Yelp Dataset

20/Jul/2019

Daniel Pérez
Bruno Miranda
Jose Antonio Amieva
Pedro Rossi

Overview

Yelp provides free of charge a subset of their very extensive business-review dataset, consisting of over 6 million reviews across 10 different metropolitan areas in North America.

We decided to use this dataset, and other supporting datasets when appropriate, as our starting Project.

Each one of us formulated a research question and set to work on the dataset to try and answer it.

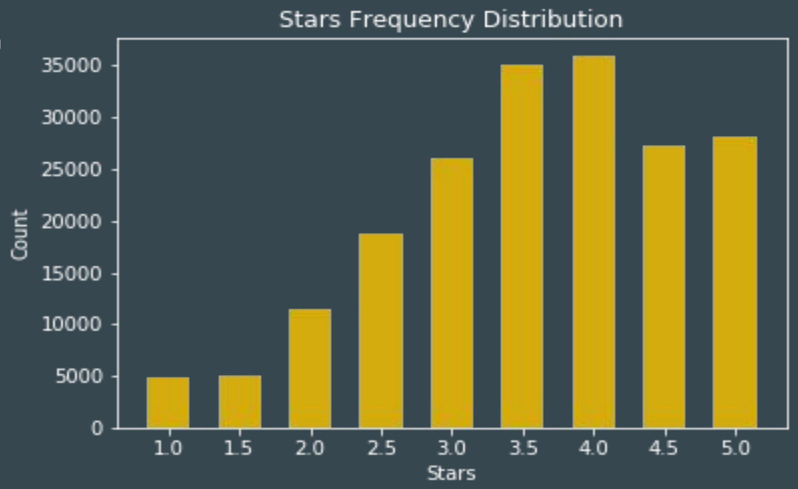
Data Cleaning and Preparing

Reading the User and Review files

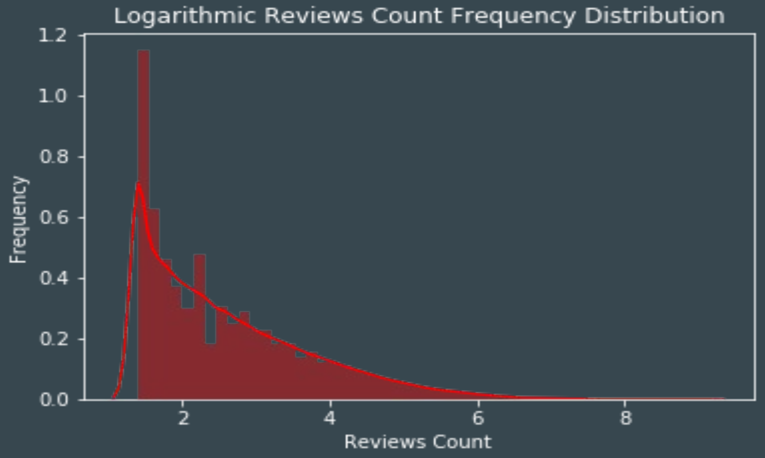
In [47]:
```
reviewIds = []
userIds = []
businessIds = []
stars = []
useful = []
funny = []
cool = []
date = []

start = timer()
with open('yelp_dataset/reviewClean.json', encoding="utf-8") as file:
    parser = ijson.parse(file)
    for prefix, event, value in parser:
        if prefix == "item.review_id":
            reviewIds.append(value)
        elif prefix == "item.user_id":
            userIds.append(value)
        elif prefix == "item.business_id":
            businessIds.append(value)
        elif prefix == "item.stars":
            stars.append(value)
        elif prefix == "item.useful":
            useful.append(value)
        elif prefix == "item.funny":
            funny.append(value)
        elif prefix == "item.cool":
            cool.append(value)
        elif prefix == "item.date":
            d = value
            date.append(datetime.datetime.strptime(value, "%Y-%m-%d %H:%M:%S"))

end = timer()
print(f"*** FINISHED READING DATASET IN {end - start} SECONDS ***")
```

KPI - Key Performance Indicator

- Imbalanced distribution of stars
  - Solution: Standardize the column



- Exponential density of reviews count frequency
  - Solution: Apply a factor of reliability for "bins" of reviews count based on



KPI - Key Performance Indicator

- Imbalanced distribution of stars
  - Solution: Standardize the column

```
# Standardize stars column
    # Create the Scaler object
scaler = preprocessing.StandardScaler()

    # Apply scaler to stars column
standard_stars = scaler.fit_transform\
                (np.array(data['stars']).reshape(-1, 1))

    # Insert standardized column to dataframe
data['Standardized Stars'] = standard_stars

# Insert KPI column to dataframe
data['KPI'] = data['Standardized Stars']*score
```

- Exponential density of reviews count frequency
  - Solution: Apply a factor of reliability for “bins” of reviews count based on

```
score = []

for review in data['review_count']:
    # Until mean receive score 0.6
    if review <= 34:
        score.append(0.6)
    # Until mean + std dev receive score 0.7
    elif review <= 144:
        score.append(0.7)
    # Until mean + 2*std dev receive score 0.8
    elif review <= 254:
        score.append(0.8)
    # Until mean + 3*std dev receive score 0.9
    elif review <= 364:
        score.append(0.9)
    # Above mean + 3*std dev receive score 1.0
    else:
        score.append(1.0)
```

Data Format

Business

	business_id	categories	city	hours	is_open	latitude	longitude	name	postal_code	review_count	stars	state
	1SWheh84yJXfytovILXOAQ	Golf, Active Life	Phoenix	None	0	33.522143	-112.018481	Arizona Biltmore Golf Club	85016	5	3.0	AZ

Review

	Review Id	User Id	Business Id	Stars	Useful	Funny	Cool	Date
0	Q1sbwvVQXV2734tPgoKj4Q	hG7b0MtEbXx5QzbzE6C_VA	ujmEBvifdJM6h6RLv4wQIg	1.0	6	1	0	2013-05-07 04:34:36
1	GJXCdrto3ASJOqKeVWPi6Q	yXQM5uF2jS6es16SJzNHfg	NZnhc2sEQy3RmzKTZnqtwQ	5.0	0	0	0	2017-01-14 21:30:33

User

	User Id	Name	Review Count	Yelping Since	Useful	Funny	Cool	Elite	Friend Count	Fans	Avg Stars
0	l6BmjZMeQD3rDxWUbiAiow	Rashmi	95	2013-10-08 23:11:33	84	17	25	2015,2016,2017	99	5	4.03
1	4XChL029mKr5hydo79Ljxg	Jenna	33	2013-02-21 22:29:06	48	22	16		1152	4	3.63

Checkin

	business_id	weekday	hour	checkins
0	3Mc-LxcqeguOXOVT_2ZtCg	Tue	0:00	12
1	SVFx6_epO22bZTZnKwlX7g	Wed	0:00	4

Research Questions

Count of Restaurants against Ratings

1- To accomplished this result, we load a huge csv file (5GB) with chuksize attribute.

2- Filter only restaurants category and merge the restaurants only dataset and reviews dataset by ‘Business ID’ so we can filter only the reviews on the restaurants

3- Label each review into 3 categories (positive, neutral and negative)

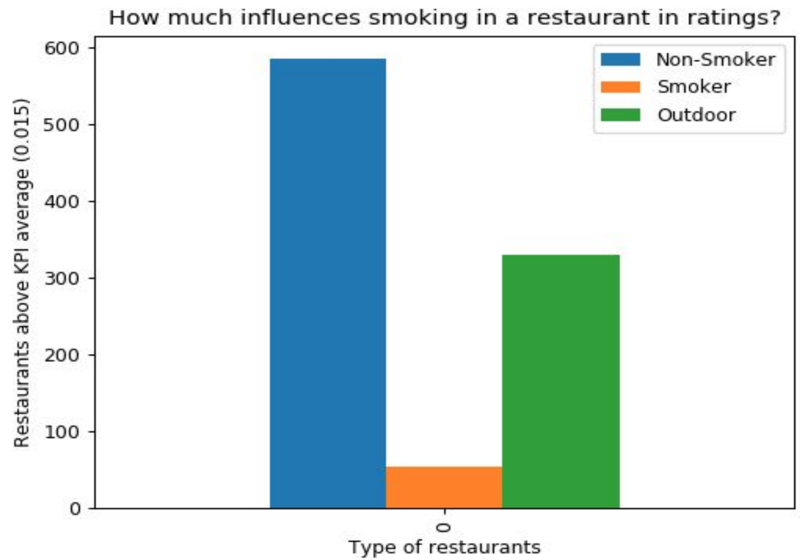
Count of Restaurants against Ratings
Count of restaurants
Average Rating
0
2000
4000
6000
8000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
261
853
2201
3740
6251
8415
8777
4398
982

How smoking affects Restaurant Reviews based on smoking attr in business csv

Smoking Value Counts:
Smoker, Non-Smoker, Outdoor

	Non-Smoker	Smoker	Outdoor
0	586	55	330

```
rest_smoking_above_avg.plot(kind='bar')
plt.title('How much influences smoking in a restaurant in ratings?')
plt.xlabel('Type of restaurants')
plt.ylabel('Restaurants above KPI average (0.015)')
```



Taking out the average of the KPI Restaurants Review (0.015) and counting each Smoking Value Counts we conclude that NON-Smoker restaurants got better results.

Deeper Analysis in reviews using words as indicators

Bad Words

In [31]:
```
restaurant_cleaned_reviews.text = restaurant_cleaned_reviews.text.str.lower()

good_words = ['great','amazing','love','best','awesome','excellent','good',
              'favorite','loved','perfect','gem','perfectly','wonderful',
              'happy','enjoyed','nice','well','super','like','better','decent',
              'pretty','enough','excited','impressed','ready','fantastic','glad
              'fabulous']
bad_words = ['bad','disappointed','unfortunately','disappointing','horrible',
             'lacking','terrible','sorry', 'disappoint']
```

In [32]:
```
#create empty colum and reset index
restaurant_cleaned_reviews['Review'] = ''
restaurant_cleaned_reviews.reset_index(drop=True).head()
```

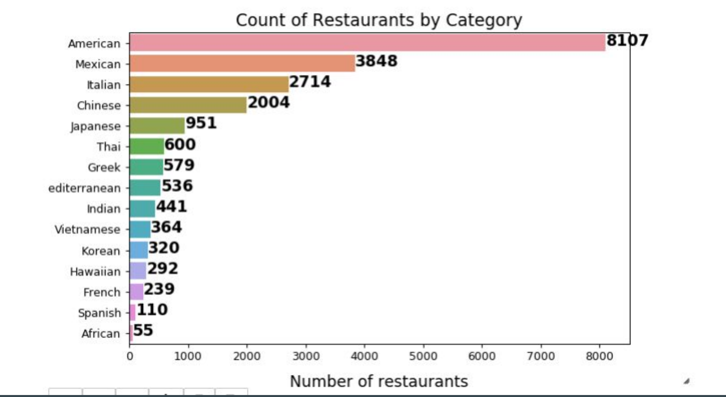
```
for good in good_words:
    if good in restaurant_cleaned_reviews['text']:
        restaurant_cleaned_reviews['Review'] = 'P'

for bad in bad_words:
    if bad in restaurant_cleaned_reviews['text']:
        restaurant_cleaned_reviews['Review'] = 'B'
```

	Review
P	2177589
	977629
B	270183

Notice how using this word indicator is very useful when it comes to get more insights

Reviews based on Cuisine Type only using business.csv

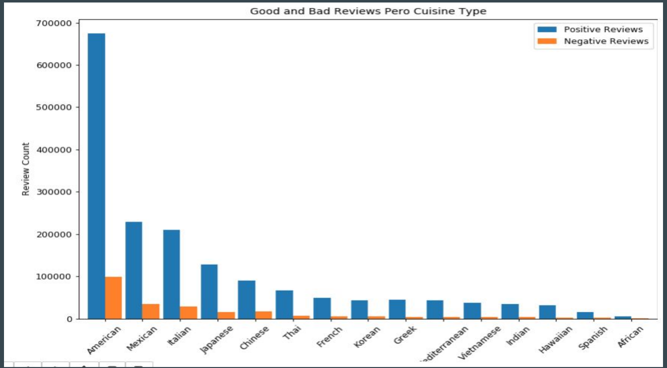
	category
American	8107
Mexican	3848
Italian	2714
Chinese	2004
Japanese	951
Thai	600
Greek	579
Mediterranean	536
Indian	441
Vietnamese	364
Korean	320
Hawaiian	292
French	239
Spanish	110
African	55

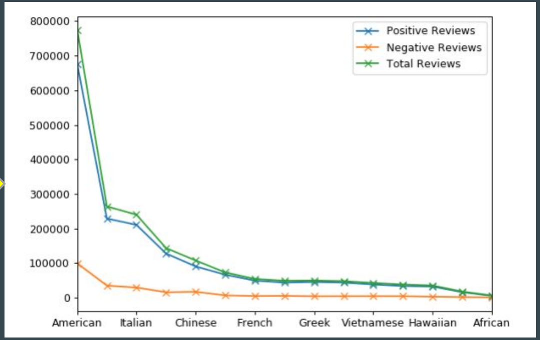


Reviews based on Cuisine Type with word Identificator

```python
plt.figure(figsize=(8,5))
cuisine_type_grouped = usa_restaurants.category.value_counts()
sns.countplot(y='category',data=usa_restaurants,
              order = cuisine_type_grouped.index)
plt.xlabel('Number of restaurants', fontsize=14, labelpad=10)
plt.ylabel('Category', fontsize=14)
plt.title('Count of Restaurants by Category', fontsize=15)
for  i, v in enumerate(usa_restaurants.category.value_counts()):
    plt.text(v, i+0.15, str(v), fontweight='bold', fontsize=14)
```

Cuisine	Positive Reviews	Negative Reviews	Total Reviews
American	675006	99461	774467
Mexican	228922	34989	263911
Italian	210546	29420	239966
Japanese	127702	15684	143386
Chinese	90417	17110	107527
Thai	66379	6607	72986
French	49405	4859	54264
Korean	43681	5279	48960
Greek	45533	4122	49655
Mediterranean	43707	4258	47965
Vietnamese	38141	4407	42548
Indian	33917	4283	38200
Hawaiian	32233	2942	35175
Spanish	15978	1789	17767
African	5382	964	6346





Do Yelp users rate Mexican-related businesses lower after Donald Trump became President?

Donald Trump has been famously discriminatory against latin people, and his supporters feel that they are now free to express in public their own discriminatory feelings, instead of just privately holding them.

This being the case, it makes sense that Yelp reviews reflect this now openly-racist point of view when it comes to Mexican-related businesses.

So we would expect that Mexican-related businesses to have a lower Yelp rating after Trump became President.

Filtering for Mexican businesses

```
mexicanRestaurants = businessDf.loc[businessDf.categories.str.contains(".exican") == True]
print(len(mexicanRestaurants))
mexicanRestaurants.head()
```

```
4628
```

	address	attributes	business_id	categories	city	hours	is_open	latitude	longitude	name	postal_
11	2450 E Indian School Rd	{'RestaurantsTakeOut': 'True', 'BusinessParkin...	1Dfx3zM-rW4n-31KeC8sJg	Restaurants, Breakfast & Brunch, Mexican, Taco...	Phoenix	{'Monday': '7:0-0:0', 'Tuesday': '7:0-0:0', 'W...	1	33.495194	-112.028588	Taco Bell	

Separate the data before and after January 20, 2017, the date when Trump took office

```
dateOfTrumpInauguration = datetime.datetime(2017, 1, 20)
reviewDf.Date = pd.to_datetime(reviewDf.Date)
reviewsBeforeInauguration = reviewDf.loc[reviewDf.Date < dateOfTrumpInauguration]
reviewsAfterInauguration = reviewDf.loc[reviewDf.Date > dateOfTrumpInauguration]
print(len(reviewsBeforeInauguration))
print(len(reviewsAfterInauguration))
```

```
4348163
2337737
```

Get the KPI from the BeforeTrump dataset

```
groupByBusinessBefore2016 = reviewsBeforeInauguration.groupby("Business Id")
avgStarsBefore2016 = groupByBusinessBefore2016[["Stars"]].mean()
avgStarsBefore2016.Stars = round(avgStarsBefore2016.Stars, 2)
avgStarsBefore2016["Review Count"] = groupByBusinessBefore2016[["Review Id"]].count()
avgStarsBefore2016 = avgStarsBefore2016.reset_index()

avgStarsBefore2016.describe()
```

	Stars	Review Count
count	172712.000000	172712.000000
mean	3.578155	25.175801
std	1.076889	87.489724
min	1.000000	1.000000
25%	3.000000	3.000000
50%	3.670000	7.000000
75%	4.440000	18.000000
max	5.000000	6661.000000

```
getKPIList(avgStarsBefore2016, avgStarsBefore2016["Review Count"], avgStarsBefore2016.Stars, 25, 87)
avgStarsBefore2016.columns = ["business_id", "Stars Pre-Trump", "Review Count Pre-Trump", "Std Stars Pre-Trump", "KPI Pre-Trump"]
avgStarsBefore2016.head()
```

	business_id	Stars Pre-Trump	Review Count Pre-Trump	Std Stars Pre-Trump	KPI Pre-Trump
0	--1UhMGODdWsrMastO9DZw	3.82	11	0.224579	0.134747
1	--6MefnULPED_I942VcFNA	3.15	26	-0.397586	-0.278310
2	--7zmmkVg-IMGaXbuVd0SQ	4.03	31	0.419585	0.293710
3	--8LPVSo5i0Oo61X01sV9A	4.00	2	0.391727	0.235036
4	--9QQLMTbFzLJ_oT-ON3Xw	3.44	9	-0.128291	-0.076974

Get the KPI from the AfterTrump dataset

```
groupByBusinessAfter2016 = reviewsAfterInauguration.groupby("Business Id")
avgStarsAfter2016 = groupByBusinessAfter2016[["Stars"]].mean()
avgStarsAfter2016.Stars = round(avgStarsAfter2016.Stars)
avgStarsAfter2016["Count"] = groupByBusinessAfter2016[["Review Id"]].count()
avgStarsAfter2016 = avgStarsAfter2016.reset_index()

avgStarsAfter2016.describe()
```

	Stars	Count
count	150282.000000	150282.000000
mean	3.562636	15.555669
std	1.269986	43.468826
min	1.000000	1.000000
25%	3.000000	2.000000
50%	4.000000	5.000000
75%	5.000000	12.000000
max	5.000000	2749.000000

```
getKPIList(avgStarsAfter2016, avgStarsAfter2016["Count"], avgStarsAfter2016.Stars, 15, 43)
avgStarsAfter2016.columns = ["business_id", "Stars Post-Trump", "Review Count Post-Trump", "Std Stars Post-Trump", "KPI Post-Trur
avgStarsAfter2016.head()
```

	business_id	Stars Post-Trump	Review Count Post-Trump	Std Stars Post-Trump	KPI Post-Trump
0	--1UhMGODdWsrMastO9DZw	4.0	15	0.344386	0.206632
1	--6MefnULPED_I942VcFNA	3.0	20	-0.443026	-0.310119
2	--7zmmkVg-IMGaXbuVd0SQ	4.0	28	0.344386	0.241070
3	--8LPVSo5i0Oo61X01sV9A	3.0	2	-0.443026	-0.265816
4	--9QQLMTbFzLJ_oT-ON3Xw	3.0	4	-0.443026	-0.265816

Merge both datasets

```
mexicanRestaurantsWithStars = pd.merge(mexicanRestaurants, avgStarsBefore2016, on="business_id", how="left")
mexicanRestaurantsWithStars = pd.merge(mexicanRestaurantsWithStars, avgStarsAfter2016, on="business_id", how="left")
mexicanRestaurantsWithStars.head()
```

hours	is_open	latitude	longitude	name	...	Standardized Stars	KPI	Stars Pre-Trump	Review Count Pre-Trump	Std Stars Pre-Trump	KPI Pre-Trump	Stars Post-Trump	Review Count Post-Trump	Std Stars Post-Trump	KPI Post-Trump
{'Monday': '7:0-0:0', 'Tuesday': '7:0-0:0', 'W...	1	33.495194	-112.028588	Taco Bell	...	-0.575015	-0.345009	3.00	13.0	-0.536876	-0.322126	3.0	6.0	-0.443026	-0.265816
{'Monday': '11:0-21:0', 'Tuesday': '10:0-21:0'...	1	36.195615	-115.040529	Maria's Mexican Restaurant & Bakery	...	0.897804	0.718243	4.35	128.0	0.716738	0.573391	4.0	61.0	0.344386	0.275509

Conclusions

```python
# Did the ratings for mexican businesses go down after Trump took office?
meanKpiPreTrump = round(mexicanRestaurantsWithStars["KPI Pre-Trump"].mean(), 2)
meanKpiPostTrump = round(mexicanRestaurantsWithStars["KPI Post-Trump"].mean(), 2)

print(f"KPI Pre-Trump = {meanKpiPreTrump}")
print(f"KPI Post-Trump = {meanKpiPostTrump}")
print()

if meanKpiPreTrump > meanKpiPostTrump:
    print(f"The KPI for Mexican-related businesses was higher before Trump became President")
else:
    print(f"The KPI for Mexican-related businesses was higher after Trump became President")
```

```
KPI Pre-Trump = -0.05
KPI Post-Trump = -0.11

The KPI for Mexican-related businesses was higher before Trump became President
```

Can Yelp features help to predict the closure of a business?

```
# Import libraries and dependencies
import pandas as pd
import matplotlib.pyplot as plt
import json
import scipy
import numpy as np
from sklearn import preprocessing
import math
import numpy as np
import seaborn as sns
%matplotlib inline
import xgboost as xgb
# from sklearn.model_selection import
StratifiedKFold
from sklearn.model_selection import
train_test_split
from sklearn.metrics import

confusion_matrix,accuracy_score
```

Can Yelp features help to predict the closure of a business?

- Stars:
  - Median (review.json);
  - Mean (review.json);
  - Stars (business.json)
- Reviews:
  - Count (business.json);
  - Count (review.json).
- Check-ins:
  - Sum (checkin.json).

```
# Selecting numerical columns for independent variables of the model
X_columns = ['stars',
             'review_count',
             'Mean',
             'Median',
             'NumberOfReviews',
             'TotalCheckins']
# Slicing dataframe into dependent variable and independent variables
X = data[X_columns]
y = data['is_open']

# Fill missing data with zeros
X = X.fillna(0.0)

X.head()
```

	stars	review_count	Mean	Median	NumberOfReviews	TotalCheckins
0	3.0	5	3.000000	4.0	5.0	20.0
1	2.5	128	2.669725	3.0	109.0	423.0
2	4.0	170	4.081081	4.5	148.0	663.0
3	5.0	3	0.000000	0.0	0.0	0.0
4	4.0	4	0.000000	0.0	0.0	0.0

Can Yelp features help to predict the closure of a business?

```
# Splitting dataset into training set and test set
train_X, test_X, train_y, test_y = train_test_split(X,
                                                    y,
                                                    test_size = 0.3,
                                                    random_state = 42)

# Define a XGBoost model for predict closure of a business
# fit model no training data
model = xgb.XGBClassifier()
model.fit(train_X, train_y)

# make predictions for test data
y_pred = model.predict(test_X)
predictions = [round(value) for value in y_pred]

# evaluate predictions
accuracy = accuracy_score(test_y, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

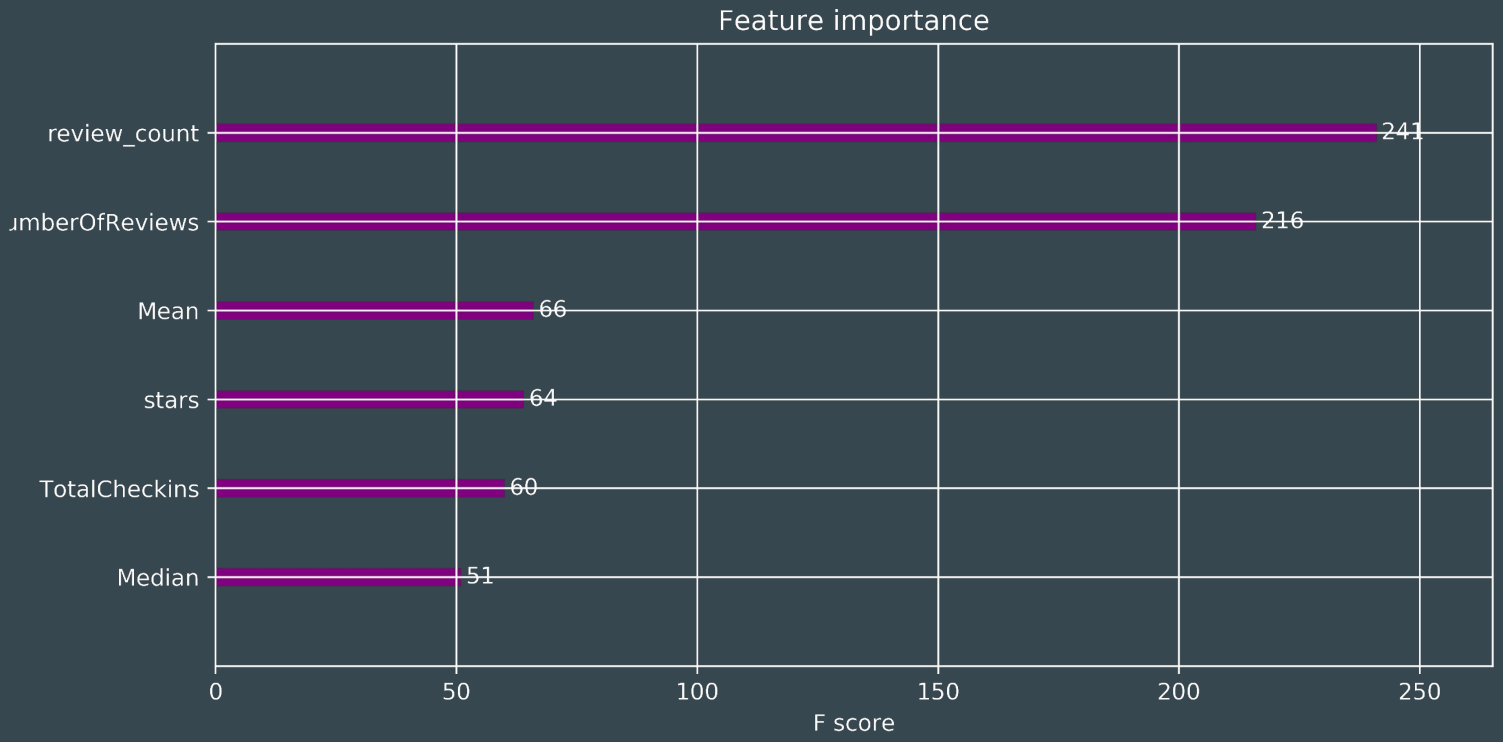
Can Yelp features help to predict the closure of a business?

But why Extreme Gradient Boost or XGBoost?

- ❖ Robust;
- ❖ Easy to implement;
- ❖ Fast;
- ❖ Most cases don’t require extensive parameter tuning.

Can Yelp features help to predict the closure of a business?

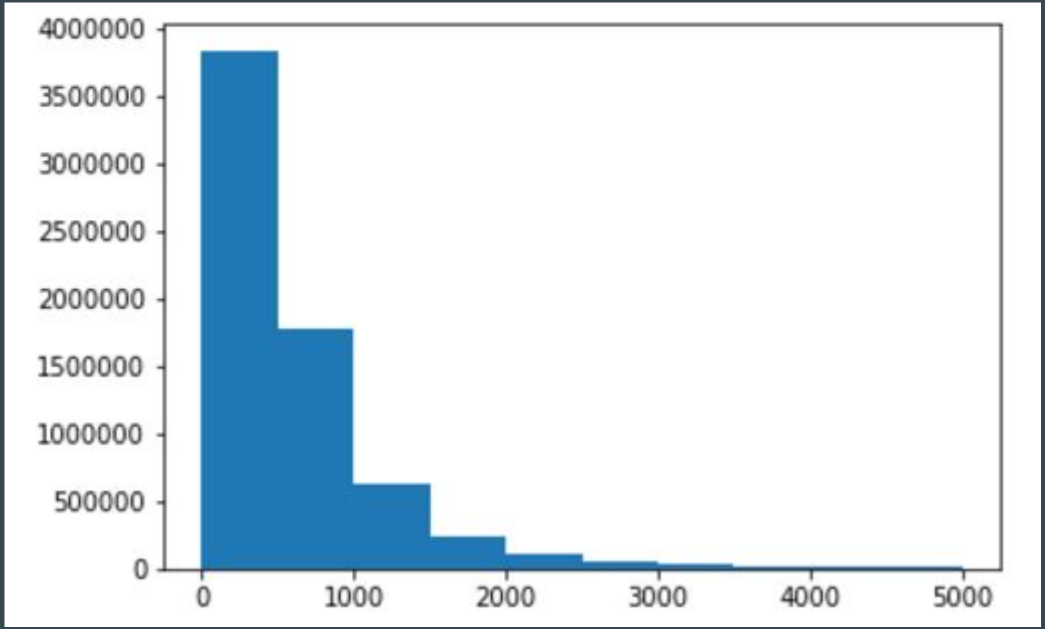
Results: Accuracy of 82.32% on the test set (30% of the whole data set)



Do people rant when angry in yelp?

Let's study the correlation of stars given to a business in a comment and length of the review written.

Distribution of review lengths:



The pearson correlation for two samples:

```
reviewSample = reviewDF[reviewDF['Review Length'] <= 1500]
reviewSample2 = reviewDF[reviewDF['Review Length'] > 1500]

pcorr = reviewDF['Stars'].corr(reviewDF['Review Length'])
pcorrRevSample = reviewSample['Stars'].corr(reviewSample['Review Length'])
pcorrRevSample2 = reviewSample2['Stars'].corr(reviewSample2['Review Length'])

print(pcorr)
print(pcorrRevSample)
print(pcorrRevSample2)
```

```
-0.1944699579426243
-0.17681591022074317
-0.09763892408259565
```

Conclusion

The general pearson correlation for the whole dataset and for the two samples has a very low negative coefficient.

This shows that there is a very weak (and not statistically significant) negative correlation.

A negative correlation means that, as suspected, the lower the stars ranking given in the comment, the longer the review written.

We can statistically conclude that there is no signficant correlation, so, yelpers do not generally rant.

Too long, did not read.

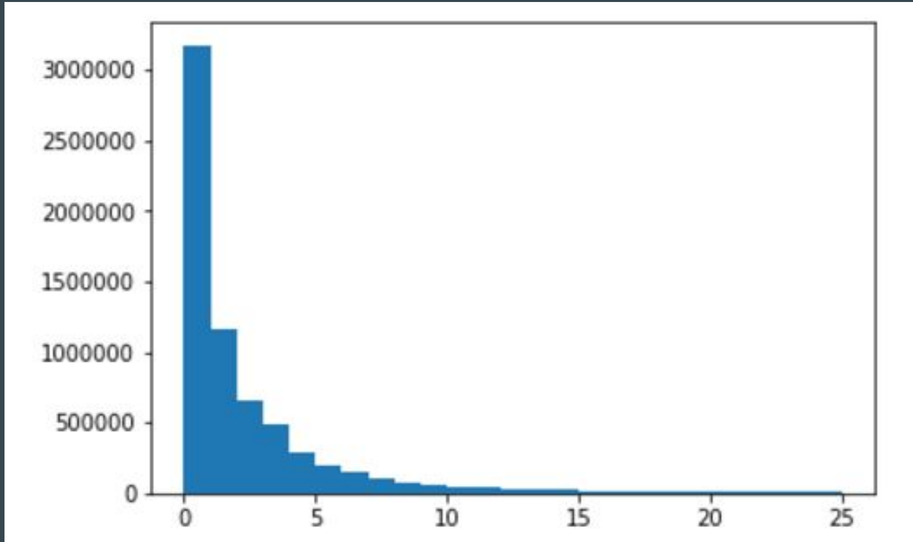
Now let's check if long comments mean less interactions with them.

Yelp has 3 tags for user interaction: Useful, Funny and Cool.

The sum of those 3 will be

counted as interaction.

Interaction distribution:



Significant data sampling:

We will eliminate the population that does not interact.

```
intSample = reviewDF[reviewDF['Interactions'] != 0]

print(len(reviewDF))
print(len(intSample))
```

```
6685900
3513797
```

Then get the pearson correlation:

```
intSample1 = intSample[intSample['Review Length'] <= 1500]
intSample2 = intSample[intSample['Review Length'] > 1500]

int1Pcorr = intSample1['Interactions'].corr(intSample1['Review Length'])
int2Pcorr = intSample2['Interactions'].corr(intSample2['Review Length'])

print(int1Pcorr)
print(int2Pcorr)
```

```
0.16081161339618688
0.08709163506220735
```

Conclusions

We see a very weak correlation between the interaction and review length for both samples.

We also see that the correlation is twice as weak for samples with more than 1,500 letters.

We can infer a certain degree of renuence to interact with longer comments from the magnitude of the studied correlations.

We cannot have a statistical confirmation of the hipothesis, then the conclusion is that, no matter the length of the review, yelpers will normally interact with them.

References

- Yelp API dataset .json format:
  - https://www.yelp.com/dataset.
- Format code snippets for presentations:
  - https://romannurik.github.io/SlidesCodeHighlighter/
- Github repository for this project:
  - https://github.com/Joseamica/Project-1
- XGBoost documentation:
  - https://www.google.com/search?q=xgboost+documentation&oq=XGBoost+docum&aqs=chrome.1.69i57j0l5.6757j0j4&sourceid=chrome&ie=UTF-8
- SKLearn documentation:
  - https://scikit-learn.org/stable/documentation.html
- Reviews.csv
  - https://wsi.li/BFYPYVEHJW3csW