Classifying Fake Product Reviews Using Support Vector Machine - CS37300
Hyeong Kyun Park

Task: Predictive modeling of where, given reviews of products in several categories, in the form of both text and rating, develop a predictor that can distinguish real and fake reviews.

Dataset:

1. reviews_test_attributes.csv
   2249 observations, 5 columns
2. reviews_train.csv
   37.2k observations, 5 columns
3. reviews_validation.csv
   999 observations, 5 columns

```
1 file_train.head()
```

	real review?	category	rating	text_
0	1.0	Books_5	5	I usually don't bother reviewing this book, bu...
1	0.0	Clothing_Shoes_and_Jewelry_5	5	This cross is simple but elegant and just the ...
2	1.0	Pet_Supplies_5	4	From my experience,this stuff will last a long...
3	0.0	Kindle_Store_5	4	the mysteries that involved a young woman who ...
4	1.0	Books_5	3	Why do these entertain me? The fact that they...

```
1 file_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37184 entries, 0 to 37183
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   real review?  37184 non-null  float64
 1   category      37184 non-null  object
 2   rating        37184 non-null  int64
 3   text_         37184 non-null  object
dtypes: float64(1), int64(1), object(2)
memory usage: 1.1+ MB
```

All datasets have:

ID (categorical)
- Indexing column

real review? (categorical)
- For reviews_test_attributes.csv: -1 (not determined as a real or fake review)
- For others: 0 or 1 (predetermined as real or fake)

category (categorical)
- Pet_Supplies_5, Books_5, Kindle_Store_5…

rating (numerical)
- Integers 1 to 5 inclusive

text_ (categorical)
- Long sentences reviewing the product

Model & Model Space (Knowledge Representation)

Linear Support Vector Machine

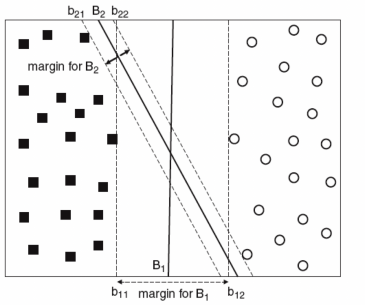
$$y = sign\left[\sum_{i=1}^{m} w_i x_i + b\right]$$

- Model space: set of weights w and b
- Input: SVM works well with unstructured data (Tf-Idf vectorizer is applied to text reviews ; categorical variables, which include rating and category, have been one-hot encoded.)

```
file_train = pd.read_csv("Desktop/fake-reviews/reviews_train.csv",
                         usecols=[0,1,2,3],
                         dtype={'real review?': float,
                                'category': str,
                                #'rating': int,
                                'text_': str})

vectorizer = TfidfVectorizer(max_features=5000)
# Transform text data to list of strings
corpora = file_train['text_'].astype(str).values.tolist()
# Obtain featurizer from data
vectorizer.fit(corpora)
# Create feature vector
X = vectorizer.transform(corpora)
X = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names())
rating = pd.get_dummies(file_train['rating'])
category = pd.get_dummies(file_train['category'])
df = pd.DataFrame(category)
df = df.join(rating)
X = X.join(df)
```

- Output: Classification, so categorical variable. In our situation, it would be whether the review is real or not (0 or 1).



Essentially we want to choose, among many equivalent hyperplanes, the one that maximizes the margin, described in the figure above.

The score function formula

This maximizes the margin subject to constraint that all training data is correctly classified.

- w: weight
- b: y-intercept
- x(i), y(i) = i'th training sample
- ai = coefficient associated with the i'th training sample

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1}^{I} \alpha_i y(i)[x(i) \cdot w + b] + \sum_{i=1}^{I} \alpha_i$$

The search function (how are you finding good models?)

I chose SVM over the original Logistic Regression because they perform well with unstructured data such as text, which is one of our review attributes. The model was able to achieve 0.91 training accuracy.

I did not perform hyperparameter tuning for the SVM, I just ran it with the parameter kernel="linear" and considered all the attributes in the prediction (no feature selection).