CFPB Data Case Study

Nick Walker

Overview

- Data
- Preprocessing
- Vectorization
- Feature Selection
- Model Training
- Model Results
- Thoughts & Conclusion

The Data

- 268,359 documents
- 7 classes of data, unevenly distributed

bank_service	20071
credit_card	29553
credit_reporting	81230
debt_collection	61457
loan	31035
money_transfers	4734
mortgage	40279

Modules and Setup

- Numpy and Pandas, for useful data structures
- NLTK, presently only for stopwords, but generally it’s a useful library for basic NLP functionalities
- Scikit-learn, which provides useful implementations of vectorizers and classifier models
- Other native Python libraries such as random and collections, for shuffling data and bookkeeping with defaultdict

Preprocessing - First Steps

- Lowercase all documents (case sensitivity is not likely important for this task)

- Removal of Stop Words (uninformative words e.g. “the”, “and”, “of”)
- Removal of non-word characters and punctuation
- Remove anonymized personal information (e.g. XXX-XXXX as a phone number)

Vectorization

Possible Choices: One-Hot encoding, binary vectorization, word count, tf-idf

Chosen solution: Count Vectorization of unigrams and bigrams (start with n_features=50000, then pare it down with feature selection)

Rationale: Simplicity, easier to select features in sklearn implementation

Feature Selection - Methodology

- **Reasoning:** We want to restrict our vocabulary to only the more informative words, to reduce noise and maximize separability of the classes
- **How to do it:** We can use Elastic Net Logistic Regression to train initial predictors for each class versus all others (One-Versus Rest). L1 and L2 Regularization parameters encourage our features to be sparse and have small coefficients

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1).$$

Courtesy of Wikipedia

Feature Selection - Process

- Step 1: Select a class we want to predict (e.g. Credit Reporting) as the positive class
- Step 2: Assign all other classes to the negative class
- Step 3: Find how many examples we have of the positive class, then sample an equal number randomly from the negative class
- Step 4: Train a binary classifier with Elastic Net Logistic Regression to learn feature weights
- Step 5: Select features above a feature weight cutoff (presently, where the absolute value is > 0)

Final Model Training

- **Possible Models:** Logistic Regression (variants), Random Forest, Naive Bayes, Support Vector Machine
- (Alternatively, feature selection could be bundled by using a Neural Network)
- **Chosen Solution:** Naive Bayes
- **Rationale:** Simplicity, easy to train, not many parameters to tune, useful simplifying assumptions about the data

Results

- **Training:** Trained the model first on a 50-50 split of the data (randomized), roughly Trained again on an 80-20 split and achieved the same result.
- **Result:** Approximately 80-81% overall accuracy regardless of alpha parameter value in both train-test splits.
- **Conclusion:** Much room for improvement of the model, likely needs refined feature selection and/or exploration of other classification models.

Limitations of the Presented Model

- Evaluated using 50-50 and 80-20 train-test splits of the data for training rather than full k-fold cross validation (time constraint)
- Could have evaluated using F1-score, precision, etc. in addition to accuracy
- Could have used grid search for parameters for the Elastic Net feature selection models to optimize their performance (again, time constraint)
- More sophisticated preprocessing could be useful, perhaps including trigrams, etc.
- The final model has a significant amount of accuracy to be gained

Questions?