SenseHq Data Challenge

Analyze public data about restaurants and create predictions about prices and preferences.

Shashank Shekhar Singh

Problem statement -

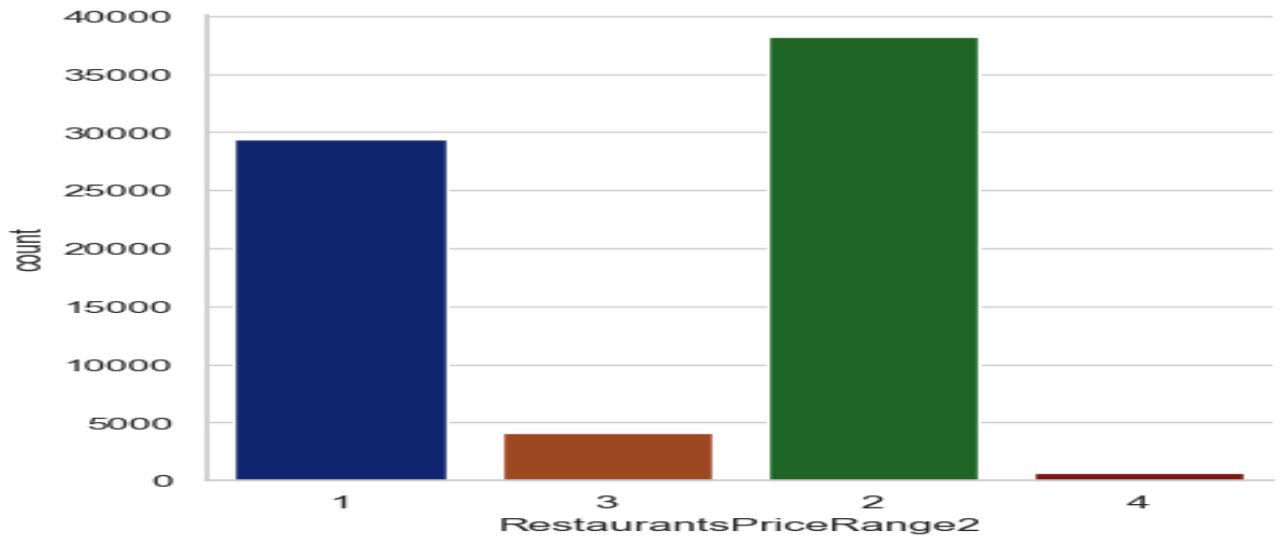
- Predict price ranges of restaurants solely based on reviews.
- Recommending restaurants based on user preferences and a city name.

Data Set -

- This problem relies on Yelp restaurant dataset available on Kaggle.
- The data contains information about business, checkins, reviews, tips, and users.
- The dataset(compressed) is approx. 4GB in size.
- Used business & reviews data for solving both the problems

EDA Summary

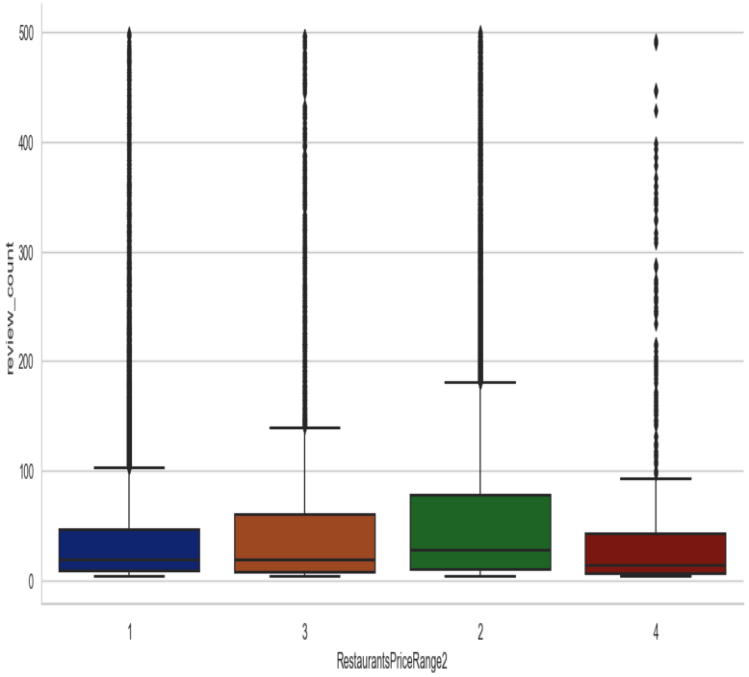
- Out of 210K business there are 85K restaurants and 72K have price information.
- There are 4 price classes with skew towards lower price ranges – possibly due to less number of high end & premium restaurants
- Las Vegas, Toronto & Phoenix are top cities by restaurant reviews

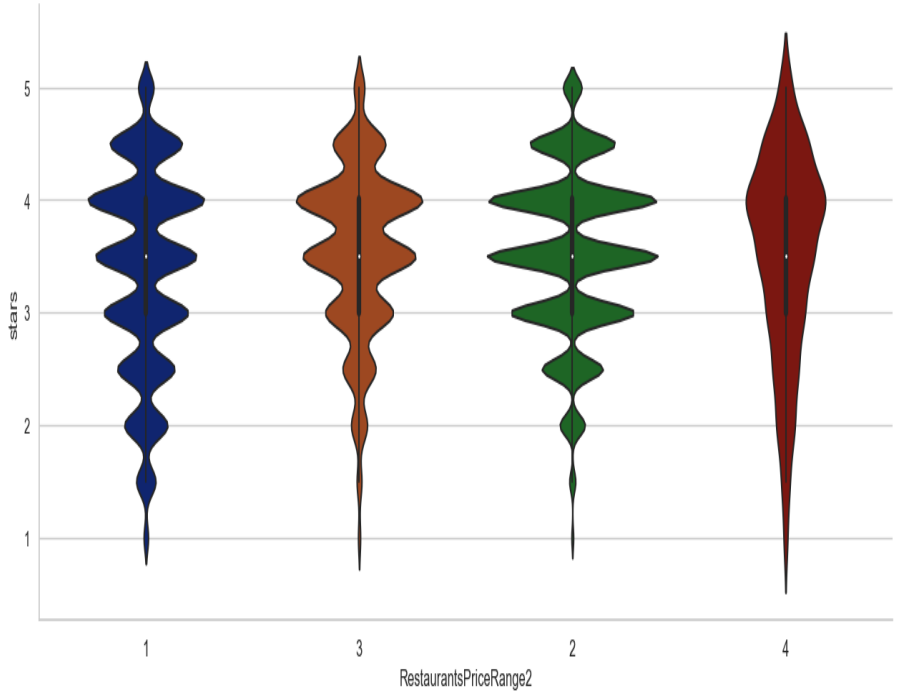


* For more details refer yelp_EDA.pdf

EDA Summary – cont.

- Avg ratings are distributed from 1 to 5 stars with the ratings for premium restaurants (4) skewed higher
- Lower priced restaurants have more wide review counts

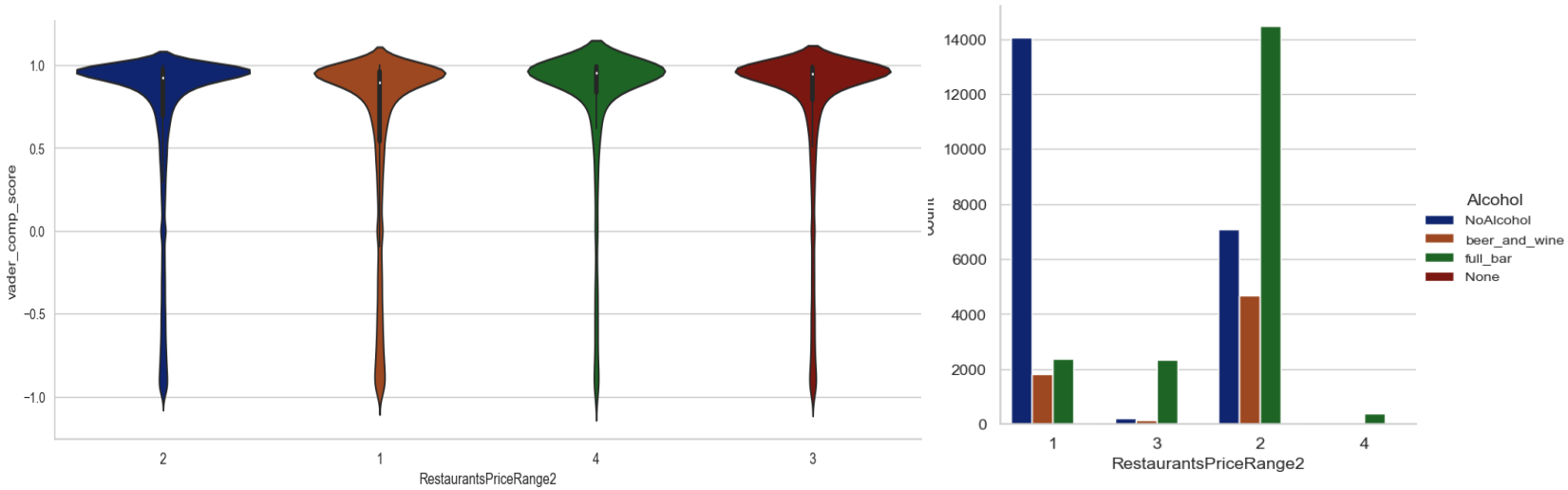




* For more details refer yelp_EDA.pdf

EDA Summary – cont.

- Most of the restaurants don't serve alcohol but High End (3) & Premium restaurants mostly are full bar.
- Mostly sentiments scores across all the price ranges are positive. For the lower priced restaurants (1 & 2), it seems they have more -ve sentiments as compared to high end & premium restaurants



- After filtering for relevant restaurants 5.5 million reviews remain.

* For more details refer yelp_EDA.pdf

Problem 1 – Predict Restaurant Price

- Data Pre Processing –
  - Review_Business_data_Sample.csv as the input.
  - This is a ~10% sample of total relevant reviews along with business features and sentiment scores.
  - Sentiment score features using VADER sentiment scoring.
  - VADER is chosen as it is trained and tuned to social media reviews and is better at handling emoticons.
  - Also used textblob sentiment scores.
  - Observed that some of the ratings conveyed a different message when compared to review text.
  - This could be because of inherent bias of user's previous experiences and hence a super score of stars was calculated as -

    super_score = stars + (vader_compound_score * textblob_polarity_score)
  - It range is from 0 to 6 against star's range from 1 to 5
  - ~20% sample of these 550K rows for creating a TF-IDF features of the raw text data. & subsequent train – test split
  - Cleansed raw text for removing punctuations & stop words.
  - Considered uni, bi & trigrams, limited to top 2000 features

* For more details refer Readme.md

Problem 1 – Predict Restaurant Price

- Model Training–
  - As seen in the EDA this is an imbalanced classification
  - Conducted 2 set of experiments –
    - price prediction model only with TF-IDF features
    - other where we'll mix TF-IDF with other features as well
  - Multinomial Naïve Bayes algorithm is chosen for baseline model training for both the experiments as it's training time is lesser and is reasonably accurate.
  - XgBoost classifier is chosen as next choice as –
    - It can handle missing values better
    - It can handle the inherent sparsity of TF-IDF better
    - Down-sampled class 2 as it has the highest number of observations.
  - F1 score is chosen as the scoring metric as we need to seek balance between precision & recall and the class distribution is skewed

* For more details refer Readme.md

Problem 1 – Predict Restaurant Price

- Model Training Results–

Model	Model Features	RestaurantPrice2_class	precision	recall	f1-score
NB	With only TF-IDF	1	0.74	0.15	0.25
		2	0.67	0.98	0.8
		3	0.5	0	0
		4	0	0	0
	With TF-IDF + other businessfeatures	1	0.62	0.67	0.64
		2	0.81	0.79	0.8
		3	0.44	0.47	0.45
		4	0.57	0.14	0.23
XGBoost	With only TF-IDF	1	0.52	0.73	0.6
		2	0.82	0.61	0.7
		3	0.32	0.5	0.39
		4	0.14	0.29	0.19
	With TF-IDF + other businessfeatures	1	0.61	0.81	0.69
		2	0.89	0.71	0.79
		3	0.44	0.67	0.53
		4	0.32	0.45	0.37

- Xgboost gives better results with both TD-IDF only features and TF-IDF + business features and recall is improving along with F1 scores for minority classes (3 & 4)

* For more details refer Readme.md

Problem 1 – Predict Restaurant Price

- Scope of improvement-
  - Use a better up-sampling strategy for minority classes
  - Can use pretrained GloVe for twitter embeddings instead of TF-IDF as-
    - it embeds the words to vectors based on the context of the words it appears.
    - Trained on twitter data and hence would have a better word embedding owing to the way people write on social platforms
    - But would require embedding the words related to food and drinks to the GloVe
  - Train own word2vec embedding
  - Try LSTM and GRU based deep learning architecture.

* For more details refer Readme.md

Problem 2 – Recommendation Engine

- Data Pre Processing –
  - Considered Las Vegas, Toronto & Phoenix for building recommendation engine owing to limited local hardware resources.
  - Get all the reviews from the city
  - Filter out the restaurants & users which have less than 20 reviews
  - Create a skinny file of only user_id, business_ids, stars(rating) and a composite rating

  Composite_rating = rating * (1+ vader_sentiment_score) +1

  Its range varies from 1 to 11

  - Create index mapping for user_id & business_id with either rating or composite rating as its value and
  - Join all the reviews for a restaurant / a user into one big sentence and compute TF-IDF matrix for each restaurant / user & compute cosine similarities– to be used for review based training
  - Create a user – restaurant mapping matrix with values as rating (or composite rating)

* For more details refer Readme.md

Problem 2 – Recommendation Engine

- Model Training –
  - Tried 2 approaches –
    - Text based recommendation -
    - Memory based Collaborative Filtering (both user based & item based)
  - Also considered both ratings star & composite ratings for CF models
  - Text based recommendation –
    - Extract User's preference vector from all the reviews written by a user (using TF-IDF vectorization)
    - Similarity extract restaurant's feature vector for all the reviews written for that restaurant (using TF-IDF Vectorization)
    - For example, if the word "price" is very frequent both in the reviews for a restaurant, and also in the reviews of an user.
    - The intuition behind this could be that everybody is talking about the price of this restaurant, and this user happens to care about the price a lot, therefore they can be somehow be matched up.
    - At this point, it cannot be told whether this user would like this restaurant (it could be that everybody is actually complaining about the price), but still, there is some relationship between this user and this restaurant.
    - At the regression step, use locally weighted linear regression

* For more details refer Readme.md

Problem 2 – Recommendation Engine

- Model Training –
  - Memory based Collaborative Filtering –
    - Each row of user restaurant matrix is the user characteristic vector & column is the restaurant characteristic vector
    - Compute pairwise cosine similarity between individual users & individual restaurants
    - **Create user based prediction** - compute the weighted average of the ratings from all the users to the restaurant, with the similarities between the particular user and other users as the weights.
    - In this method subtract the mean ratings of each user to eliminate the bias that some user tends to always give higher ratings (while some users might always give lower ratings).
    - **Create item based prediction -** compute the weighted average of the ratings of all the restaurants for the user, with the similarities between the particular restaurants and other restaurants as the weights.
  - Used RMSE as the model scoring metric
  - Computed Final predictions from CF methods as they performed better.

* For more details refer Readme.md

Problem 2 – Recommendation Engine

- Model Training Results–

City	Model (rating)	Traning RMSE	Testing RMSE
Las Vegas	Text Based Recommendation	1.6597	1.6959
	User Based CF	1.1962	1.1952
	Item Based CF	1.2168	1.217
Toronto	Text Based Recommendation	1.7526	1.7782
	User Based CF	1.0295	1.0329
	Item Based CF	1.0563	1.0605
Phoenix	Text Based Recommendation	1.532	1.5644
	User Based CF	1.1956	1.1884
	Item Based CF	1.1892	1.178

City	Model (composite rating)	Traning RMSE	Testing RMSE
Toronto	Text Based Recommendation	3.8584	3.8593
	User Based CF	2.7438	2.7556
	Item Based CF	2.4153	2.4312

- Stars Rating based collaborative filtering methods perform better as compared to composite score rating
- Item based collaborative filtering method performed better for all 3 cities

* For more details refer Readme.md

Problem 1 – Recommendation Engine

- Scope of improvement-
  - Can try matrix factorization approaches like Singular Value decomposition & Alternative Least Squares
  - Can create a much rich composite rating score using tree based methods and then use them in collaborative filtering approach.

* For more details refer Readme.md

Thank You