IST-736 Text Mining
Homework-4

Sharat Sripada (vssripad@syr.edu)

Introduction

This week's homework is based on a dataset comprising restaurant reviews.

A quick glance of the data shows three columns namely - text (presumably restaurant reviews written by people on platforms like yelp, google etc.) and couple of labels viz. lie and sentiment. It is unknown and perhaps outside the scope of this homework on how the data was labelled - human or machine.

The lie column has two values true or false (abbreviated as 't' and 'f') and the sentiment column has values positive or negative (abbreviated as 'p' or 'n').

Also it is not known if there is any correlation between lie and sentiment and so, for the study of this homework they will be considered as mutually exclusive labels and therefore be split into two datasets with lie and sentiment being the target variables for the models that would be developed.

For the prediction or the machine-learning portion of this homework, Naïve Bayes algorithm (specifically, Naïve Bayes Multinomial method) would be used. To vectorize the data, we will continue to use the TF-IDF library available with sklearn.

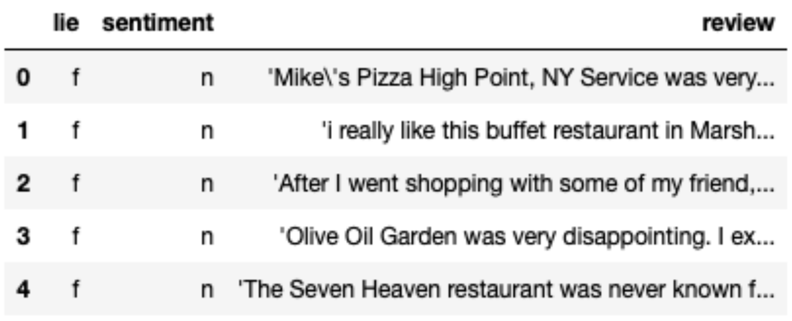
Data source and EDA

The source of the data is a tsv file named 'deception_data_converted_final.tsv'. This file would be translated to a Pandas data-frame using the read_csv() utility with separator as \t (or tab). Once the data-frame is available we study shape and other properties.

Output shown in order below:

```
(92, 3)  <- shows 92 rows

       lie sentiment review
count   92        92     92
unique   2         2     91   <- 2 unique values 't'/'f' and 'n'/'p' as explained in introduction
top      t         n      ?
freq    46        46      2
```

A sample of the data-frame using head property is here:



	lie	sentiment	review
0	f	n	'Mike\'s Pizza High Point, NY Service was very...
1	f	n	'i really like this buffet restaurant in Marsh...
2	f	n	'After I went shopping with some of my friend,...
3	f	n	'Olive Oil Garden was very disappointing. I ex...
4	f	n	'The Seven Heaven restaurant was never known f...

Fig: Raw data-frame loaded from tsv using read_csv

The EDA will mostly comprise the following steps:

1. Look for important trends in text among 92 reviews using wordclouds
   - Cleanup text for stop-words and other punctuation marks etc. so that any vectorization methods w e will employ later will be efficient
2. Plot frequencies of categories in the 'lie' and 'sentiment' column
   - Look for any correlations between the lie and sentiment columns – for instance, are the fake or fals e reviews influenced by negative sentiment or vice-versa.

Wordclouds

The first visualization would be that of the raw text:

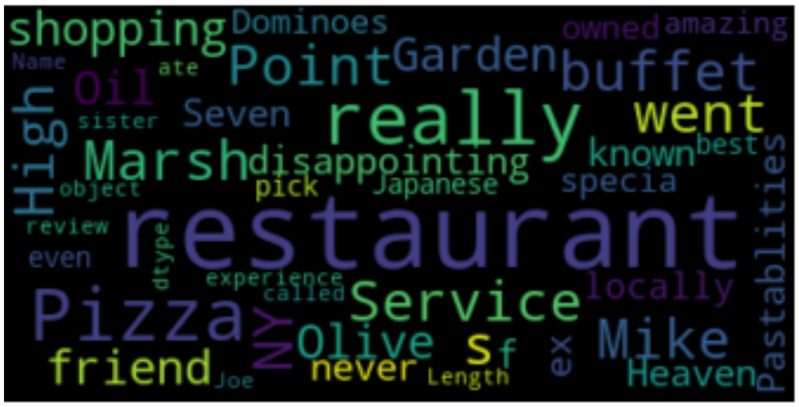


Fig: Word-cloud prior removing stop-words

Some of the words that catch the eye are restaurant, pizza, service, disappointing etc. Use the NLTK stop-words we clean up words and obtain a second word-cloud:

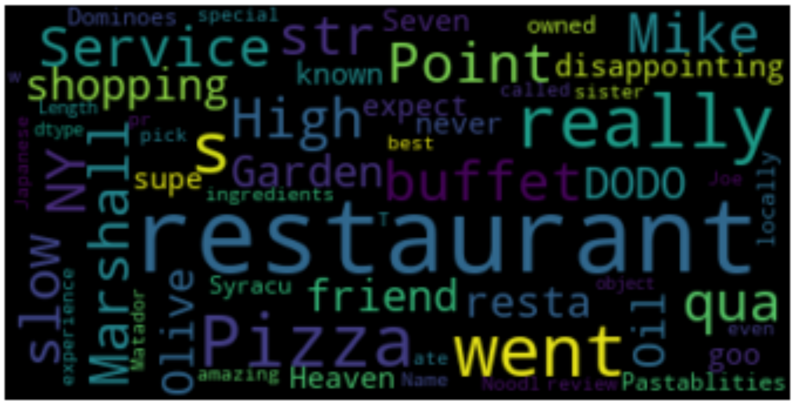


Fig: Word-cloud after removing stop-words

Using stop-words, we reduce the number of words from 6971 to 4176.

And that likely exposes or brings to foreground words like buffet, Marshall, Pastabilities etc., but this is still very subtle. However, filtering for stop-words will likely improve run-times and save us compute when building feature-sets in the vectorization process at a later stage when building models and pipelines.

Frequency Plots

The frequency plots for categories in the sentiment and lie column show balanced data:

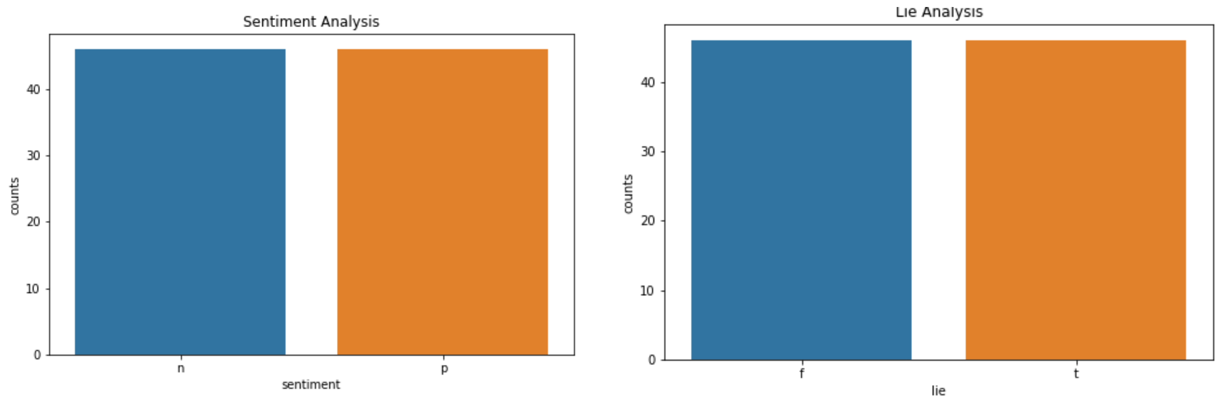


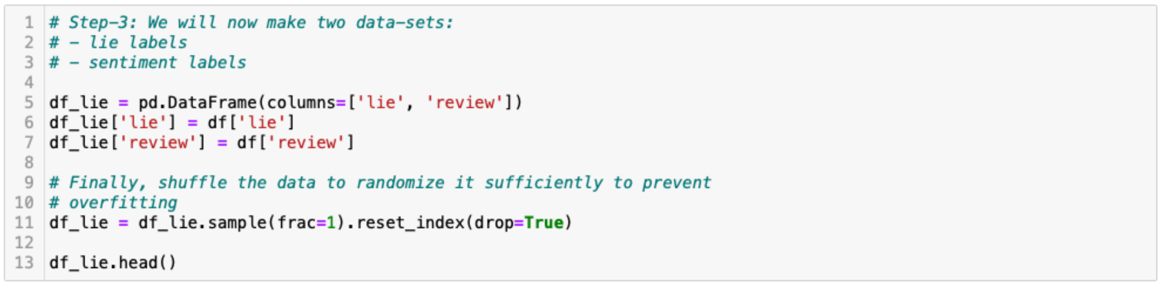
Fig: Bar-plots for categories in lie and sentiment

Further, there seem no correlations to early EDA between lie and sentiment. The results are shown here:

```
Fake and negative: 23
Fake and positive: 23
True and negative: 23
True and positive: 23
```

This seems to indicate that the data has been curated possibly for labelling.

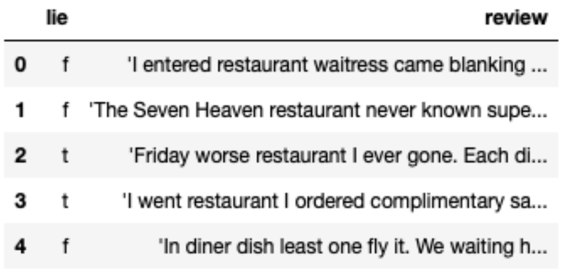
The final step here is to split the dataframe based on multi-labels and develop models independently:



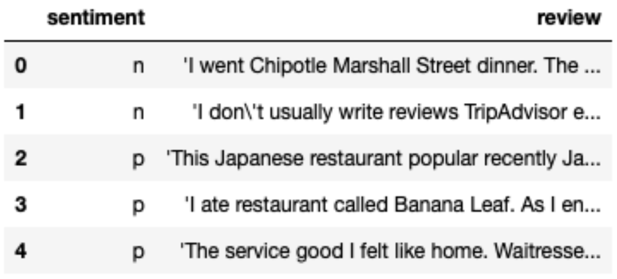
```
# Step-3: We will now make two data-sets:
# - lie labels
# - sentiment labels

df_lie = pd.DataFrame(columns=['lie', 'review'])
df_lie['lie'] = df['lie']
df_lie['review'] = df['review']

# Finally, shuffle the data to randomize it sufficiently to prevent
# overfitting
df_lie = df_lie.sample(frac=1).reset_index(drop=True)

df_lie.head()
```

Fig: New data-frame for lie and sentiment

We take a similar approach for the sentiment column. A sample of the new data-frames are shown here:



	lie	review
0	f	'I entered restaurant waitress came blanking ...
1	f	'The Seven Heaven restaurant never known supe...
2	t	'Friday worse restaurant I ever gone. Each di...
3	t	'I went restaurant I ordered complimentary sa...
4	f	'In diner dish least one fly it. We waiting h...



	sentiment	review
0	n	'I went Chipotle Marshall Street dinner. The ...
1	n	'I don\'t usually write reviews TripAdvisor e...
2	p	'This Japanese restaurant popular recently Ja...
3	p	'I ate restaurant called Banana Leaf. As I en...
4	p	'The service good I felt like home. Waitresse...

Fig: Split data-frames for each of the labels

Classification and Models

In this section, we will present the Naïve Bayes Multinomial algorithm and how we build models with it. For this, we will use sklearn's make_pipeline utility to stack estimators like TfidfVectorizer() and MultinomialNB(). That TF-IDF would be used to vectorize the text data and fed to the Multinomial Naïve Bayes algorithm.

Note, that we will use a K-Fold cross-validation process to carve out train and test data.

Why K-Fold cross-validation method?

Cross-validation is a powerful tool that helps us better use our data (spread) and therefore provides useful information about the performance of the algorithms we may choose.

Here's a good visualization of how cross-validation works:

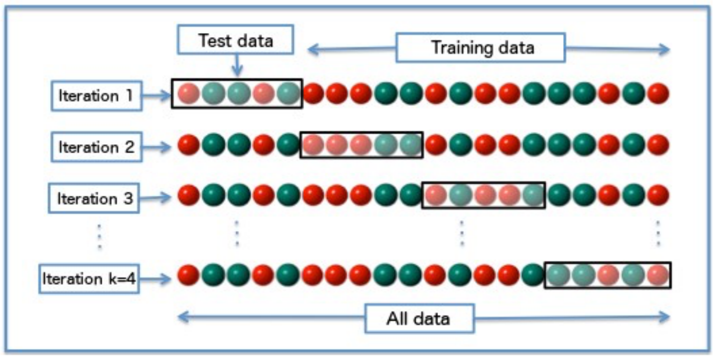
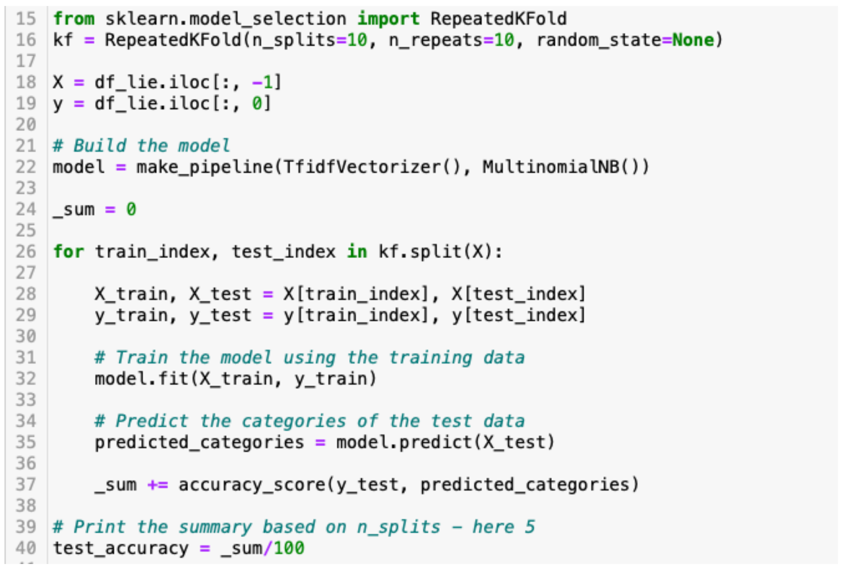


Diagram of k-fold cross-validation with k=4.

Data is split more than once (controlled by folds or commonly known as variable k), than in the classic 1-split 80-20, 90-10 or 70-30 representing train and validation/test data.

Each split or fold is a good representation of the whole data and therefore helps train or develop comprehensive and robust models. Accuracy or performance is subsequently derived by averaging the results over several iterations:



```python
from sklearn.model_selection import RepeatedKFold
kf = RepeatedKFold(n_splits=10, n_repeats=10, random_state=None)

X = df_lie.iloc[:, -1]
y = df_lie.iloc[:, 0]

# Build the model
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

_sum = 0

for train_index, test_index in kf.split(X):

    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Train the model using the training data
    model.fit(X_train, y_train)

    # Predict the categories of the test data
    predicted_categories = model.predict(X_test)

    _sum += accuracy_score(y_test, predicted_categories)

# Print the summary based on n_splits - here 5
test_accuracy = _sum/100
```

Fig: Showing pipelines, split of train/test data and the result aggregation

Also, we specifically use the RepeatedKFold utility so, we can repeat the experiment a few times.

At this stage we will develop two models viz. model for lie and sentiment separately with the same review or text data vectorized using the TF-IDF method.

Results and accuracy

The results can be summarized in the table below:

	Model - Lie (TF-IDF) K-Fold = 10/Repeat - 10	Model – Sentiment (TF-IDF) K-Fold = 10/Repeat - 10
Accuracy	0.47	0.80

Despite broadly taking steps to randomize/shuffle the data at the outset and further using the cross-validation method, we see poor accuracy or overfitting here.

This is possibly because we have low training samples. Also, as seen in the EDA section, the data seems to be too well balanced possibly indicating it was hand-picked and therefore biased or missing samples that can help build better models

Conclusion

In this homework we have demonstrated using methodologies like cross-validation to split data for train/test (rather than take a flat 1:4 test:train ratio) to overcome known issues with overfitting. Overfitting is the process where model fits the training data well but generalizes to unseen data poorly.

Further, we moved to other supervised learning methods like the Naïve Bayes text classification algorithm for the first time in this course. We have also carried forward the TF-IDF method of vectorization.

Despite, taking measures and sufficiently randomizing or shuffling the data-frame the results or average accuracy results are not desirable. We may possibly need more data to train the models so it can predict results reliably.

Importantly, this is a foundation to topic modeling for restaurant reviews (which can be easily extended to reviews like movies etc.) and help build real-time analytic dashboards and derive ratings, overall sentiment etc. so business owners can make changes to improve customer experiences and offer coupons or promotions to disgruntled customers. It can also help flag reviews as fake or false so administrators can act on them and accordingly take them down.