Classification of Hotel Reviews

Author: Haoran Chen, Jiageng Wu

Date: 12/19/2019

Goal

The goal of this project is to create a system that performs binary sentiment analysis on a dataset of hotel reviews, then calculate a rating to each hotel and do something more on it

Design and task division

Haoran Chen will implement the following modules: the comparison of precision of various classifiers(compare_classifiers.py), and feature visualization of the average rating by state(visualization.py)

Jiageng Wu will focus on normalizing the data and fitting the data. Some of the reviews doesn't have rating domain, some have mixed the state name and the address, this part could cost hours of work

Part I Preparation of train set and data set

The dataset was found on kaggle, provided by Datafiniti's Bussiness Database, The dataset includes hotel location, name, rating, review data, title, username, and more. What we are dealing with is a free sample of a large dataset. The full dataset is available through Datafiniti.

There are two csv files, 'data/Datafiniti_Hotel_Reviews.csv' and 'data/raw_data.csv'. The first file is a more formalized dataset, containing 10000 reviews of hotels, along with a rating of 1~5 stars, which makes a good train set for the binary sentiment analysis. Here is a snapshot of the file

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	id	dateAddec	address	categories	primaryCat	city	country	keys	latitude	longitude	name	postalCod	provinc	reviews.	review	review	reviews.text
2	AVwc252V	2016-10-3	5921 Valer	Hotels,Hot	Accommo	Rancho Sa	US	us/ca	32.99096	-117.186	Rancho Va	92067	CA	2013-1	5	https:/	Our experience at Ra
3	AVwc252V	2016-10-3	5921 Valer	Hotels,Hot	Accommo	Rancho Sa	US	us/ca	32.99096	-117.186	Rancho Va	92067	CA	2014-0	5	https:/	Amazing place. Every
4	AVwc252V	2016-10-3	5921 Valer	Hotels,Hot	Accommo	Rancho Sa	US	us/ca	32.99096	-117.186	Rancho Va	92067	CA	2015-0	5	https:/	We booked a 3 night A
5	AVwdOclq	2015-11-2	7520 Teag	Hotels,Hot	Accommo	Hanover	US	us/md	39.15593	-76.7163	Aloft Arun	21076	MD	2016-0	2	https:/	Currently in bed writi
6	AVwdOclq	2015-11-2	7520 Teag	Hotels,Hot	Accommo	Hanover	US	us/md	39.15593	-76.7163	Aloft Arun	21076	MD	2016-0	5	https:/	I live in Md and the A
7	AVwdOclq	2015-11-2	7520 Teag	Hotels,Hot	Accommo	Hanover	US	us/md	39.15593	-76.7163	Aloft Arun	21076	MD	2016-0	5	https:/	I stayed here with my
8	AVwdOclq	2015-11-2	7520 Teag	Hotels,Hot	Accommo	Hanover	US	us/md	39.15593	-76.7163	Aloft Arun	21076	MD	2016-0	5	https:/	Beautiful rooms and
9	AVwdOclq	2015-11-2	7520 Teag	Hotels,Hot	Accommo	Hanover	US	us/md	39.15593	-76.7163	Aloft Arun	21076	MD	2016-0	5	https:/	We stayed here while
10	AVwdOclq	2015-11-2	7520 Teag	Hotels,Hot	Accommo	Hanover	US	us/md	39.15593	-76.7163	Aloft Arun	21076	MD	2016-0	5	https:/	I travel a lot with my

In the train set we give each review a sentiment label based on the rating attached on each review. These rating data is given by real users so they precisely conclude the sentiment of the review. In this train set we label a review positive if rating ≥ 4, negative if rating ≤ 3. The labeled data will be stored in a json file: 'data/train_set.json'. Json files are perfectly compatible with the dictionary in python so it is a good choice for data persistence. For each key in the json file, value is a (text, label) tuple, text is the review in string, label is 1(positive) or 0(negative). The train set generation code is in generate_train_set.py.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
30666	3060 Gree	Hotels	Branson	US	36.63277	-93.272	Dockers In	65616	Bull Creek	2016-09-2	2016-11-2	5	The service	good locat	Smithville
30668	3060 Gree	Hotels	Branson	US	36.63277	-93.272	Dockers In	65616	Bull Creek	2016-08-(	2016-11-2	5	The check	Excellent lc	Smithville
30675	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2016-06-2	2016-08-1	5	We had an	Great peor	St. George
30677	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2016-07-(	2016-08-1	5	Long story	Amazing c	Salt Lake C
30678	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2016-06-2	2016-08-1	5	Hotel was i	Very pleas	Tampa
30679	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2012-03-(	2016-08-1	5	great locat	Very pleas	Tampa
30687	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2016-07-(	2016-08-1	5	Was comp	Great stay	Casper
30689	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2016-07-1	2016-08-1	5	We drove t	A new cha	Aurora
30690	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2013-06-1	2016-08-1	5	Really nice.	A new cha	Aurora
30691	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2013-09-1	2016-08-1	5	old french	A new cha	Aurora
30692	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2013-06-1	2016-08-1	5	really nice.	A new cha	Aurora
30693	708 Horizc	Hotels & M	Grand Junc	US	39.1073	-108.544	Days Inn	81506	CO	2016-05-2	2016-08-1	5	We stay at	Excellent p	Milan
30699	3005 Frank	Hotel, Con	Eugene	US	44.04402	-123.048	Candlewoc	97403	OR	2016-04-2	2016-11-(	5	Perfect pla	Great staff	Dallas
30700	3005 Frank	Hotel, Con	Eugene	US	44.04402	-123.048	Candlewoc	97403	OR	2015-02-2	2016-11-(	7.5	Restaurant	Williamette	Dallas
30704	3005 Frank	Hotel, Con	Eugene	US	44.04402	-123.048	Candlewoc	97403	OR	2015-06-1	2016-11-(	8.8	Room serv	Room serv	California

The data set is a larger scale of reviews crawled from several different sources such as tripAdvisor, yellowpages, choicehotels, etc. We found that the data set does not strictly follow its format. For example, some state names are mixed up with the abbreviations; some ratings are 10-based while some others didn't have rating period. So the rating domain is not a general evaluation method, have to ignore it.

During extraction of data set we extract name, address, city, postcode of each hotel, then use a python module: zipcodes to generate the state information, on the right is an example of api of zipcodes. By query of the zipcode digits, we can extract the first match's state abbreviation using the following command: zipcodes.matching('code')[0]['state']

```
>>> from pprint import pprint
>>> import zipcodes

>>> # Simple zip-code matching.
>>> pprint(zipcodes.matching('77429'))
[{'acceptable_cities': [],
  'active': True,
  'area_codes': ['281', '832'],
  'city': 'Cypress',
  'country': 'US',
  'county': 'Harris County',
  'lat': '29.9857',
  'long': '-95.6548',
  'state': 'TX',
  'timezone': 'America/Chicago',
  'unacceptable_cities': [],
  'world_region': 'NA',
  'zip_code': '77429',
  'zip_code_type': 'STANDARD'}]
```

The normalized data has the following format: each hotel has an entry in the json object, containing name, address, city, state, postcode, and all the reviews on this hotel, here is a sample:


```
"5": {
    "name": "Little Paradise Hotel",
    "address": "435 E Avenida Olancha",
    "city": "Palm Springs",
    "state": "CA",
    "postcode": "92264",
    "reviews": [
        "We had a wonderful, relaxing time. The staff were completely attentive and accommodating. We ha
        "We were in Palm Springs on the day the temperature was 123 degrees! Despite that, we had a wond
        "We arrived in Palm Springs not sure what to expect....it's a quiet kinda place in 'low season'.
        "This place really is a little paradise!! A beautiful boutique style hotel with EVERYTHING metic
        "Spent 3 nights at Little Paradise boutique hotel on a mother-daughter trip. Wonderful and relax
        "This little hotel surpassed all my expectations. The rooms are well equiped with a little kitch
        "If you want peaceful and relaxing look no further. Although set up like a single floor motel it
```


Part II Model comparison and selection

When all categorical variables are transformed, and all numerical features normalized, we need to split our data into training and test sets. We'll use 80% of the data to training and 20% for testing,

In this section, we will study six different algorithms, and determine the best at modeling and predicting

our data.

Accuracy measures how often the classifier makes the correct prediction. It's the ratio of the number of correct predictions to the total number of predictions (the number of test data points).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision tells us what proportion of events we classified as a certain class, actually were that class. It is a ratio of true positives to all positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (sensitivity) tells us what proportion of events that actually were the of a certain class were classified by us as that class. It is a ratio of true positives to all the positives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

For classification problems that are skewed in their classification distributions like in our case, accuracy by itself is not an appropriate metric. Instead, precision and recall are much more representative.

Some of the available Classification algorithms in scikit-learn:

- Support Vector Machines (SVM)
- Multinomial Naive Bayes (MultinomialNB)
- Logistic Regression
- K-Nearest Neighbors (KNeighbors)
- Decision Trees
- Random Forest

Support Vector Machines

The strengths of the model are: It works well with no linearly separable data and high dimensional spaces. The main weakness is that it may be quite inefficient to train, so it is no suitable for "industrial scale" applications.

One possible real-world application where this model can be applied is for classifying people with and without common diseases.

It is a good candidate because it is often a quite accurate classifier and works well with binary features and high dimensional datasets.

Multinomial Naive Bayes

Naive Bayes classifier for multinomial models. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression. However, in practice, fractional counts such as tf-idf may also work

Strength: need less training data. Highly scalable. It scales linearly with the number of predictors and data points. Not sensitive to irrelevant features. Weakness: In real life, it is almost impossible that we get a set of predictors which are completely independent.

Logistic Regression

Strengths: Convenient probability scores for observations, efficient implementations available across tools, multi-collinearity is not really an issue and can be countered with L2 regularization to an extent. Weakness: doesn't perform well when feature space is too large, doesn't handle large number of categorical features/variables well, and relies on transformations for non-linear features.

K-Nearest Neighbors

K-NN is a non-parametric algorithm which means there are no assumptions to be met to implement K-NN. Parametric models like linear regression has lots of assumptions to be met by data before it can be implemented which is not the case with K-NN. One of the biggest advantages of K-NN is that K-NN can be used both for classification and regression problems.

Weakness: K-NN works well with small number of input variables but as the numbers of variables grow K-NN algorithm struggles to predict the output of new data point. k-NN doesn't perform well on imbalanced data. If we consider two classes, A and B, and the majority of the training data is labeled as A, then the model will ultimately give a lot of preference to A. This might result in getting the less common class B wrongly classified.

Decision Tree

Strengths: Can work with numerical and categorical features. Non-parametric model: no assumptions about the shape of data. Feature selection happens automatically: unimportant features will not influence the result. The presence of features that depend on each other (multicollinearity) also doesn't affect the quality.

Weakness: prone to overfitting

Random Forests

The strenghts of the model are: It works well with binary features, as it is an ensembling of decision trees. It does not expect linear features. It works well with high dimensional spaces and large number of training examples. The main weakness is that it may overfit when dealing wih noisy data.

One possible real world application where this model can be applied is for predicting stock market

prices.

It is a good candidate because it is often a quite accurate classificator and works well with binary features and high dimensional datasets.

Run compare_classifiers.py and console will print the statistics of those models on the train set. In addition, all the classifiers will be persisted in the '/classifiers' folder, using the joblib module. Of course the vectorizer is also persisted.

```
************ Support Vector Machine Model************
Precision: 0.85
--------------------
              precision    recall  f1-score   support

         neg       0.77      0.65      0.71       559
         pos       0.87      0.92      0.90      1441

    accuracy                           0.85      2000
   macro avg       0.82      0.79      0.80      2000
weighted avg       0.84      0.85      0.84      2000

************** Naive Bayes Model ********************
Precision: 0.82
--------------------
              precision    recall  f1-score   support

         neg       0.76      0.54      0.63       559
         pos       0.84      0.93      0.88      1441

    accuracy                           0.82      2000
   macro avg       0.80      0.74      0.76      2000
weighted avg       0.82      0.82      0.81      2000
```

```
************** Decision Tree Model ******************
Precision: 0.74
--------------------
              precision    recall  f1-score   support

         neg       0.54      0.49      0.51       559
         pos       0.81      0.84      0.82      1441

    accuracy                           0.74      2000
   macro avg       0.67      0.66      0.67      2000
weighted avg       0.73      0.74      0.74      2000

************** Random Forest Model ******************
Precision: 0.79
--------------------
              precision    recall  f1-score   support

         neg       0.68      0.46      0.55       559
         pos       0.81      0.92      0.86      1441

    accuracy                           0.79      2000
   macro avg       0.75      0.69      0.70      2000
weighted avg       0.78      0.79      0.77      2000
```

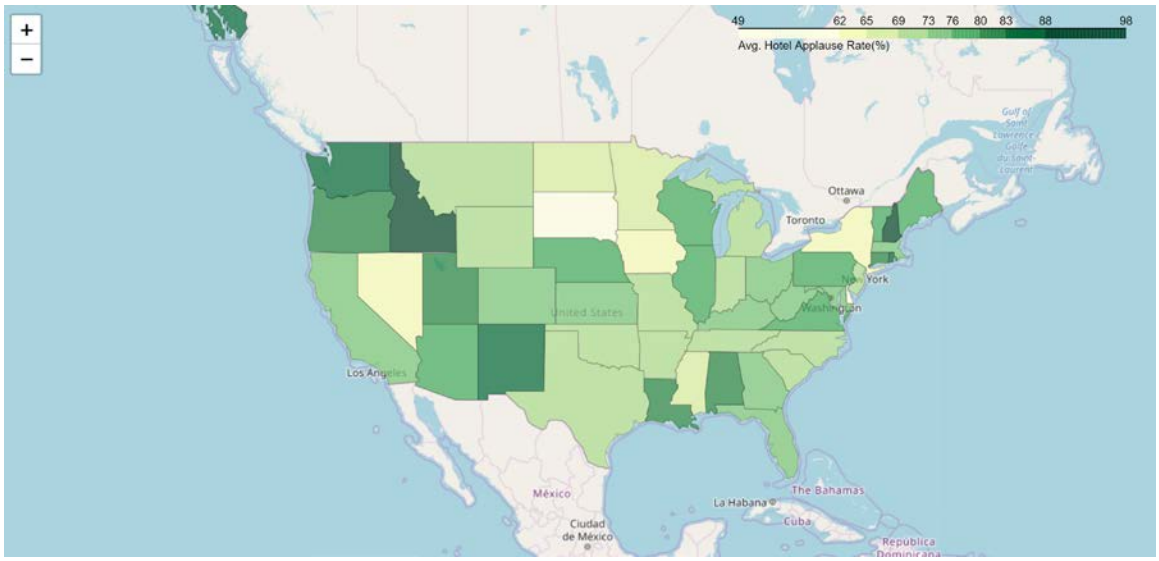
```
************ K-nearest Neighbors Model **************
Precision: 0.76
--------------------
              precision    recall  f1-score   support

         neg       0.78      0.19      0.31       559
         pos       0.76      0.98      0.85      1441

    accuracy                           0.76      2000
   macro avg       0.77      0.59      0.58      2000
weighted avg       0.76      0.76      0.70      2000

************** Logistic Regression ******************
Precision: 0.84
--------------------
              precision    recall  f1-score   support

         neg       0.81      0.57      0.67       559
         pos       0.85      0.95      0.90      1441

    accuracy                           0.84      2000
   macro avg       0.83      0.76      0.78      2000
weighted avg       0.84      0.84      0.83      2000
```

In view of the statics, support vector machine model has the best precision on the train set.

Part III Visualization of average rating per state

For each hotel, add the total review counts to the state's total count, add positive review count to state's applause count, at last calculate each state's applause rate, then use pandas to generate dataframe and pass the json file to the folium module.

Use the folium module to generate a html file, showing the average applause rate of each state:



To understand the data, we tried linear regression between the applause rate and the population of

each state. Run correlation.py to print the score.

The score turned out to be 0.01, seems that these two data has no correlation.

Conclusion

We compared six different models from sklearn to choose the best classier, it turned out that the svm model has the best precision. Then we applied the classifier to a large scale of data to generate applause rate of each hotel, and the average applause rate of each state. However we tried to understand the ratings data, fitting it with the population data of each state, it seems that those two variables are not correlated.