Checkpoint on Your Yelp Project

March 1, 2020

0.0.1 Yelp Dataset Analysis

Authors - David Terpay - Abhishek Pingle

Project Description Our project will center on how we can accurately and consistently determine what attributes of business are valuable to customers for a particular location. With this problem and solution, existing and new businesses can ensure that they provide their customers with what they want and customers can leave good reviews for the services/products/etc provided by the business. Our motivation is to help businesses improve their services by making improvements that specifically target customer preferences instead of spending money in an untargeted fashion. By identifying the factors that determine better reviews, businesses can learn what their competitors are doing to better accommodate their customers and generate additional value by focusing on those areas.

[65]:
```
# Importing all necessary libraries for this project
import pandas as pd
import matplotlib.pyplot as plt
import re
from spellchecker import SpellChecker

# Importing necessary ML models
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score, accuracy_score, log_loss,␣
↪confusion_matrix
import numpy as np
```

[2]:
```
# Preprocessing data and filtering to grab only Phoenix, AZ data
small_reviews = pd.read_json('yelp_dataset/tip.json', lines=True)
businesses = pd.read_json('yelp_dataset/business.json', lines=True)

# Grabbing only business data only from Phoenix, AZ
arizona_businesses = businesses[(businesses['state'] == 'AZ') &␣
↪(businesses['city'] == 'Phoenix')]

# Grabbing only Phoenix, AZ business reviews
```

```
arizona_reviews = small_reviews[small_reviews['business_id'].
 ↪isin(arizona_businesses['business_id'])]
f"There are {len(arizona_businesses)} businesses and {len(arizona_reviews)}␣
 ↪reviews that we will be utilizing"
```

[2]:
```
'There are 18764 businesses and 138434 reviews that we will be utilizing'
```

[3]:
```
def check_validity(text, spelling_tolerance = .5, acceptable_min_length= 3):
    '''
    This function helps ensure that the reviews are in a valid formatting. It␣
 ↪will
    check the following:
    1. Alphanumeric Characters only
    2. There are at least acceptable_min_length words in the review
    3. Are there any misspellings or lemmatization
    4. Corrects misspellings if there are less than spelling_tolerance errors
    5. Converts the text to lowercase

    Args:
        text: text to be filtered and/or corrected
        spelling_tolerance: percentage of words that can be misspelled
        acceptable_min_length: minimum number of words that can be in a review

    returns:
        '' if not valid, else it returns lowercase corrected text
    '''

    # We want only alphanumeric reviews
    if not re.match("^[a-zA-Z0-9_ ]+$", text):
        return ''

    tokenized = text.split(" ")
    # We want to ensure we have a certain amount of words
    if len(tokenized) < acceptable_min_length:
        return ''

    spell = SpellChecker(distance=1)
    misspelled = spell.unknown(tokenized)

    # If more than spelling_tolerance % of reviews are misspelled, ignore them
    if (len(misspelled) / len(tokenized)) >= spelling_tolerance:
        return ''
    # else replace each misspelled word with the correct word
    else:
        for word in misspelled:
            text = text.replace(word, spell.correction(word))
        return text.lower()
```

```
# Applying function to clean the reviews
arizona_reviews['text'] = arizona_reviews['text'].apply(check_validity)
# Removing invalid reviews
arizona_reviews.drop(arizona_reviews[arizona_reviews['text'] == ''].index,␣
↪inplace=True)

def split(text):
    return text.split(" ")

# Adding tokenized column
arizona_reviews['tokenized'] = arizona_reviews['text'].apply(split)
# Adding length column
arizona_reviews['num_words'] = arizona_reviews['tokenized'].apply(len)

# Removing unreviewed businesses
arizona_businesses = arizona_businesses[arizona_businesses['business_id'].
↪isin(arizona_reviews['business_id'])]

# Scatter plot of stars vs review count
def num_atts(attributes):
    if attributes is None:
        return 0
    else:
        return len(attributes)
arizona_businesses['num_attributes'] = arizona_businesses['attributes'].
↪apply(num_atts)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/ipykernel_launcher.py:42: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/pandas/core/frame.py:4117: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/ipykernel_launcher.py:50: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/ipykernel_launcher.py:52: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[4]:
```
arizona_reviews.head()
```

[4]:

	user_id	business_id	text	date	compliment_count	tokenized	num_words
161	ucj0RT3ZsWHFXQssLq-wmQ	VcYpdJIoDgwBOdIJ_IvadA	van gogh vroom vroom in my belly	2012-05-25 20:37:20	0	[van, gogh, vroom, vroom, in, my, belly]	7
232	2gu5B_JbJlfZLhihSr-apQ	pGLI8cMlay44bxYYjb4yIQ	checking out the wedding sight	2011-10-11 17:13:32	0	[checking, out, the, wedding, sight]	5
373	ctXFXk9-m6PCMoI-Nz2_XQ	u4icCXldY7ALU2eV3wdPgw	can be difficult getting in and out of this lo…	2016-05-04 08:39:28	0	[can, be, difficult, getting, in, and, out, of…	13
413	8LHmg3kR_e1S-8WSxMyCdg	8yfNxQH1MOaPorSR_Lmj9g	avoid this place	2015-01-26 19:30:38	0	[avoid, this, place]	3
479	yis0tDX83YnFLzUbKZ_KFg	veUZ0PzegME87yvicpAWuw	fire roasted burrito is delicious	2012-10-06 16:21:04	0	[fire, roasted, burrito, is, delicious]	5

[5]:
```
arizona_businesses.head()
```

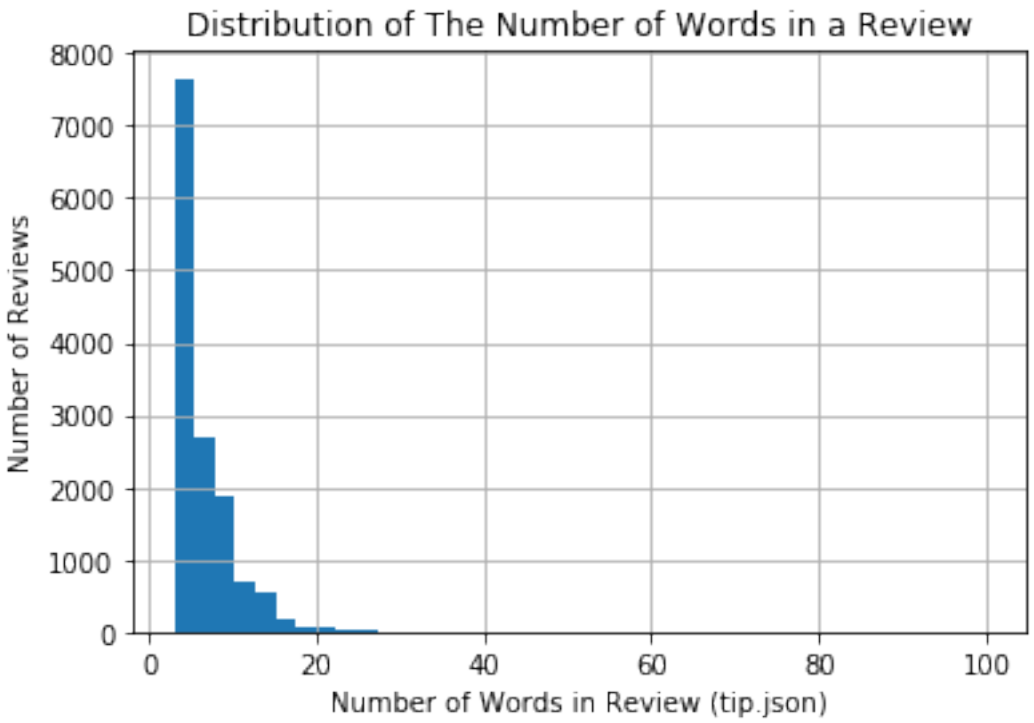
[5]:

	business_id	name
0	1SWheh84yJXfytovILXOAQ	Arizona Biltmore Golf Club
11	1Dfx3zM-rW4n-31KeC8sJg	Taco Bell
54	c-BELKj0SvNhBesQMf-bKw	Circle K
72	HYunM2pknhIh8lbiMa7THw	Dunn-Edwards Paints

```
78  44YFU284Z3KDEy25QyVoUw  Nee House Chinese Restaurant

                       address     city state postal_code   latitude  \
0   2818 E Camino Acequia Drive  Phoenix    AZ       85016  33.522143
11      2450 E Indian School Rd  Phoenix    AZ       85016  33.495194
54          3101 W Northern Ave  Phoenix    AZ       85051  33.552850
72           233 E Camelback Rd  Phoenix    AZ       85012  33.509011
78   13843 N Tatum Blvd, Ste 15  Phoenix    AZ       85032  33.613020

     longitude  stars  review_count  is_open  \
0  -112.018481    3.0             5        0
11 -112.028588    3.0            18        1
54 -112.125975    2.5             3        1
72 -112.069667    3.0            16        1
78 -111.977036    3.5           269        1

                                        attributes  \
0                          {'GoodForKids': 'False'}
11  {'RestaurantsTakeOut': 'True', 'BusinessParkin…
54            {'BusinessAcceptsCreditCards': 'True'}
72  {'BusinessAcceptsCreditCards': 'True', 'ByAppo…
78  {'Caters': 'True', 'GoodForKids': 'True', 'Noi…

                                        categories  \
0                                Golf, Active Life
11  Restaurants, Breakfast & Brunch, Mexican, Taco…
54  Convenience Stores, Automotive, Food, Gas Stat…
72  Interior Design, Contractors, Hardware Stores,…
78                           Chinese, Restaurants

                                            hours  num_attributes
0                                            None               1
11  {'Monday': '7:0-0:0', 'Tuesday': '7:0-0:0', 'W…              14
54  {'Monday': '0:0-0:0', 'Tuesday': '0:0-0:0', 'W…               1
72  {'Monday': '6:0-17:0', 'Tuesday': '6:0-17:0', …               5
78  {'Monday': '11:0-21:0', 'Tuesday': '11:0-21:0'…              19
```

[6]:
```
# Distribution of words in reviews
arizona_reviews['num_words'].hist(bins=40)
plt.title("Distribution of The Number of Words in a Review")
plt.xlabel("Number of Words in Review (tip.json)")
plt.ylabel("Number of Reviews")
```

[6]:
```
Text(0, 0.5, 'Number of Reviews')
```

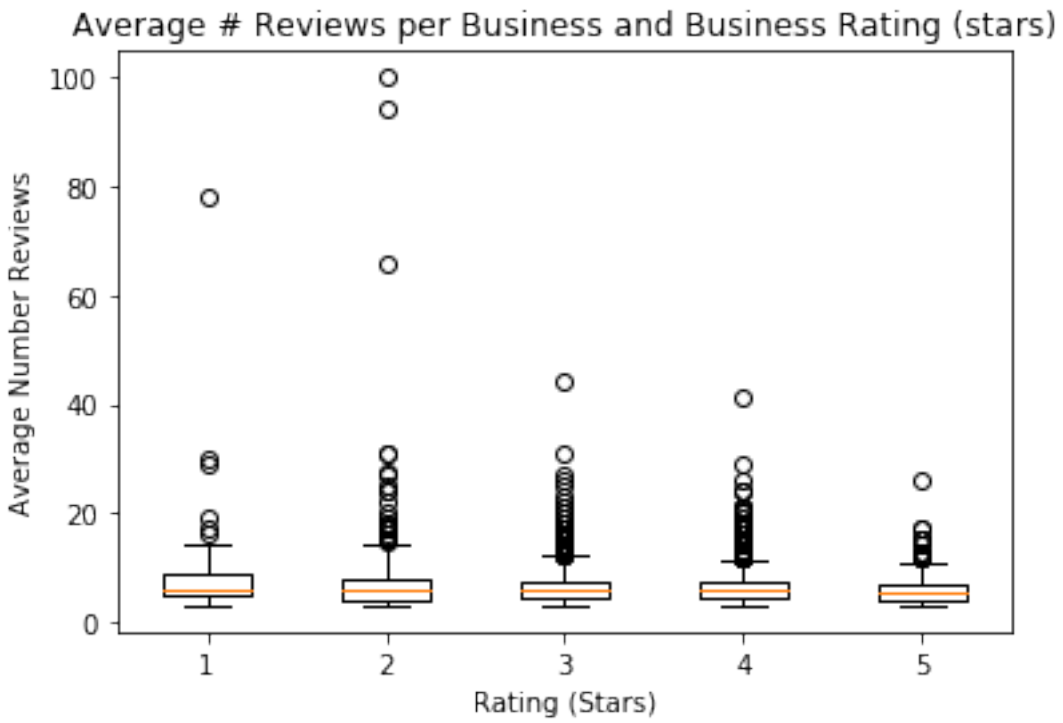


[7]:
```
# scatter plot of average number of words in review and the rating of the␣
 ↪business
grouped_reviews = arizona_reviews.groupby("business_id")
stats = grouped_reviews.describe()
review_means = stats['num_words']['mean']
```

[8]:
```
means = [[], [], [], [], []]
num_attributes = [[], [], [], [], []]
for index in arizona_businesses.index:
    rating = int(arizona_businesses['stars'][index])
    means[rating - 1].
 ↪append(review_means[arizona_businesses['business_id'][index]])
    num_attributes[rating - 1].
 ↪append(arizona_businesses['num_attributes'][index])
```

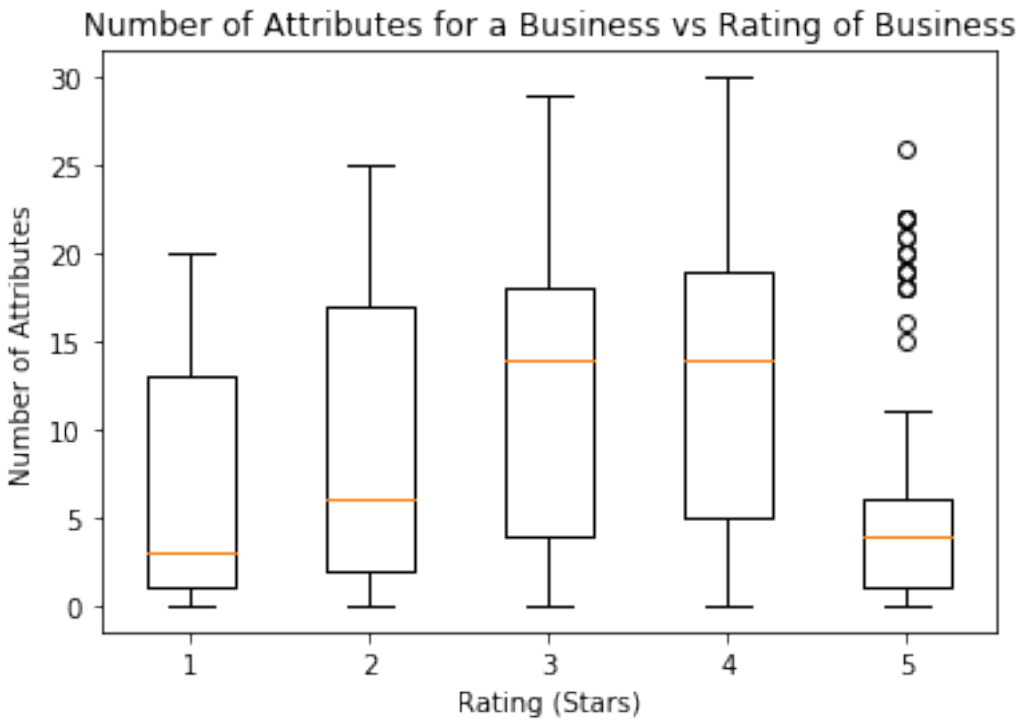
[9]:
```
plt.boxplot(means)
plt.title("Average # Reviews per Business and Business Rating (stars)")
plt.xlabel("Rating (Stars)")
plt.ylabel("Average Number Reviews")
```

[9]:
```
Text(0, 0.5, 'Average Number Reviews')
```



[10]:
```
plt.boxplot(num_attributes)
plt.title("Number of Attributes for a Business vs Rating of Business")
plt.xlabel("Rating (Stars)")
plt.ylabel("Number of Attributes")
```

[10]: Text(0, 0.5, 'Number of Attributes')



Data after Pre-Processing

[12]:
```
arizona_businesses.head()
```

[12]:

	business_id	name	address	city	state	postal_code	latitude	longitude	stars	review_count	is_open
0	1SWheh84yJXfytovILXOAQ	Arizona Biltmore Golf Club	2818 E Camino Acequia Drive	Phoenix	AZ	85016	33.522143	-112.018481	3.0	5	0
11	1Dfx3zM-rW4n-31KeC8sJg	Taco Bell	2450 E Indian School Rd	Phoenix	AZ	85016	33.495194	-112.028588	3.0	18	1
54	c-BELKj0SvNhBesQMf-bKw	Circle K	3101 W Northern Ave	Phoenix	AZ	85051	33.552850	-112.125975	2.5	3	1
72	HYunM2pknhIh8lbiMa7THw	Dunn-Edwards Paints	233 E Camelback Rd	Phoenix	AZ	85012	33.509011	-112.069667	3.0	16	1
78	44YFU284Z3KDEy25QyVoUw	Nee House Chinese Restaurant	13843 N Tatum Blvd, Ste 15	Phoenix	AZ	85032	33.613020				

```
78 -111.977036    3.5           269        1

                                          attributes  \
0                            {'GoodForKids': 'False'}
11  {'RestaurantsTakeOut': 'True', 'BusinessParkin…
54             {'BusinessAcceptsCreditCards': 'True'}
72  {'BusinessAcceptsCreditCards': 'True', 'ByAppo…
78  {'Caters': 'True', 'GoodForKids': 'True', 'Noi…

                                          categories  \
0                                  Golf, Active Life
11  Restaurants, Breakfast & Brunch, Mexican, Taco…
54  Convenience Stores, Automotive, Food, Gas Stat…
72  Interior Design, Contractors, Hardware Stores,…
78                               Chinese, Restaurants

                                               hours  num_attributes
0                                               None               1
11  {'Monday': '7:0-0:0', 'Tuesday': '7:0-0:0', 'W…              14
54  {'Monday': '0:0-0:0', 'Tuesday': '0:0-0:0', 'W…               1
72  {'Monday': '6:0-17:0', 'Tuesday': '6:0-17:0', …               5
78  {'Monday': '11:0-21:0', 'Tuesday': '11:0-21:0'…              19
```

[13]:
```
arizona_reviews.head()
```

[13]:
```
                   user_id             business_id  \
161  ucj0RT3ZsWHFXQssLq-wmQ  VcYpdJIoDgwBOdIJ_IvadA
232  2gu5B_JbJlfZLhihSr-apQ  pGLI8cMlay44bxYYjb4yIQ
373  ctXFXk9-m6PCMoI-Nz2_XQ  u4icCXldY7ALU2eV3wdPgw
413  8LHmg3kR_e1S-8WSxMyCdg  8yfNxQH1MOaPorSR_Lmj9g
479  yis0tDX83YnFLzUbKZ_KFg  veUZOPzegME87yvicpAWuw

                                                text                 date  \
161                   van gogh vroom vroom in my belly 2012-05-25 20:37:20
232                     checking out the wedding sight 2011-10-11 17:13:32
373  can be difficult getting in and out of this lo… 2016-05-04 08:39:28
413                                 avoid this place 2015-01-26 19:30:38
479                fire roasted burrito is delicious 2012-10-06 16:21:04

     compliment_count                                          tokenized  \
161                 0          [van, gogh, vroom, vroom, in, my, belly]
232                 0              [checking, out, the, wedding, sight]
373                 0  [can, be, difficult, getting, in, and, out, of…
413                 0                                  [avoid, this, place]
479                 0           [fire, roasted, burrito, is, delicious]

     num_words
```

```
161          7
232          5
373         13
413          3
479          5
```

[37]:
```
features = {}
for f in arizona_businesses['attributes'].index:
    attributes = arizona_businesses['attributes'][f]
    if attributes is not None:
        features.update(attributes)
```

[38]:
```
feature_mapping = {}
index = 0
# We want to go through each of the features and select ones that are booleans
for key in features:
    attributes = eval(features[key])

    # If we have a dictionary we have to add more features (business parking in␣
 ↪a garage, street, etc.)
    if type(attributes) == dict:
        feature_mapping[key] = {}
        for att in attributes:

            # Add only features that are booleans
            if type(attributes[att]) == bool:
                feature_mapping[key][att] = index
                index += 1

    # Add only features that are booleans
    elif type(attributes) == bool:
        feature_mapping[key] = index
        index += 1
```

[40]:
```
# After cleaning up, we see that our feature array will contain 73 different␣
 ↪possible features
num_features = index
feature_mapping
```

[40]:
```
{'GoodForKids': 0,
 'RestaurantsTakeOut': 1,
 'BusinessParking': {'garage': 2,
  'street': 3,
  'validated': 4,
  'lot': 5,
  'valet': 6},
 'RestaurantsDelivery': 7,
```

```
'OutdoorSeating': 8,
'BusinessAcceptsCreditCards': 9,
'RestaurantsGoodForGroups': 10,
'RestaurantsReservations': 11,
'HasTV': 12,
'Ambience': {'romantic': 13,
 'intimate': 14,
 'touristy': 15,
 'hipster': 16,
 'divey': 17,
 'classy': 18,
 'trendy': 19,
 'upscale': 20,
 'casual': 21},
'ByAppointmentOnly': 22,
'BikeParking': 23,
'Caters': 24,
'RestaurantsTableService': 25,
'GoodForMeal': {'dessert': 26,
 'latenight': 27,
 'lunch': 28,
 'dinner': 29,
 'brunch': 30,
 'breakfast': 31},
'DriveThru': 32,
'HairSpecializesIn': {'straightperms': 33,
 'coloring': 34,
 'extensions': 35,
 'africanamerican': 36,
 'curly': 37,
 'kids': 38,
 'perms': 39,
 'asian': 40},
'BusinessAcceptsBitcoin': 41,
'WheelchairAccessible': 42,
'BestNights': {'monday': 43,
 'tuesday': 44,
 'friday': 45,
 'wednesday': 46,
 'thursday': 47,
 'sunday': 48,
 'saturday': 49},
'GoodForDancing': 50,
'HappyHour': 51,
'Music': {'dj': 52,
 'background_music': 53,
 'no_music': 54,
```

```
 'jukebox': 55,
 'live': 56,
 'video': 57,
 'karaoke': 58},
'DogsAllowed': 59,
'CoatCheck': 60,
'Corkage': 61,
'AcceptsInsurance': 62,
'BYOB': 63,
'RestaurantsCounterService': 64,
'DietaryRestrictions': {'dairy-free': 65,
 'gluten-free': 66,
 'vegan': 67,
 'kosher': 68,
 'halal': 69,
 'soy-free': 70,
 'vegetarian': 71},
'Open24Hours': 72}
```

[96]:
```
# Creating our features and target arrays
def create_feature(attributes):
    # initializing our features array
    features = np.zeros(num_features)

    if attributes is None:
        return features

    # Go through all of the features for this dict
    for att in attributes:

        # check if the attribute is in our mapping
        if att in feature_mapping:

            # if it is a string just convert to 1 if 0 if false
            if type(feature_mapping[att]) != dict:
                val = eval(attributes[att])
                if val is not None:
                    features[feature_mapping[att]] = int(val)
            else:
                for sub_att in att:
                    if sub_att in feature_mapping[att]:
                        val = eval(attributes[att][sub_att])
                        if val is not None:
                            features[feature_mapping[att][sub_att]] = int(val)
    return features

features, targets = [], []
```

```
for business in arizona_businesses.index:
    features.append(create_feature(arizona_businesses['attributes'][business]))
    targets.append(1 if arizona_businesses['stars'][business] >= 4.5 else 0)
features = np.array(features)
targets = np.array(targets)
```

[97]:
```
# 5 fold cross validation
kfold = KFold(n_splits=5)

# Changing the meta parameters to find best possibility
learning_rates = [.1, .2, .3]
n_estimators = [50, 100, 150]
max_depth = [2,3,4]

roc_scores = []
accuracies = []
log_losses = []
confusion_matrices = []


for train_indices, testing_indices in kfold.split(features):
    # Grabbing the training and testing data
    training_features, training_targets =  features[train_indices],␣
 ↪targets[train_indices]
    testing_features, testing_targets = features[testing_indices],␣
 ↪targets[testing_indices]

    for l_r in learning_rates:
        for n_e in n_estimators:
            for m_d in max_depth:

                # Training and evaluating the performance of the model
                model = GradientBoostingClassifier(learning_rate=l_r,␣
 ↪n_estimators=n_e, max_depth=m_d)
                model.fit(training_features, training_targets)

                predictions = model.predict(testing_features)
                roc_scores.append(roc_auc_score(testing_targets, predictions))
                accuracies.append(accuracy_score(testing_targets, predictions))
                log_losses.append(log_loss(testing_targets, predictions))
                confusion_matrices.append(confusion_matrix(testing_targets,␣
 ↪predictions))
```

[98]:
```
print(f'The highest ROC_AUC that we achieved was {max(roc_scores)}')
print(f'The highest accuracy that we achieved was {max(accuracies)}')
print(f'The Confusion matrix with the highest␣
 ↪accuracy\n{confusion_matrices[accuracies.index(max(accuracies))]}')
```

```
print(f'The Confusion matrix with the highest ROC␣
 ↪AUC\n{confusion_matrices[roc_scores.index(max(roc_scores))]}')
print(f'The Confusion matrix with the lowest log␣
 ↪loss\n{confusion_matrices[log_losses.index(max(log_losses))]}')
```

```
The highest ROC_AUC that we achieved was 0.5699144982433026
The highest accuracy that we achieved was 0.8090010976948409
The Confusion matrix with the highest accuracy
[[719  15]
 [159  18]]
The Confusion matrix with the highest ROC AUC
[[663  41]
 [166  41]]
The Confusion matrix with the lowest log loss
[[666  34]
 [184  27]]
```