

Group 10

PROJECT REPORT

**MARKET RESEARCH**

**ANALYSIS USING**

**MACHINE LEARNING**

Brian Nguyen

Mohana Surya Chaitanya Narella

Sai Nikhil Vaddavalli



TABLE OF CONTENTS

PROJECT DESCRIPTION 3

DATASET DESCRIPTION 4

METHODOLOGY 5

**Data Collection** **5**

**Data Preprocessing** **5**

**text mining** **6**

**exploratory data analysis** **7**

**machine learning** **9**

results and implications 10

**exploratory data analysis** **10**

**For Businesses** **10**

**For Users** **11**

**machine learning** **11**

**recommender app** **12**

codes 13

**PROJECT DESCRIPTION**

• Online reviews play an important role in helping businesses, especially restaurants, to track their performance, and pinpoint weaknesses to operate optimally.

• In this project, we use Yelp’s dataset on businesses, reviews, and tips data.

• Yelp is an American public company headquartered in San Francisco, California. The company develops, hosts, and markets the Yelp.com website and the Yelp mobile app, which publish crowd-sourced reviews about businesses. It also operates an online reservation service called Yelp Reservations.

• The main goal of our project is to create a Recommender Application that recommends appropriate restaurants to the users based on their preferences.

• Recommendations have a huge impact on business value and reduce the gap between the customer interface and the application.

• In the process of achieving this goal we have performed exploratory data analysis on the reviews using World Cloud which gave us an insight about the buzzing words which are most commonly used by the people.

• This analysis allowed us to improve the performance of the businesses by identifying the reasons which proved to be the important factors of the business.

• In addition to reviews, cuisines are also an important aspect to run a business. The Cuisines which are mostly preferred by the people are found, so including them would improve the businesses.

• Visualization of tips and stars were performed over months and years, and customer behavior is found over that period.

• Inorder to achieve this goal we have used Machine Learning models and Natural Language Processing on Yelp’s dataset.

• Using Machine Learning models the classification of the reviews through the rating is attained. Support Vector Machine, Decision Tree, Random Forest, and Multilayer Perceptron classifier models were used for the classification.

• Thus, reviews were classified as positive and negative reviews and model performances were evaluated.

• With the use of Categories, Reviews, Stars, and Location a recommender application is created using some features of the data.

• At the end some key parameters which could improve the business value of the organization were found.

**DATASET DESCRIPTION**

* The Yelp dataset we would be working on is primarily sourced from Kaggle.
* The Dataset contains data of 5,200,000 user reviews, 174,000 businesses, and 11 metropolitan locations in 4 countries related to various business categories, and their rating.
* It consists of separate tables pertaining to each of the following categories: Businesses, Reviews, Tips, Check-in, and Users.
* The Review table has 8021124 rows and 10 columns and as there are only 20 null values in the table they were dropped for further analysis. Data cleaning has been done so that exploratory data analysis on the reviews can be performed. This table is also used to create visualizations of stars by years and months.
* The Business table has 209393 rows and 14 columns and as the null values in the table, they were dropped for further analysis. Data cleaning has been done so that exploratory data analysis on the businesses, cuisines, and reviews can be performed.
* The Tips table has 1320761 rows and 5 columns. This table would be used to create visualizations of tips by years and months.
* The Check-in table has 175187 rows and 2 columns.
* The Users table has 1968703 rows and 23 columns
* Each of these data files was saved in JSON format and based on the requirement each of these individual files has been merged to conduct further Data Analysis.

**METHODOLOGY**

Diagram

Description automatically generated

Machine Learning Projects comprise of the above-mentioned processes. Each of these plays an important role in defining a solution to the problem. Following the above-mentioned procedure step by step gave us a lot of insights over the data and provided insights and corrective measures to improve the value of the businesses.

**Data Collection:**

* The data has been sourced from Kaggle.
* The data is a subset of Yelp businesses, reviews, and user data.
* Originally the data is in the form of separate tables.
* Each of the tables consists of data relating to check-in, Tip, reviews, business.
* The data stored in each of these tables are in the form of JSON documents.
* The data is further compressed to load the dataset onto the Google Cloud Platform.

**Data Preprocessing:**

* Missing data can always misguide the analysis.
* Data issues lead to a lot of discrepancies in Machine Learning Model analysis.
* Correlation between the variables also impacts the Model.
* Correlation between the variables has been checked, to ensure a proper Model Analysis.

Chart, treemap chart

Description automatically generated

**Text Mining:**

* [Text mining](https://monkeylearn.com/what-is-text-mining/) is an automatic process that uses natural language processing to extract valuable insights from unstructured text.
* Packages used are NLTK.
* Converting the text to the Lower case.
* Remove punctuations error from the text.
* Removing Stop Words which are unnecessary.
* Tokenization of the words.
* Stemming of the words reduces into root words.
* Count Vectorization is applied to the text for the transformation of the text.

Chart

Description automatically generated

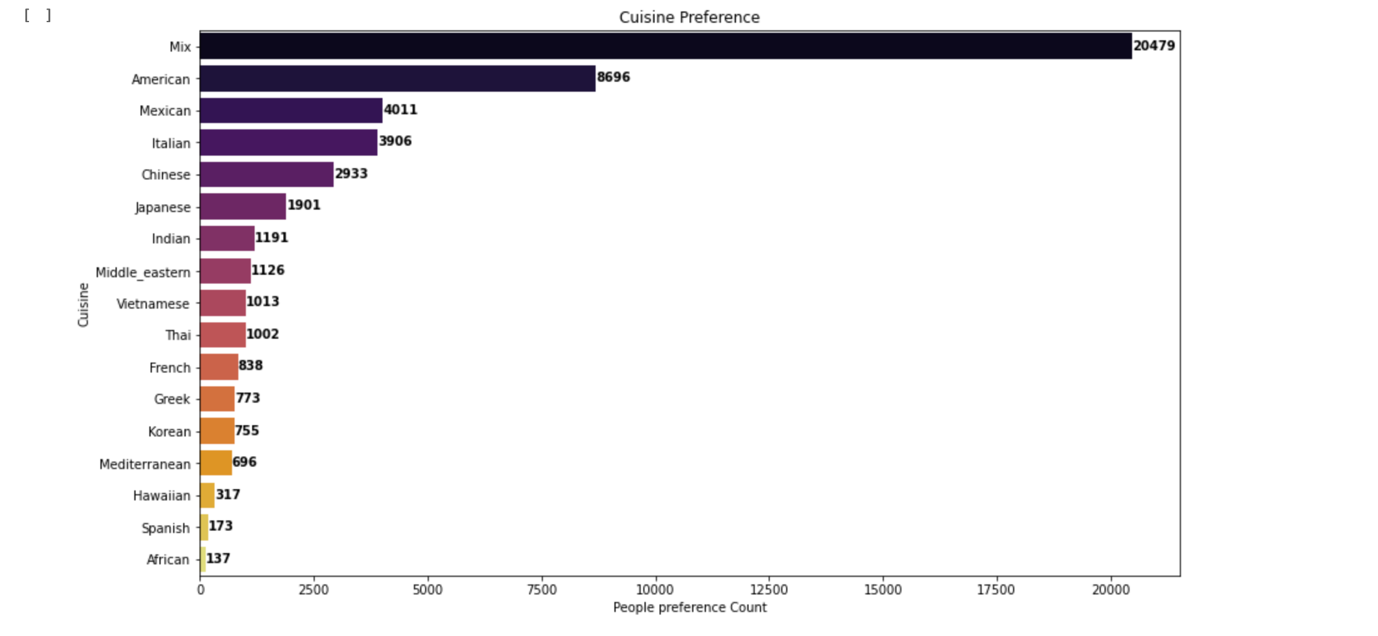
**Exploratory Data Analysis:**

* Packages Used: Seaborn, Matplotlib
* The visualizations provided great insight into a lot of data.
* These visualizations are usually Bar charts, Histograms, etc.
* To get an in-depth analysis for each of the Star ratings Word clouds have developed.
* An overall word cloud for the entire text is shown below.

Text

Description automatically generated

* Restaurants which have the following gallery, haircut, furniture, Museum, serene locations etc are some of the preferred choices by the people.
* Some of the views which are emphasized by the people with concerning the restaurants are Lunch, fine dining, specials, price, service and meeting are some of the important changes which could be brought in.



* The most prominent types of Cuisines are American, Mexican, Italian, Chinese, Japanese, Indian.
* Majority of the people prefer a mixed category of the cuisines.

Chart, bar chart

Description automatically generated

* The majority of the reviews are given from Toronto followed by Las Vegas and so on.

Chart, bar chart, histogram

Description automatically generated

* On average majority of the star ratings are given from the range of 3.0 to 4.0
* Most of the people rate the restaurants with 4.0 assuming it to be a good experience.

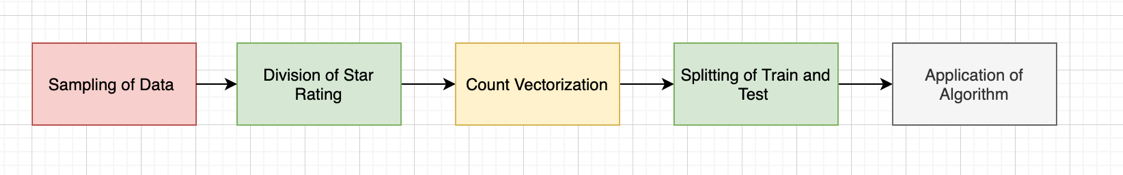
Chart

Description automatically generated

* McDonald’s followed by Subway seemed to be the most top-rated restaurant.
* Apart from both the above-mentioned restaurants, there are Taco Bell and Pizza Hut and several others, etc. which happened to be some of the top restaurants which are preferred by the people.

**Machine Learning:**

* Packages used are Scikit-Learn
* Algorithms used in the process are SVM, Decision Tree Classifier, Random Forest Classifier.
* Confusion Matrix and Classification Report.



* Natural Language Processing’s involvement in the process is to clean the text for further processing.
* Stop Word Removal.
* Tokenization of the words
* Count Vectorization transformation
* Upon cleaning the text and moving onto further analysis the Machine Learning algorithms are applied to the clean text.

**RESULTS AND IMPLICATIONS**

In this section, we will assess the results yielded from the exploratory data analysis, machine learning, and recommender application.

**Exploratory Data Analysis**

**For businesses:**

From the exploratory analysis of text length, we can see the distribution of text length is highly left-skewed. The majority of reviews are written with less than 1000 characters. Note that we have not removed any special characters, stop words, and characters of languages other than English. From this result, we can imply that most users prefer writing shorter reviews. We suspect this is due to time constraints and consideration for ease of use for other users when reading a review.

After a series of text cleaning and text mining, we attempted to build word clouds of reviews based on their “stars” ratings. For review with a “stars” rating of 1, we noted the appearance of negative words associated with a store-front business such as “Closed”, “dismal”, and “horrified”. As the “stars” rating increases, the number of negative words decreases and we also see an increase of positive words such as “attractive”, “nice”, “happy”.

Based on the text data mined from the processes earlier, we also performed an extraction of cuisine information for restaurants although it was not explicitly provided as a variable in our dataset. The set of different cuisines includes American, Mexican, Italian, Japanese, Chinese, Thai, etc. Our barplot of the counts of each cuisine showed that a majority of restaurants provide food fusion that is not specific defined as a certain cuisine and lead first with 20,479 businesses. “American”, “Mexican”, “Italian”, and “Chinese” follows as second, third, fourth and fifth spot accordingly. We also performed other counting analysis and learned that Toronto, Las Vegas have significantly more reviews than other cities. This is expected as these 2 cities are world-famous for travelers and have a relatively large population. Note that our data set only includes a limited number of cities in the US and Canada.

For the question of which restaurant name brand has the most number of reviews: we see that fast-food chains like McDonald’s, Subway, Taco Bell take the top 30 spots. This is expected as fast-food chains have a large number of restaurants around the country. Yelp should compile an overall score of each restaurant chain to give customers an overlook of each brand.

From the distribution plot of stars ratings given for each business, we saw that the distribution is relatively right-skewed with a high frequency of businesses having 3.5 to 4 stars and a very small number of businesses getting a 1 and 5 stars overall rating.

**For Users:**

From the Heatmap of Users’ variables “stars”, “useful”, “funny”, “cool”, we see that all variables except “stars are highly correlated. The underlying reason for this phenomenon is because the variables “stars”, although given by users, is targeted for, and is used to rate businesses. Meanwhile variables “useful”, “funny”, and “cool” are used to rate the Review itself and is given by other Users. Note we see the correlation coefficient between “useful”, “funny”, and “cool” are high and ranges between 0.67 to 0.78.

In terms of users’ behavior in giving tips (“funny”, “cool”, “useful”) we see that there is a sudden slow down of usage in 2013. The usage trend is gradually decreasing. This coincides with the integration of this functionality within the Yelp app where Yelp has slowly minimized the importance of the tipping functionality on their mobile app since 2013.

In terms of users’ behavior in activeness based on months, we see that there is a slight uptick in users’ activity in July and August (see Number of tips by moths plot) and “Distribution of stars by Month”. We can assume that users use the Yelp app more frequently to rate businesses and other reviews in July and August. We suspect the underlying of this phenomenon is because these 2 months also coincide with the start of a school year.

**Machine Learning**

In this section, we will outline the results yielded in each different model as well as their performance using metrics such as precision, recall, f1-score, and confusion matrix. We chose to exclude ROC and AUC because our analysis is multiclass. Note that all models were trained with a sampled dataset of 1000 entries that were also cross-validated by splitting into training and testing sets.

In the Support Vector Machine, we were able to achieve an overall f1-score of 51 percent across all 5 stars ratings. Note that the model did not perform well with start ratings of 2 – 3 and was not able to detect any correctly. On the other hand, the model excelled at predicting 5-star reviews with accuracy up to 98% on the training set. Despite being unable to distinguish between star ratings from 1 to 5. The visualization of the Support Vector Machine model still provided a relatively good separation between each group with group 5 most distinguishable from others.

In the Decision tree and random forest, we were also only able to achieve a low accuracy score of 49% and 56% accordingly. Note that we used the best parameters from the Decision Tree model to apply in Random Forest. From our visualization of Trees in both models, we see that the branches are not pure and have many branches (this is not against the criteria we wanted to see).

Multiplayer perceptron classifier was the best-performing model that we had. In this model, we were able to achieve a relatively high accuracy score of 70% on star ratings 1 and 5 although the overall accuracy was still low at 56%. As a result, we were able to show 2 examples where the model has correctly predicted the ratings of 1- and 5-star reviews.

**Recommender App**

We took a different approach when building our recommendation bot and limit the type of businesses to be only restaurants. A slightly different and more detailed text-cleaning and processing were also used to build the recommender bot. Based on criteria including name, stars, categories, our recommender then provide a list of the top (n = 3) restaurants with similar characteristics. In the example provided business\_id “tLpkSwdtqqoXwU0JAGnApw” - Wendy’s, we were able to pick other restaurants with similar attributes in that they are all fast food restaurants with 3.5 or above stars.

**CODES**

import pandas as pd

import numpy as np

import json

import matplotlib.pyplot as plt

import seaborn as sns

from nltk.tokenize import word\_tokenize

import string

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from wordcloud import WordCloud, STOPWORDS

**Reviews Dataset:**

df1 = pd.read\_csv('/content/drive/My Drive/review.gz', compression='gzip')

df1.head()

df1.columns

df1.shape

df1.info()

df1.describe()

df1.corr()

**Missing Value Identification:**

df1.isnull().sum()

df1 = df1.dropna()

# ****Heat**** ****Map:****

plt.figure(figsize=(15,8))

sns.heatmap(df1.corr(),cmap='BuGn',annot=True)

plt.title("Heatmap")

# ****Text Length:****

df1['text'] = df1['text'].astype(str)

df1['length'] = df1['text'].apply(len)

df1.head()

sns.displot(df1, x="length",color="red")

plt.title("Distribution of Length of Text")

plt.figure(figsize=(25,20))

**Visualizations:**

# Distribution of Stars by Year

df1['date'] = pd.to\_datetime(df1['date'])

df1['Year'] = df1['date'].dt.year

df1['Month'] = df1['date'].dt.month

f,ax2 = plt.subplots(figsize = (14,5))

stars = df1.groupby('Year').count()['stars'].to\_frame()

sns.barplot(stars.index, stars['stars'],palette = 'icefire')

ax2.set\_title('Distribution of Stars over years')

ax2.set\_ylabel('Number of Stars')

# Distribution of Stars by Months

f,ax3 = plt.subplots(figsize = (12,5))

star = df1.groupby('Month').count()['stars'].to\_frame()

sns.barplot(star.index, star['stars'],palette = 'hot')

ax3.set\_title('Distribution of Stars by Months')

ax3.set\_ylabel('Number of Stars')

# ****Text**** ****Cleaning:****

from nltk.corpus import stopwords

stop = stopwords.words('english')

df1["text"]= df1["text"].apply(lambda words: ' '.join. (word.lower() for word in words.split() if word not in stop))

df1.head()

df1['unique\_words'] = df1['text'].apply(lambda x: len(set(str(x).split())))

df1.head()

reviewurl = df1.loc[df1["text"].str.contains("http",na=False)]

print(f"Number of reviews with url: {len(reviewurl)}")

def word\_cloud(data):

wordcloud = WordCloud(max\_font\_size=40, relative\_scaling=.5).generate(str(data))

plt.figure()

plt.imshow(wordcloud)

plt.axis("off")

plt.show()

word\_cloud(df1["text"].loc[df1.stars == 1, ])

word\_cloud(df1["text"].loc[df1.stars == 2, ])

word\_cloud(df1["text"].loc[df1.stars == 3, ])

word\_cloud(df1["text"].loc[df1.stars == 4, ])

# ****Exploratory Data Analysis****

**Word Cloud for Reviews:**

from os import path

from PIL import Image

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

text = df1['text'].values

wordcloud = WordCloud().generate(str(text))

plt.figure( figsize=(10,10 ))

plt.imshow(wordcloud)

plt.axis("off")

plt.title("The Word Cloud for the Reviews")

plt.show()

**Business Dataset:**

df\_business = pd.read\_json('/content/drive/My Drive/Data files/business.json',lines=True)

df\_business.head()

df\_business.shape

df\_business.isnull().sum()

df\_business = df\_business.dropna()

df\_business['Cuisine'] = df\_business['Cuisine'].replace(np.nan, 'Mix')

df\_business.head()

Cuisine\_list = df\_business['Cuisine'].value\_counts()

plt.figure(figsize=(14,8))

sns.countplot(y ='Cuisine', data = df\_business, palette = "inferno",order = Cuisine\_list.index)

for i, v in enumerate(df\_business.Cuisine.value\_counts()):

plt.text(v, i+0.1, str(v), fontweight='bold', fontsize=10)

plt.xlabel("People preference Count")

plt.title("Cuisine Preference")

plt.show()

city=df\_business['city'].value\_counts()

city=city.sort\_values(ascending=False)

city=city.iloc[0:20]

plt.figure(figsize=(16,4))

ax = sns.barplot(city.index, city.values,palette = "inferno",alpha=0.7,)

plt.title("Most number of reviews",fontsize = 15)

locs, labels = plt.xticks()

plt.setp(labels,rotation = 30)

plt.ylabel('Count of the Reviews', fontsize=15)

plt.xlabel('City', fontsize=15)

rects = ax.patches

labels = city.values

for rect, label in zip(rects, labels):

height = rect.get\_height()

ax.text(rect.get\_x() + rect.get\_width()/2, height + 5, label, ha='center', va='bottom')

plt.show()

stars=df\_business['stars'].value\_counts()

stars=stars.sort\_index()

plt.figure(figsize=(8,4))

ax= sns.barplot(stars.index, stars.values, alpha=0.8,palette='Greens')

plt.title("Star Rating Distribution",fontsize=15)

plt.ylabel('Number of businesses', fontsize=15)

plt.xlabel('Star Ratings ', fontsize=15)

rects = ax.patches

labels = stars.values

for rect, label in zip(rects, labels):

height = rect.get\_height()

ax.text(rect.get\_x() + rect.get\_width()/2, height + 5, label, ha='center', va='bottom')

plt.show()

plt.figure(figsize=(15,8))

cnt = df\_business['name'].value\_counts()[:20].to\_frame()

sns.barplot(cnt['name'], cnt.index, palette = "Set1",alpha = 0.8)

plt.title("Top Restaurant Businesses ",fontsize=12)

plt.ylabel('Name of the Stores', fontsize=12)

plt.xlabel('Variation ', fontsize=12)

df\_business.name.value\_counts().index[:20].tolist()

state\_count = pd.DataFrame(df\_business['state'].value\_counts()[:20])

plt.figure(figsize=(12,6))

g = sns.barplot(x=state\_count.index, y=state\_count['state'], palette = 'Set1')

plt.title('Review Count by States');

g.set\_xticklabels(g.get\_xticklabels(),rotation=90)

plt.show()

cloud = WordCloud(width=1440, height= 1080,max\_words= 1000).generate(' '.join(df\_business['categories'].astype(str)))

plt.figure(figsize=(12, 8))

plt.imshow(cloud)

plt.title("Word Cloud for Categories",fontsize=12)

plt.axis('off');

**Check-in Dataset:**

df\_checkin = pd.read\_json('/content/drive/My Drive/Data files/checkin.json',lines=True)

df\_checkin

df\_checkin.shape

**Tips Dataset:**

df\_tip = pd.read\_json('/content/drive/My Drive/Data files/tip.json',lines=True)

df\_tip

df\_tip.shape

# Distribution of Tips by Year

df\_tip['date'] = pd.to\_datetime(df\_tip['date'])

df\_tip['Year'] = df\_tip['date'].dt.year

df\_tip['Month'] = df\_tip['date'].dt.month

f,ax = plt.subplots(figsize = (10,5))

tips = df\_tip.groupby('Year').sum()['compliment\_count'].to\_frame()

sns.barplot(tips.index,tips['compliment\_count'],palette='rocket')

ax.set\_title('Distribution of tips over years')

ax.set\_ylabel('Number of Tips')

# Number of Tips by Months

f,ax1 = plt.subplots(figsize = (16,5))

tip = df\_tip.groupby('Month').sum()['compliment\_count'].to\_frame()

sns.barplot(tip.index,tip['compliment\_count'],palette='cubehelix')

ax1.set\_title('Number of tips by months')

ax1.set\_ylabel('Number of Tips')

# ****Data Models and Performance****

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, accuracy\_score, roc\_auc\_score, roc\_curve

from sklearn.model\_selection import GridSearchCV

#Too much data have to sample

sampled\_df1 = df1.sample(n = 1000, random\_state=23)

sampled\_df1.shape

sampled\_df1.head()

# Set the Target by the 5 Stars Rating

classes = sampled\_df1[(sampled\_df1['stars']==1) | (sampled\_df1['stars']==2)| (sampled\_df1['stars']==3) | (sampled\_df1['stars']==4) | (sampled\_df1['stars']==5)]

classes.head()

print(classes.shape)

# Seperate the dataset into X and Y for prediction

x = classes['text']

y = classes['stars']

print(x.head())

print(y.head())

# Process Text (Review) Data for Vectorization

def clean\_text(text):

cln = [word for word in text if word not in string.punctuation]

cln = ''.join(cln)

return [word.lower() for word in cln.split() if word.lower() not in stopwords.words('english')]

sampled\_df1["text"] = sampled\_df1["text"].apply(clean\_text)

sampled\_df1.count()

# Vectorization (converting words into a vector) then apply to all Reviews

vocab = CountVectorizer(analyzer=clean\_text).fit(x)

x = vocab.transform(x)

# Splitting the dataset X into training and testing set:

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2, random\_state=101)

### **Support Vector Machine:**

# Modelling:

# Train a SVM model by tunning both C and gamma, report the best parameters

from sklearn.svm import SVC

svc = SVC()

svc.fit(x\_train, y\_train.values.ravel())

svc\_pred = svc.predict(x\_test)

# Performance Report

print("Confusion Matrix for Support Vector Machine:")

print(confusion\_matrix(y\_test,svc\_pred))

print("Score:",round(accuracy\_score(y\_test,svc\_pred)\*100,2))

print("Classification Report:")

print(classification\_report(y\_test,svc\_pred))

# Visualize our SVM

from sklearn.datasets import make\_blobs

x, y = make\_blobs(n\_samples=100, centers=5,

random\_state=0, cluster\_std=1.2)

plt.scatter(x[:, 0], x[:, 1], c=y, s=50, cmap='Set2');

### **Decision Tree & Random Forest:**

# Decision Tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

#this how to tune parameters

opt\_tree = DecisionTreeClassifier(random\_state = 0)

param\_DT = {"max\_depth": range(1,10),

"min\_samples\_split": range(2,10,1),

"max\_leaf\_nodes": range(2,10)}

grid\_tree = GridSearchCV(opt\_tree,param\_DT,cv=5)

grid\_tree.fit(x\_train,y\_train)

tree\_pred = grid\_tree.predict(x\_test)

# Performance Report

print("Confusion Matrix for Decision Tree Model:")

print(confusion\_matrix(y\_test,tree\_pred))

print("Score:",round(accuracy\_score(y\_test,tree\_pred)\*100,2))

print("Classification Report:")

print(classification\_report(y\_test,tree\_pred))

# Visualize Tree

from sklearn import tree

tree.plot\_tree(grid\_tree.best\_estimator\_);

# Random Forest using parameter from Decision Tree

from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier(random\_state=0)

param\_grid = {

'n\_estimators': [5,10,100,200],

'max\_features': ['auto', 'sqrt', 'log2'],

'max\_depth' : [4,5,6],

'criterion' :['gini', 'entropy']}

rfc\_cv= GridSearchCV(estimator=rfc, param\_grid=param\_grid, cv= 5)

rfc\_cv.fit(x\_train, y\_train)

print('The Best parameters for the Random Forest', rfc\_cv.best\_params\_)

rnd\_tree = RandomForestClassifier(n\_estimators=5, max\_depth = 5,max\_leaf\_nodes = 2, min\_samples\_split = 2)

rnd\_tree.fit(x\_train,y\_train)

forest\_pred = rfc\_cv.predict(x\_test)

# Performance Report

print("Confusion Matrix for Random Forest Model:")

print(confusion\_matrix(y\_test,forest\_pred))

print("Score:",round(accuracy\_score(y\_test,forest\_pred)\*100,2))

print("Classification Report:")

print(classification\_report(y\_test,forest\_pred))

# Visualize a tree (estimator) in our Random Forest

from sklearn import tree

tree.plot\_tree(rnd\_tree.estimators\_[1]); # Tree no. 1

# MULTILAYER PERCEPTRON CLASSIFIER

from sklearn.neural\_network import MLPClassifier

mlp = MLPClassifier()

mlp.fit(x\_train,y\_train)

mlp\_pred = mlp.predict(x\_test)

# Performance Report

print("Confusion Matrix for Multilayer Perceptron Classifier:")

print(confusion\_matrix(y\_test,mlp\_pred))

print("Score:",round(accuracy\_score(y\_test,mlp\_pred)\*100,2))

print("Classification Report:")

print(classification\_report(y\_test,mlp\_pred))

### **Show off Model Performance via Example:**

# POSITIVE REVIEW

ex1 = df1['text'][106894]

print(ex1)

print("Actual Rating: ",sampled\_df1['stars'][106894])

ex1\_t = vocab.transform([ex1])

print("Predicted Rating:")

mlp.predict(ex1\_t)[0]

# NEGATIVE REVIEW

ex2 = df1['text'][3463892]

print(ex2)

print("Actual Rating: ",sampled\_df1['stars'][3463892])

ex2\_t = vocab.transform([ex2])

print("Predicted Rating:")

mlp.predict(ex2\_t)[0]

# ****Recommendation Application for Restaurant Only****

This recommender will use Categories, Reviews, Stars and Location to make decision

# Location data is in a different data table

df\_business = pd.read\_json('/content/drive/My Drive/Data files/business.json',lines=True)

df\_business.isnull().sum()

df\_business = df\_business.dropna()

df\_business.head()

df\_business.shape

# Choose only restaurants

df2 = df\_business[df\_business['categories'].str.contains('Restaurants')]

print('Final Shape: ',df2.shape)

# Join Reviews and Categories

df\_yelp\_review\_iter = pd.read\_csv('/content/drive/My Drive/review.gz', compression='gzip', chunksize = 100000)

df\_yelp\_review = pd.DataFrame()

i=0

for df in df\_yelp\_review\_iter:

df = df[df['business\_id'].isin(df2['business\_id'])]

df\_yelp\_review = pd.concat([df\_yelp\_review, df])

i=i+1

print(i)

if i==4: break

# Make sure we matched the same business from both tables

df2 = df2[df2['business\_id'].isin(df\_yelp\_review['business\_id'])]

df2.head

df\_yelp\_review['text']

# We will this function to process and clean our Review Data

import re

def clean\_text(text):

## Remove puncuation

text = text.translate(string.punctuation)

## Convert words to lower case and split them

text = text.lower().split()

## Remove stop words

stops = set(stopwords.words("english"))

text = [w for w in text if not w in stops and len(w) >= 3]

text = " ".join(text)

# Clean the text

text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)

text = re.sub(r"what's", "what is ", text)

text = re.sub(r"\'s", " ", text)

text = re.sub(r"\'ve", " have ", text)

text = re.sub(r"n't", " not ", text)

text = re.sub(r"i'm", "i am ", text)

text = re.sub(r"\'re", " are ", text)

text = re.sub(r"\'d", " would ", text)

text = re.sub(r"\'ll", " will ", text)

text = re.sub(r",", " ", text)

text = re.sub(r"\.", " ", text)

text = re.sub(r"!", " ! ", text)

text = re.sub(r"\/", " ", text)

text = re.sub(r"\^", " ^ ", text)

text = re.sub(r"\+", " + ", text)

text = re.sub(r"\-", " - ", text)

text = re.sub(r"\=", " = ", text)

text = re.sub(r"'", " ", text)

text = re.sub(r"(\d+)(k)", r"\g<1>000", text)

text = re.sub(r":", " : ", text)

text = re.sub(r" e g ", " eg ", text)

text = re.sub(r" b g ", " bg ", text)

text = re.sub(r" u s ", " american ", text)

text = re.sub(r"\0s", "0", text)

text = re.sub(r" 9 11 ", "911", text)

text = re.sub(r"e - mail", "email", text)

text = re.sub(r"j k", "jk", text)

text = re.sub(r"\s{2,}", " ", text)

return text

df\_yelp\_review['text'] = df\_yelp\_review['text'].apply(clean\_text)

from nltk.tokenize import WordPunctTokenizer

from nltk.corpus import stopwords

from nltk.stem import SnowballStemmer

#Vectorize reviews

vectorizer\_reviews = CountVectorizer(min\_df = .01,max\_df = .99, tokenizer = WordPunctTokenizer().tokenize)

vectorized\_reviews = vectorizer\_reviews.fit\_transform(df\_yelp\_review['text'])

#Vectorize categories

vectorizer\_categories = CountVectorizer(min\_df = 1, max\_df = 1., tokenizer = lambda x: x.split(', '))

vectorized\_categories = vectorizer\_categories.fit\_transform(df2['categories'])

#Categories Vectors Shape:

print(vectorized\_reviews.shape)

print(vectorized\_categories.shape)

def bot():

#Welcome statement and ask for input

print (" ")

print("Welcome to Similar Restaurant Recommender Bot!")

business\_choose = input("Find similar restaurants to (Please Enter Biz\_ID) \_\_\_\_\_")

a = df2[df2['business\_id'] == business\_choose]

from scipy import sparse

businessxreview = sparse.csr\_matrix(pd.get\_dummies(df\_yelp\_review['business\_id']).values)

new\_reviews = df\_yelp\_review.loc[df\_yelp\_review['business\_id'] == business\_choose, 'text']

new\_categories = df2.loc[df2['business\_id'] == business\_choose, 'categories']

from scipy.spatial.distance import cdist

# find most similar reviews

dists1 = cdist(vectorizer\_reviews.transform(new\_reviews).todense().mean(axis=0),

vectorized\_reviews.T.dot(businessxreview).T.todense(),

metric='correlation')

# find most similar categories

dists2 = cdist(vectorizer\_categories.transform(new\_categories).todense().mean(axis=0),

vectorized\_categories.todense(),

metric='correlation')

# combine the two vectors in one matrix

dists\_together = np.vstack([dists1.ravel(), dists2.ravel()]).T

dists = dists\_together.mean(axis=1)

# select the closest 10

closest = dists.argsort().ravel()[:10]

output = df2.loc[df2['business\_id'].isin(df2['business\_id'].iloc[closest]), ['business\_id', 'categories', 'name', 'stars', 'city', 'state']]

# Print Results:

print('')

print('\*\*\*\*\*\*','Top 3 Similar Restaurants to Business\_id', business\_choose,'\*\*\*\*\*\*')

print('')

print(' 1. ')

print(output.iloc[0][['name', 'stars', 'city', 'state']])

print('')

print(' 2. ')

print(output.iloc[1][['name', 'stars', 'city', 'state']])

print('')

print(' 3. ')

print(output.iloc[2][['name', 'stars', 'city', 'state']])

# Use this business\_id as an example: tLpkSwdtqqoXwU0JAGnApw

# Notice that all result we got are fast foods restaurant

# This means that our Recommnender has learned the common features among these places

bot()