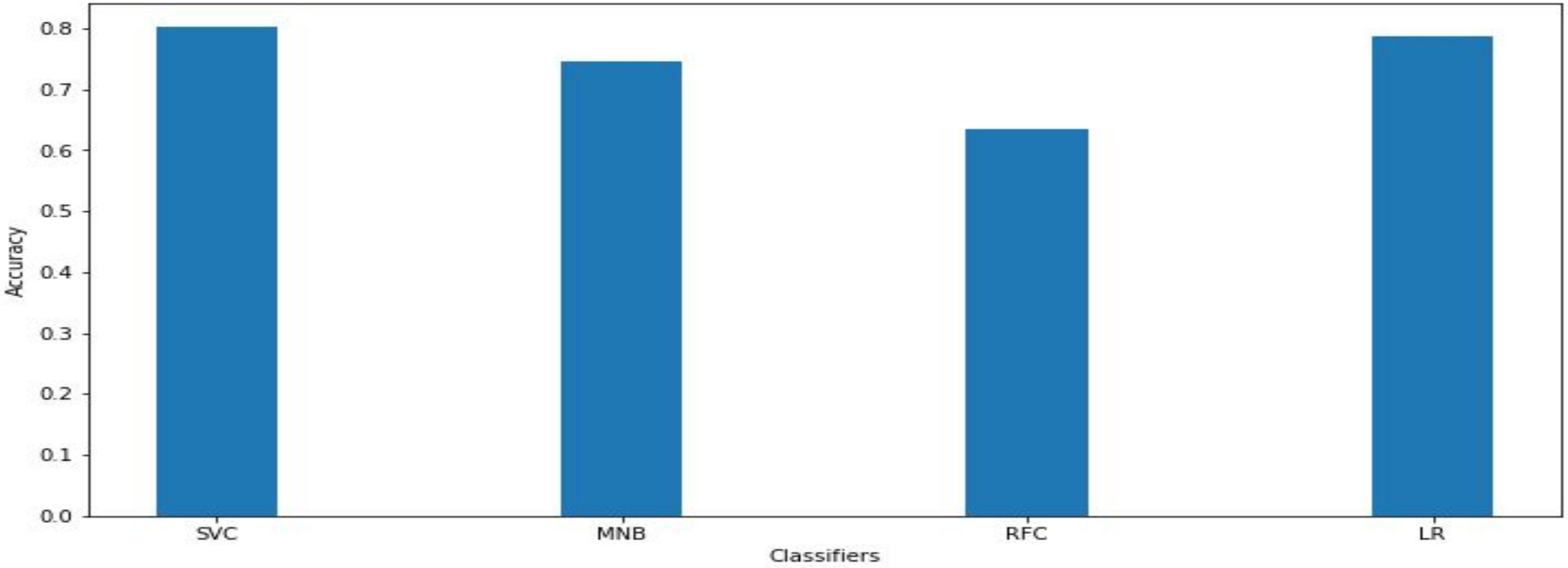
```
Estimator: Support Vector Machine (SVC)
Best params: {'clf__estimator': LinearSVC(C=1.0, class_weight=None, dual=True,
fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0), 'tfidf__max_df': 0.25, 'tfidf__min_df': 1, 'tfidf__ngram_range': (1, 2)}
Best training accuracy: 0.953
Accuracy:
 0.952
Confusion Matrix:
 [[2362  115]
 [ 125 2398]]
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      2477
           1       0.95      0.95      0.95      2523

   micro avg       0.95      0.95      0.95      5000
   macro avg       0.95      0.95      0.95      5000
weighted avg       0.95      0.95      0.95      5000

Test set accuracy score for best params: 0.952
```

Sentiment Analysis Results



Sentiment Analysis Results

```
Evaluation results for classifier: Support Vector Machine (SVC)

Accuracy:
 0.8021333333333334
Confusion Matrix:
 [[1027  139   48]
 [ 149  944  179]
 [  55  172 1037]]
Classification report:
               precision    recall  f1-score   support

           1       0.83      0.85      0.84      1214
           3       0.75      0.74      0.75      1272
           5       0.82      0.82      0.82      1264

   micro avg       0.80      0.80      0.80      3750
   macro avg       0.80      0.80      0.80      3750
weighted avg       0.80      0.80      0.80      3750
```

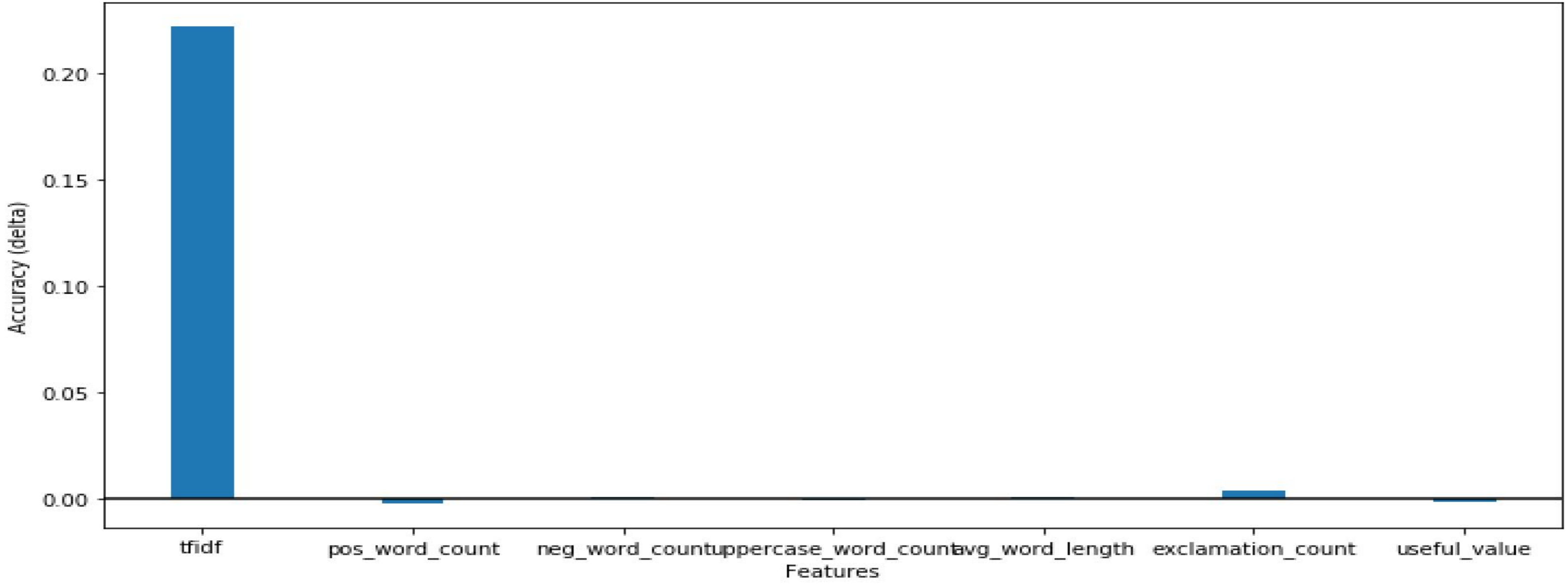
Classifiers

- Best Performing Classifier: Linear Support Vector Machine
- Possible Assumptions why it outperformed the rest:
    - Maximizes its full potential when handling:
        - Dataset with high dimensionality space
            - Easier to form linear separation
        - Dataset that are linear
        - Dataset with large amount of data points

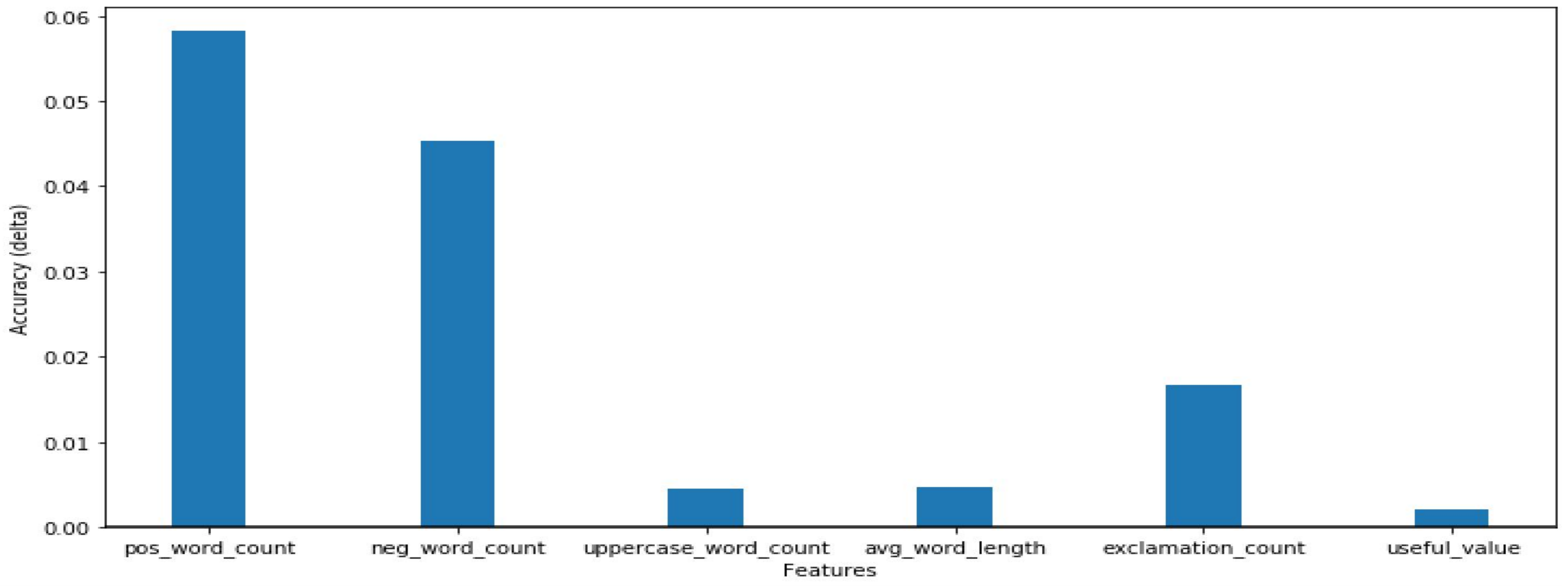
Feature Ablation

- **Total Score:**
  - Run K=10 fold cross validation (instead of fitting the pipeline on the training data)
  - Returns an array of values, each having the score for an individual run
  - Compute mean score when using all features in feature pipeline
- **Score without feature:**
  - Drop one feature (ex: tfidf)
  - Run K=10 fold cross validation and again compute mean score
  - Compute delta between total score and score when taking out that feature
  - Repeat process for each feature (tfidf, pos_word_count, neg_word_count, uppercase_word_count, avg_word_length, exclamation_point_count, useful_value)
- Use Matplotlib to plot accuracy deltas for each feature

Feature Ablation Results



Feature Ablation Results



Conclusions

Ways to improve

- Gather better training dataset to learn a model:
  - Explicitly labeled with sentiment (manual)
  - Utilize more attributes in dataset (other than simply text review)
  - Add more complex features based on different attributes
- Expand feature importance by using WordNet to gather more positive/negative/neutral terms
- Add more categories for category classifier to predict
  - Ex: scenery, price, dessert, etc

What we learned

- Can do full-pipeline of supervised learning techniques
- Gathering ‘good’ data is a difficult task
- Surprisingly accurate results from standard pipeline
- Features are very hard to engineer in order to make a big impact on performance
- Choosing the right classifier for your dataset can also make a big difference
- Jump started learning of machine learning techniques and methodology

Any Questions?

Thanks!