Star ratings from yelp reviews

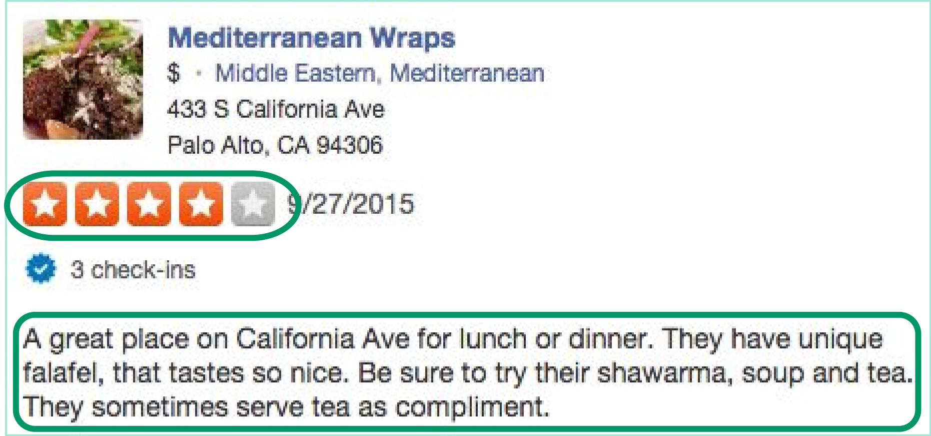
Shilpa Pantula

Outline

1. Hypothesis
2. Data source
3. Feature generation
4. Building model
5. Results
6. Next steps

Hypothesis

Can we predict star rating from review text?



Mediterranean Wraps

$ · Middle Eastern, Mediterranean

433 S California Ave
Palo Alto, CA 94306

★★★★☆ 9/27/2015

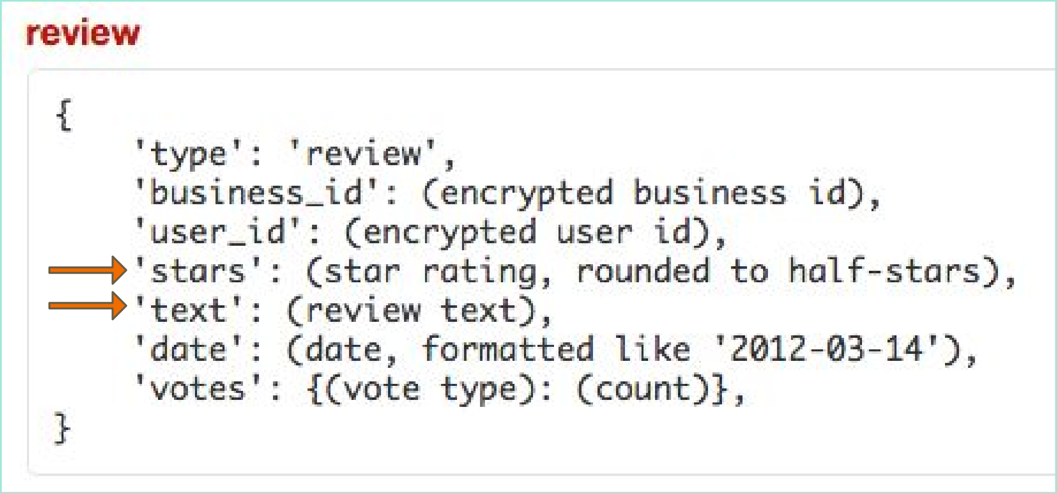
3 check-ins

A great place on California Ave for lunch or dinner. They have unique falafel, that tastes so nice. Be sure to try their shawarma, soup and tea. They sometimes serve tea as compliment.

Data source

Yelp dataset challenge: Round 6: Las Vegas

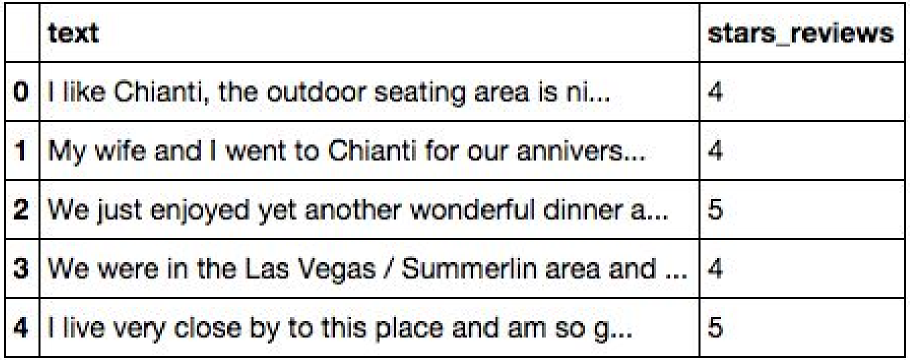
http://www.yelp.com/dataset_challenge



review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

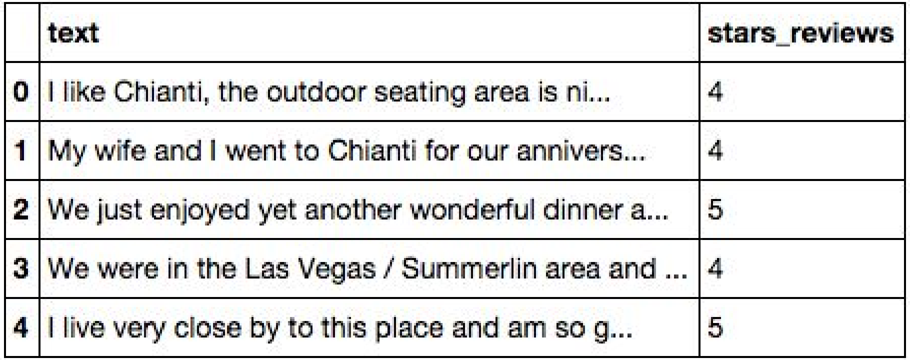
Understanding data



	text	stars_reviews
0	I like Chianti, the outdoor seating area is ni...	4
1	My wife and I went to Chianti for our annivers...	4
2	We just enjoyed yet another wonderful dinner a...	5
3	We were in the Las Vegas / Summerlin area and ...	4
4	I live very close by to this place and am so g...	5

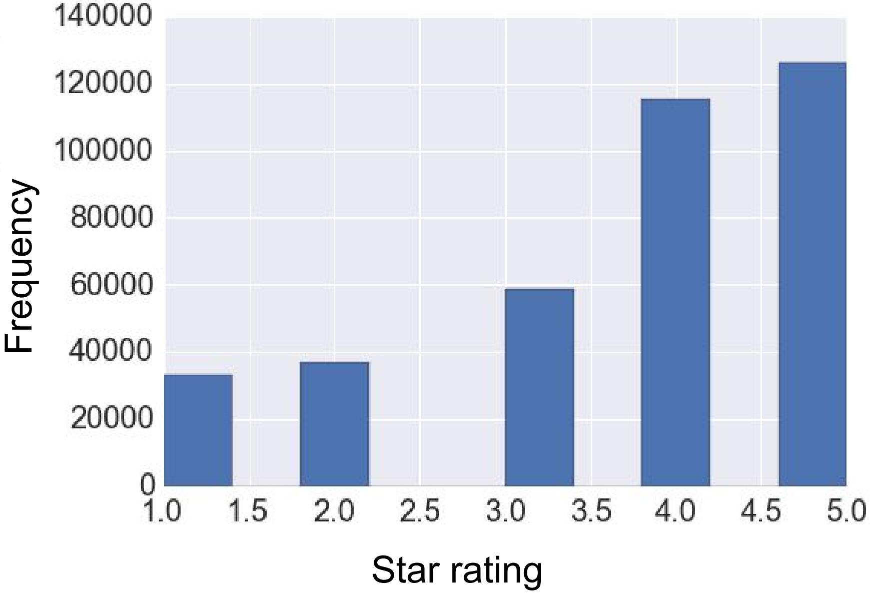
Total of 370,193 records

Understanding data



	text	stars_reviews
0	I like Chianti, the outdoor seating area is ni...	4
1	My wife and I went to Chianti for our annivers...	4
2	We just enjoyed yet another wonderful dinner a...	5
3	We were in the Las Vegas / Summerlin area and ...	4
4	I live very close by to this place and am so g...	5

Total of 370,193 records



Sentiment analysis

vaderSentiment library gives polarity scores of a sentence

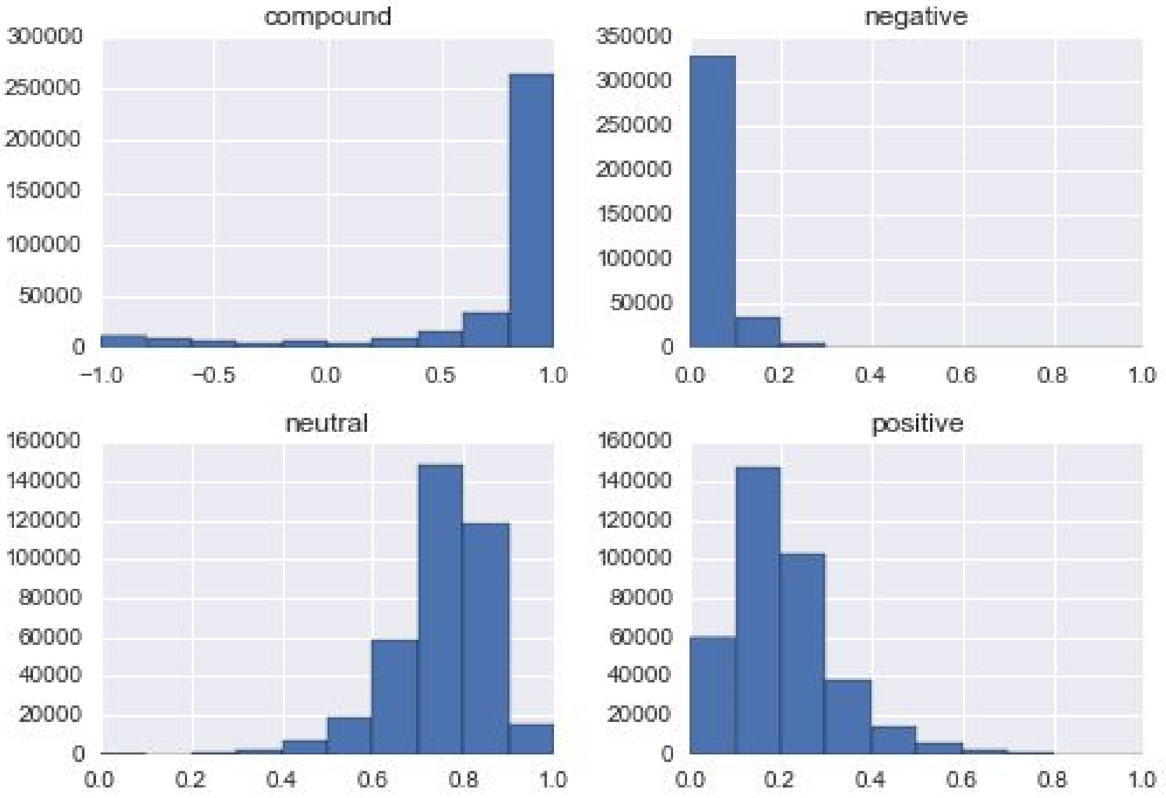
```
from vaderSentiment.vaderSentiment import sentiment
```

s1 = 'The book is good.'

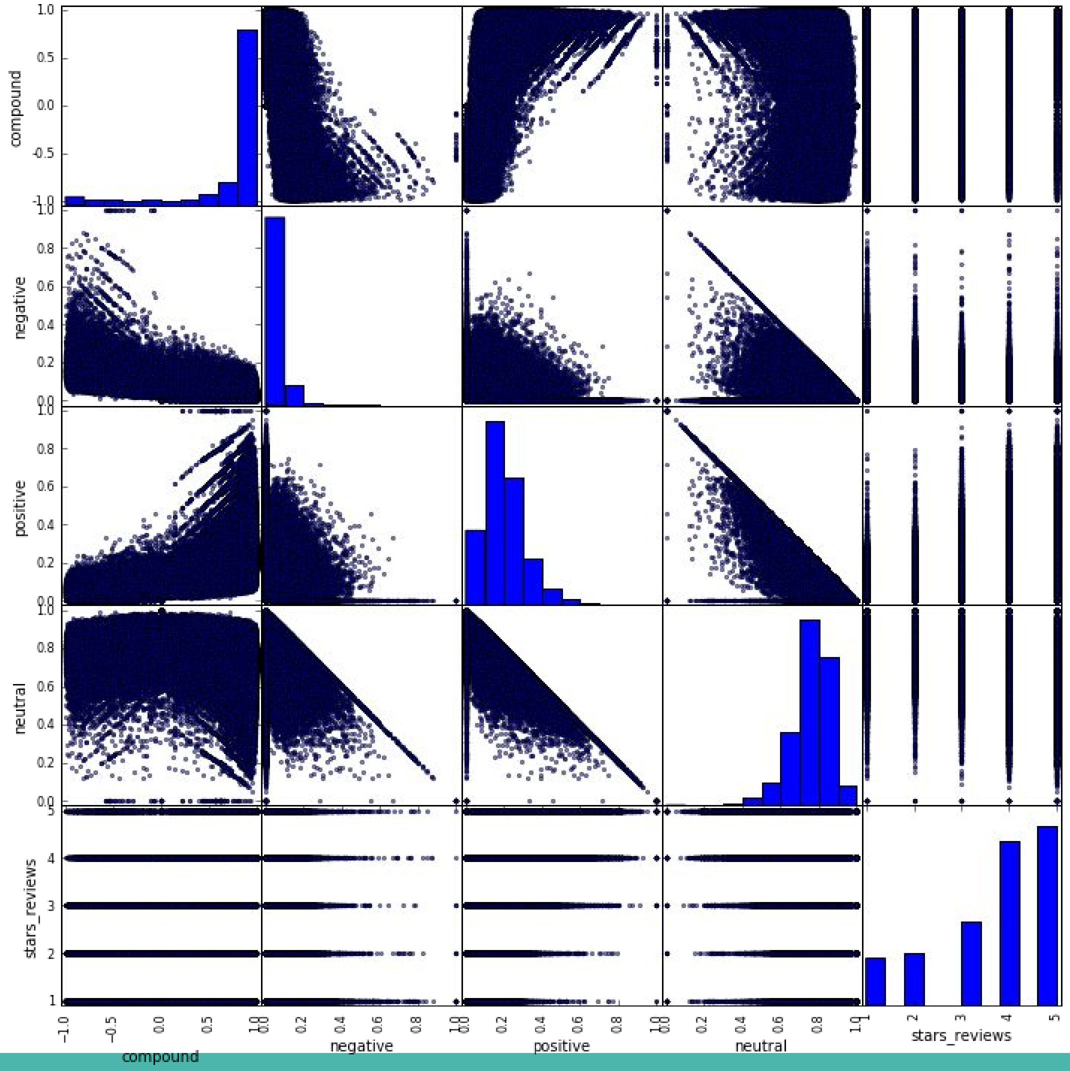
s2 = 'The book is bad.'

	compound	neg	neu	pos
s1	0.4404	0.0	0.508	0.492
s2	-0.5423	0.538	0.462	0.0

Histogram of polarity scores

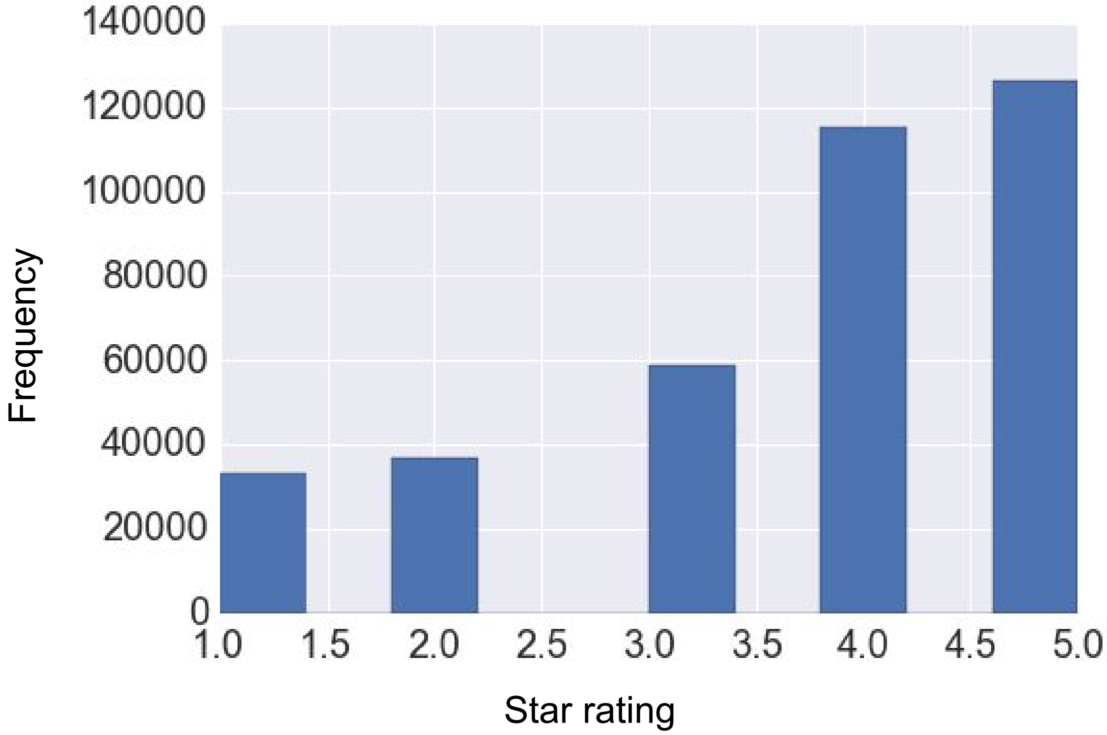


Scatter plots



Classification results

Classifier	**Score**
Knn(5)	0.39
GaussianNB	**0.42**
LogisticRegression(C=2)	**0.422**
RandomForestClassifier(n_estimators=20)	0.41
LinearSVC(C=2)	**0.42**
tree.DecisionTreeClassifier(max_leaf_nodes=10)	**0.42**


140000
120000
100000
80000
60000
40000
20000
0
Frequency
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Star rating

Regression

Training model twice:

1. Train on X_train
2. Second time, train on the error

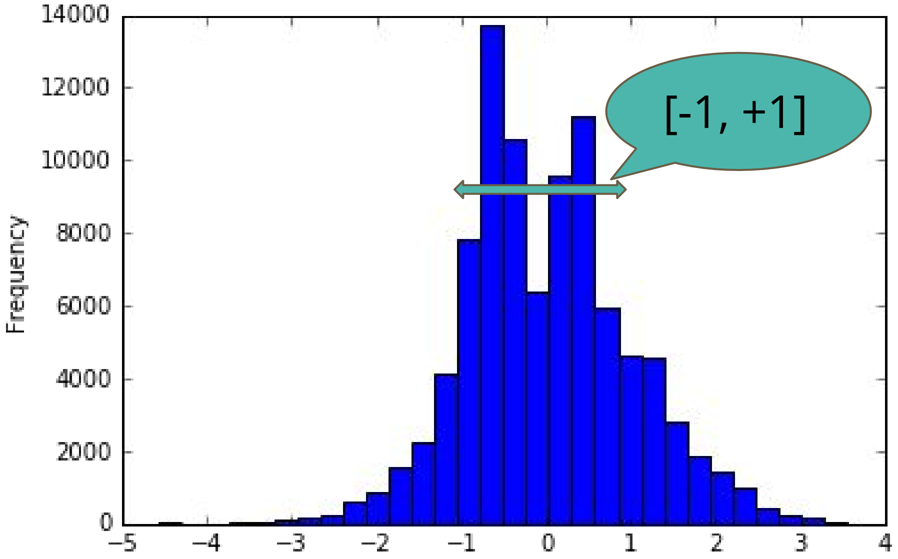
Regression

```
sk_lrn_model = LinearRegression()

sk_lrn_model.fit(X_train, y_train)

y_train_err = y_train - sk_lrn_model.predict(X_train)
```

Regression

```
second_model = KNeighborsRegressor(500)

second_model.fit(X_train, y_train_err)

y_pred = sk_lrn_model.predict(X_test) + second_model.predict(X_test)


from sklearn.metrics import r2_score

print r2_score(y_test.values, y_pred)

>> 0.41
```

Regression results: Histogram of errors



Combining Tf-idf and sentiment scores

Steps:

1. Generate vaderSentiment scores
2. Generate tf-idf matrix
3. Use the generated dataset as features to train new model

Term frequency

Using TfidfVectorizer,

First 100 rows of data, 6398 columns

All data, 370193 rows, ?? columns

Filter out ?

Pseudocode

```
def filter_for_tfidf(s):
    s = s.replace('...', ' ')                             # Removing ellipses
    tokens = nltk.word_tokenize(s)                        # tokenizing text
    tagged = nltk.pos_tag(tokens)                         # tagging with part of speech
    filtering = ['JJ', 'JJR', 'JJS']                     # choosing adjectives
    words = [k for k, v in tagged if v in filtering]
    filtered = ' '.join(words)
    return filtered

adjectives_data = textdata.text.apply(filter_for_tfidf)
```

Countvectorizer

‘good', 561
'great', 352
'nice', 160
'best', 157
'delicious', 137
'little', 130
'fresh', 95
'hot', 89
'ive', 85
'sure', 81
'small', 74
'better', 71
'friendly', 70
'big', 67
'favorite', 67
'happy', 62
'bad', 61
'excellent', 59
'decent', 58
'new', 57
'huge', 57
'cheese', 56
'different', 56
'large', 54
'super', 53

Countvectorizer

‘good', 561
'great', 352
'nice', 160
'best', 157
'delicious', 137
'little', 130
'fresh', 95
'hot', 89
'ive', 85
'sure', 81
'small', 74
'better', 71
'friendly', 70
'big', 67
'favorite', 67
'happy', 62
'bad', 61
'excellent', 59
'decent', 58
'new', 57
'huge', 57
'cheese', 56
'different', 56
'large', 54
'super', 53

Next steps

- Improve part of speech tagging
- Improve model by doing Tf-idf along with sentiment scores