Final Project - Team 3

August 14, 2021

1 Final Project - Food.com Ratings Classification

Name: **Jimmy Nguyen**, **Jose Luis Estrada**, **Ashutosh Singh**

Class Assignment: **ADS 504 Final Project - Baseline Models**

2 Packages

[1]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import os
import json
import re
import random
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Perceptron
from sklearn.metrics import plot_confusion_matrix
from sklearn import tree
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import classification_report
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
import scipy.sparse
from sklearn.neural_network import MLPClassifier
```

```
from sklearn.linear_model import SGDClassifier

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score

import warnings # warnings packagedefine bust size
warnings.filterwarnings('ignore') # hide warnings
%matplotlib inline
plt.style.use('seaborn')
pd.set_option('display.max_colwidth', None)
```

---

3 Linear Classifier (Logistic) Model

3.1 Interactions Data

[2]:
```
interact = pd.read_csv("data/RAW_interactions.csv")
interact.head()
```

[2]:
```
   user_id  recipe_id        date  rating  \
0    38094      40893  2003-02-17       4
1  1293707      40893  2011-12-21       5
2     8937      44394  2002-12-01       4
3   126440      85009  2010-02-27       5
4    57222      85009  2011-10-01       5

                                        review
0                                                                    Great with
a salad. Cooked on top of stove for 15 minutes.Added a shake of cayenne and a
pinch of salt.  Used low fat sour cream.  Thanks.
1  So simple, so delicious! Great for chilly fall evening. Should have doubled
it ;)<br/><br/>Second time around, forgot the remaining cumin. We usually love
cumin, but didn't notice the missing 1/2 teaspoon!
2
This worked very well and is EASY.  I used not quite a whole package (10oz) of
white chips.  Great!
3
I made the Mexican topping and took it to bunko.  Everyone loved it.
4
Made the cheddar bacon topping, adding a sprinkling of black pepper. Yum!
```

3.1.1 Select only needed columns

[3]:
```
interact = interact[['recipe_id','review','rating']]
interact.head()
```

[3]:
```
   recipe_id  \
0      40893
1      40893
2      44394
3      85009
4      85009

                                          review  \
0                                                        Great with
a salad. Cooked on top of stove for 15 minutes.Added a shake of cayenne and a
pinch of salt.  Used low fat sour cream.  Thanks.
1  So simple, so delicious! Great for chilly fall evening. Should have doubled
it ;)<br/><br/>Second time around, forgot the remaining cumin. We usually love
cumin, but didn't notice the missing 1/2 teaspoon!
2
This worked very well and is EASY.  I used not quite a whole package (10oz) of
white chips.  Great!
3
I made the Mexican topping and took it to bunko.  Everyone loved it.
4
Made the cheddar bacon topping, adding a sprinkling of black pepper. Yum!

   rating
0       4
1       5
2       4
3       5
4       5
```

[4]:
```
interact.shape
```

[4]:
```
(1132367, 3)
```

3.2 Recipes Data

[5]:
```
recipes = pd.read_csv("data/RAW_recipes.csv")
recipes.head(1)
```

[5]:
```
                                         name      id  minutes  \
0  arriba   baked winter squash mexican style  137739       55

   contributor_id   submitted  \
```

```
0          47892  2005-09-16

                        tags  \
0  ['60-minutes-or-less', 'time-to-make', 'course', 'main-ingredient',
'cuisine', 'preparation', 'occasion', 'north-american', 'side-dishes',
'vegetables', 'mexican', 'easy', 'fall', 'holiday-event', 'vegetarian',
'winter', 'dietary', 'christmas', 'seasonal', 'squash']

                               nutrition  n_steps  \
0  [51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]       11

        steps  \
0  ['make a choice and proceed with recipe', 'depending on size of squash , cut
into half or fourths', 'remove seeds', 'for spicy squash , drizzle olive oil or
melted butter over each cut squash piece', 'season with mexican seasoning mix
ii', 'for sweet squash , drizzle melted honey , butter , grated piloncillo over
each cut squash piece', 'season with sweet mexican spice mix', 'bake at 350
degrees , again depending on size , for 40 minutes up to an hour , until a fork
can easily pierce the skin', 'be careful not to burn the squash especially if
you opt to use sugar or butter', 'if you feel more comfortable , cover the
squash with aluminum foil the first half hour , give or take , of baking', 'if
desired , season with salt']

                               description  \
0  autumn is my favorite time of year to cook! this recipe \r\ncan be prepared
either spicy or sweet, your choice!\r\ntwo of my posted mexican-inspired
seasoning mix recipes are offered as suggestions.

     ingredients  \
0  ['winter squash', 'mexican seasoning', 'mixed spice', 'honey', 'butter',
'olive oil', 'salt']

   n_ingredients
0              7
```

[6]:
```
recipes.shape
```

[6]:
```
(231637, 12)
```

3.2.1 Select only needed columns

[7]:
```
recipes = recipes[['id', 'name', 'minutes', 'nutrition', 'n_steps',␣
↪'ingredients', 'n_ingredients']]
recipes.head()
```

[7]:
```
        id                                          name  minutes  \
0   137739  arriba   baked winter squash mexican style       55
1    31490           a bit different  breakfast pizza       30
2   112140                 all in the kitchen  chili      130
3    59389                        alouette  potatoes       45
4    44061          amish  tomato ketchup  for canning      190

                                    nutrition  n_steps  \
0       [51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]       11
1   [173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0]        9
2  [269.8, 22.0, 32.0, 48.0, 39.0, 27.0, 5.0]        6
3   [368.1, 17.0, 10.0, 2.0, 14.0, 8.0, 20.0]       11
4   [352.9, 1.0, 337.0, 23.0, 3.0, 0.0, 28.0]        5

                        ingredients  \
0
['winter squash', 'mexican seasoning', 'mixed spice', 'honey', 'butter', 'olive
oil', 'salt']
1
['prepared pizza crust', 'sausage patty', 'eggs', 'milk', 'salt and pepper',
'cheese']
2  ['ground beef', 'yellow onions', 'diced tomatoes', 'tomato paste', 'tomato
soup', 'rotel tomatoes', 'kidney beans', 'water', 'chili powder', 'ground
cumin', 'salt', 'lettuce', 'cheddar cheese']
3         ['spreadable cheese with garlic and herbs', 'new potatoes',
'shallots', 'parsley', 'tarragon', 'olive oil', 'red wine vinegar', 'salt',
'pepper', 'red bell pepper', 'yellow bell pepper']
4
['tomato juice', 'apple cider vinegar', 'sugar', 'salt', 'pepper', 'clove oil',
'cinnamon oil', 'dry mustard']

   n_ingredients
0              7
1              6
2             13
3             11
4              8
```

3.3 Joining Recipes and Interactions Data based on Recipe id

[8]:
```
%%time
df = pd.merge(recipes, interact, how = "inner", left_on = 'id',
              right_on = 'recipe_id')
df.head()
```

```
CPU times: user 375 ms, sys: 48 ms, total: 423 ms
Wall time: 427 ms
```

[8]:
```
       id                                        name  minutes  \
0  137739  arriba   baked winter squash mexican style       55
1  137739  arriba   baked winter squash mexican style       55
2  137739  arriba   baked winter squash mexican style       55
3   31490           a bit different  breakfast pizza       30
4   31490           a bit different  breakfast pizza       30

                                  nutrition  n_steps  \
0      [51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]       11
1      [51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]       11
2      [51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]       11
3  [173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0]        9
4  [173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0]        9

     ingredients  \
0  ['winter squash', 'mexican seasoning', 'mixed spice', 'honey', 'butter',
'olive oil', 'salt']
1  ['winter squash', 'mexican seasoning', 'mixed spice', 'honey', 'butter',
'olive oil', 'salt']
2  ['winter squash', 'mexican seasoning', 'mixed spice', 'honey', 'butter',
'olive oil', 'salt']
3         ['prepared pizza crust', 'sausage patty', 'eggs', 'milk', 'salt and
pepper', 'cheese']
4         ['prepared pizza crust', 'sausage patty', 'eggs', 'milk', 'salt and
pepper', 'cheese']

   n_ingredients  recipe_id  \
0              7     137739
1              7     137739
2              7     137739
3              6      31490
4              6      31490

                                        review  \
0   I used an acorn squash and recipe#137681 Sweet Mexican spice blend. Only
used 1 tsp honey & 1 tsp butter between both halves,, sprinkled the squash
liberally with the spice mix. Baked covered for 45 minutes uncovered or 15.  I
basted the squash   with the the butter/honey from the cavity  allowing it to
get a golden color.  Lovely Squash recipe Thanks Cookgirl
1
This was a nice change. I used butternut squash and the sweet option using a
good local honey and unsalted butter. I did not add salt. We ate this on top of
recipe#322603 with Balkan yogurt. I may make this again same option. Made for
Ramadan Tag 2010.
2
Excellent recipe! I used butternut squash and the sweet option. The mexican
spice mix put this over the top. Thanks for sharing.
```

```
3                                          Have not tried this, but it sounds
delicious. Reminds me of a layover I had at the Atlanta airport.  I had a ham,
egg, and cheese pizza at one of the pizza chain places on the concourse.  About
$2.99 with coffee… It was one of the best breakfast dishes I ever had!  (But a
strange place to find a delicious breakfast…lol)
4
This recipe was wonderful.  Instead of using the precooked sausage I substituted
uncooked sausage then cooked and drained it. It turned out perfect!

   rating
0       5
1       5
2       5
3       0
4       5
```

[9]:
```
df.shape
```

[9]:
```
(1132367, 10)
```

3.4 Handling Null Values

[10]:
```
df.isnull().sum()
```

[10]:
```
id                0
name              1
minutes           0
nutrition         0
n_steps           0
ingredients       0
n_ingredients     0
recipe_id         0
review          169
rating            0
dtype: int64
```

[11]:
```
df = df.dropna()
df.isnull().sum()
```

[11]:
```
id              0
name            0
minutes         0
nutrition       0
n_steps         0
ingredients     0
n_ingredients   0
recipe_id       0
```

```
review           0
rating           0
dtype: int64
```

3.5 Convert Review Text into Features

[12]:
```
corpus_df = df[['review']]
corpus = corpus_df['review'].tolist()
corpus[:1]
```

[12]:
```
[' I used an acorn squash and recipe#137681 Sweet Mexican spice blend. Only used
1 tsp honey & 1 tsp butter between both halves,, sprinkled the squash liberally
with the spice mix. Baked covered for 45 minutes uncovered or 15.  I basted the
squash   with the the butter/honey from the cavity  allowing it to get a golden
color.  Lovely Squash recipe Thanks Cookgirl']
```

[13]:
```
%%time

vectorizer = TfidfVectorizer(stop_words='english')
reviews = vectorizer.fit_transform(corpus)
reviews = pd.DataFrame.sparse.from_spmatrix(reviews, columns = vectorizer.
 ↪get_feature_names())
reviews = reviews.drop(['wasn'], axis = 1)
reviews.head()
```

```
CPU times: user 1min 50s, sys: 710 ms, total: 1min 51s
Wall time: 1min 51s
```

[13]:

	00	000	0000	000000	0000001	0000laalaa	000170	000ft	000g	000mg	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	…	œvolcano	œwasteâ	œwe	œwhat	œwhiteâ	œwow	œyes	œzipâ	šo	šopsky
0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 151148 columns]

3.6 Transform Ratings with labelEncoder

[14]:
```
y = df['rating']
y.value_counts()
```

[14]:
```
5    816229
4    187333
0     60846
3     40852
2     14122
1     12815
Name: rating, dtype: int64
```

[15]:
```
y.shape
```

[15]:
```
(1132197,)
```

[16]:
```
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
rating_labels = [str(i) for i in le.classes_]
rating_labels
```

[16]:
```
['0', '1', '2', '3', '4', '5']
```

3.7 Feature Selection - Top 20 Features

[17]:
```
selector = SelectKBest(chi2, k=20).fit(reviews, y)
cols = selector.get_support(indices=True)
top_20 = reviews.iloc[:,cols]
top20 = pd.DataFrame(top_20.columns, columns = ["Top 20 Features"])
top20
```

[17]:

	Top 20 Features
0	awful
1	bad
2	bland
3	delicious
4	disappointed
5	disappointing
6	disgusting
7	good
8	horrible
9	liked
10	maybe
11	ok
12	okay
13	pretty

```
14          sorry
15       tasteless
16        terrible
17           think
18           waste
19           worst
```

[18]:
```
X = reviews.iloc[:,cols]
X.head()
```

[18]:

	awful	bad	bland	delicious	disappointed	disappointing	disgusting	\
0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.150443	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	

	good	horrible	liked	maybe	ok	okay	pretty	sorry	tasteless	\
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.076545	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	terrible	think	waste	worst
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

4 Baseline Model

4.0.1 L1

[19]:
```
alphas = [0.0001,0.001,0.01,0.1,1,10,100,1000]
penalty = 'l1'
```

[20]:
```
%%time
l1_scores = []

for a in alphas:
    log  = make_pipeline(SGDClassifier(loss = "log",
                                       penalty = penalty,
                                       alpha = a,
                                       #class_weight = "balanced",
                                       max_iter=1000,
```

```
                                        tol=1e-3,
                                        random_state = 1))
    print("Current Alpha:", a)
    avg_score = np.mean(cross_val_score(log, X, y,
                                         cv=5,scoring='accuracy',
                                         n_jobs=-1))
    print("Current Average Score:", avg_score)
    l1_scores.append(avg_score)
```

```
Current Alpha: 0.0001
Current Average Score: 0.7264548481903267
Current Alpha: 0.001
Current Average Score: 0.7206201746657727
Current Alpha: 0.01
Current Average Score: 0.7209248920479864
Current Alpha: 0.1
Current Average Score: 0.7209248920479864
Current Alpha: 1
Current Average Score: 0.7209248920479864
Current Alpha: 10
Current Average Score: 0.7209248920479864
Current Alpha: 100
Current Average Score: 0.3100177603259773
Current Alpha: 1000
Current Average Score: 0.3100177603259773
CPU times: user 1.7 s, sys: 353 ms, total: 2.05 s
Wall time: 2min 28s
```

4.0.2 L2

[21]:
```
penalty = 'l2'
```

[22]:
```
%%time
l2_scores = []

for a in alphas:
    log  = make_pipeline(SGDClassifier(loss = "log",
                                        penalty = penalty,
                                        alpha = a,
                                        #class_weight = "balanced",
                                        max_iter=1000,
                                        tol=1e-3,
                                        random_state = 1))
    print("Current Alpha:", a)
    avg_score = np.mean(cross_val_score(log, X, y,
                                         cv=5,scoring='accuracy',
```

```
                                          n_jobs=-1))
    print("Current Average Score:", avg_score)
    l2_scores.append(avg_score)
```

```
Current Alpha: 0.0001
Current Average Score: 0.7232354440976163
Current Alpha: 0.001
Current Average Score: 0.7207058488207105
Current Alpha: 0.01
Current Average Score: 0.7209248920479864
Current Alpha: 0.1
Current Average Score: 0.7209248920479864
Current Alpha: 1
Current Average Score: 0.7209248920479864
Current Alpha: 10
Current Average Score: 0.7209248920479864
Current Alpha: 100
Current Average Score: 0.3100177603259773
Current Alpha: 1000
Current Average Score: 0.3100177603259773
CPU times: user 1.69 s, sys: 213 ms, total: 1.91 s
Wall time: 2min 22s
```

4.1 Table Report

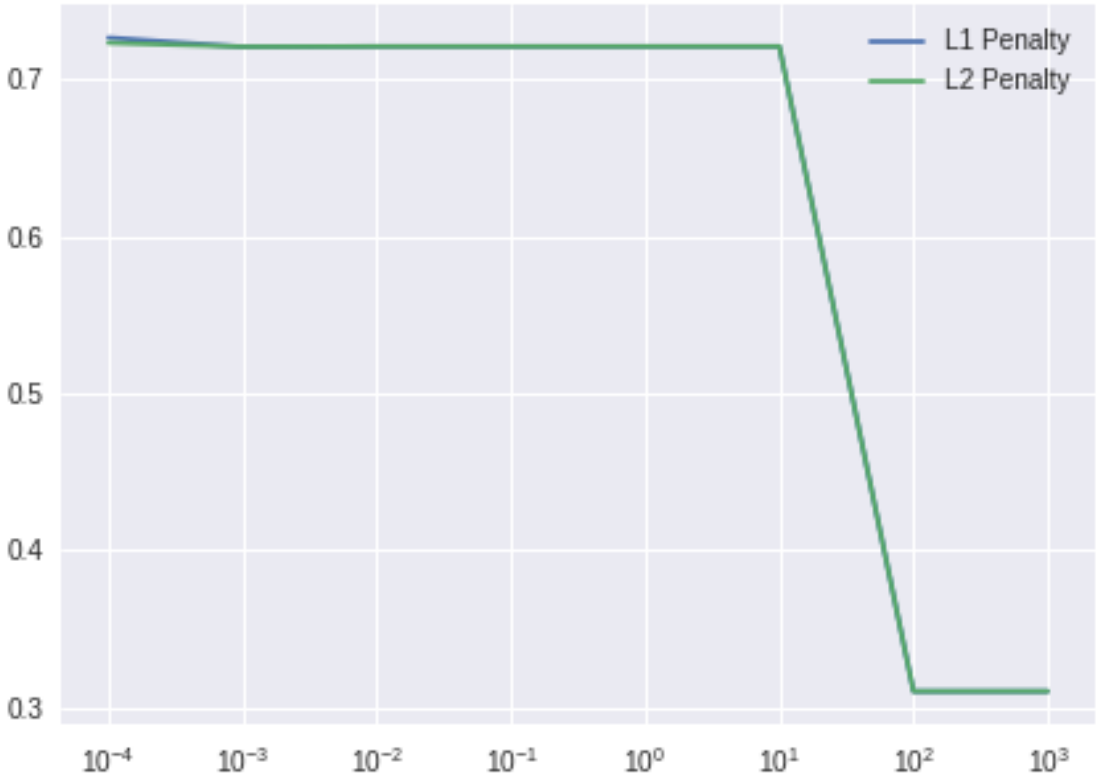
[23]:
```
table_report = {'Alphas':alphas,
                'L1 Penalty - Accuracy': l1_scores,
                'L2 Penalty - Accuracy': l2_scores}

table_df = pd.DataFrame(table_report)
table_df = table_df.set_index('Alphas')
table_df
```

[23]:

Alphas	L1 Penalty - Accuracy	L2 Penalty - Accuracy
0.0001	0.726455	0.723235
0.0010	0.720620	0.720706
0.0100	0.720925	0.720925
0.1000	0.720925	0.720925
1.0000	0.720925	0.720925
10.0000	0.720925	0.720925
100.0000	0.310018	0.310018
1000.0000	0.310018	0.310018

4.2 Regularization Accuracy Plot

[24]:
```
fig,ax=plt.subplots(figsize=(7,5))
ax.plot(table_df.index,
        table_df["L1 Penalty - Accuracy"],
       label = 'L1 Penalty')
ax.set(xscale="log")

ax.plot(table_df.index,
        table_df["L2 Penalty - Accuracy"],
       label = 'L2 Penalty')
ax.set(xscale="log")

plt.legend()
plt.show()
```



5 Improving the Baseline Model

5.1 Feature Engineering

[18]:
```
%%time
df[['calories',
    'total fat',
    'sugar','sodium',
    'protein',
    'saturated fat',
    'carbohydrates']] = df.nutrition.str.split(",",expand=True)

df['calories'] =  df['calories'].apply(lambda x: x.replace('[',''))
df['carbohydrates'] =  df['carbohydrates'].apply(lambda x: x.replace(']',''))

df[['calories',
    'total fat ',
    'sugar',
    'sodium',
    'protein',
    'saturated fat',
    'carbohydrates']] = df[['calories',
                            'total fat',
                            'sugar',
                            'sodium',
                            'protein',
                            'saturated fat',
                            'carbohydrates']].astype('float')


df.drop(['id', 'name', 'nutrition',
         'ingredients','recipe_id'], axis=1,inplace = True)
df = df.iloc[:,:-1]
df.head()
```

```
CPU times: user 5.57 s, sys: 604 ms, total: 6.17 s
Wall time: 6.17 s
```

[18]:

	minutes	n_steps	n_ingredients	\
0	55	11	7	
1	55	11	7	
2	55	11	7	
3	30	9	6	
4	30	9	6	

	review	\
0	I used an acorn squash and recipe#137681 Sweet Mexican spice blend. Only used 1 tsp honey & 1 tsp butter between both halves,, sprinkled the squash	

```
liberally with the spice mix. Baked covered for 45 minutes uncovered or 15.  I
basted the squash   with the the butter/honey from the cavity  allowing it to
get a golden color.  Lovely Squash recipe Thanks Cookgirl
1
This was a nice change. I used butternut squash and the sweet option using a
good local honey and unsalted butter. I did not add salt. We ate this on top of
recipe#322603 with Balkan yogurt. I may make this again same option. Made for
Ramadan Tag 2010.
2
Excellent recipe! I used butternut squash and the sweet option. The mexican
spice mix put this over the top. Thanks for sharing.
3                                        Have not tried this, but it sounds
delicious. Reminds me of a layover I had at the Atlanta airport.  I had a ham,
egg, and cheese pizza at one of the pizza chain places on the concourse.  About
$2.99 with coffee… It was one of the best breakfast dishes I ever had!  (But a
strange place to find a delicious breakfast…lol)
4
This recipe was wonderful.  Instead of using the precooked sausage I substituted
uncooked sausage then cooked and drained it. It turned out perfect!
```

	rating	calories	total fat	sugar	sodium	protein	saturated fat	carbohydrates
0	5	51.5	0.0	13.0	0.0	2.0	0.0	4.0
1	5	51.5	0.0	13.0	0.0	2.0	0.0	4.0
2	5	51.5	0.0	13.0	0.0	2.0	0.0	4.0
3	0	173.4	18.0	0.0	17.0	22.0	35.0	1.0
4	5	173.4	18.0	0.0	17.0	22.0	35.0	1.0

[19]:
```
df.shape
```

[19]:
```
(1132197, 12)
```

5.2 Combine with Review Data

[20]:
```
%%time
top_20 = reviews.iloc[:,cols]
final_df = pd.DataFrame(np.hstack([df, top_20]),
                        columns = df.columns.tolist() + top_20.columns.tolist())
final_df = final_df.drop(['review'], axis = 1)
```

```
CPU times: user 1.77 s, sys: 612 ms, total: 2.38 s
```

```
Wall time: 2.39 s
```

[21]:
```
final_df = final_df.drop_duplicates(subset=['minutes',
                            'n_steps',
                            'n_ingredients',
    'calories',
    'total fat',
    'sugar','sodium',
    'protein',
    'saturated fat',
    'carbohydrates'])

final_df.head()
```

[21]:

	minutes	n_steps	n_ingredients	rating	calories	total fat	sugar	sodium	\
0	55	11	7	5	51.5	0.0	13.0	0.0	
3	30	9	6	0	173.4	18.0	0.0	17.0	
7	130	6	13	4	269.8	22.0	32.0	48.0	
8	45	11	11	4	368.1	17.0	10.0	2.0	
10	190	5	8	5	352.9	1.0	337.0	23.0	

	protein	saturated fat	…	maybe	ok	okay	pretty	sorry	tasteless	terrible	\
0	2.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	22.0	35.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	39.0	27.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
8	14.0	8.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10	3.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	think	waste	worst
0	0.0	0.0	0.0
3	0.0	0.0	0.0
7	0.0	0.0	0.0
8	0.0	0.0	0.0
10	0.0	0.0	0.0

[5 rows x 31 columns]

[22]:
```
final_df.shape
```

[22]:
```
(231529, 31)
```

5.3 Final Linear Classifier

[23]:
```
X = final_df.drop(['rating'], axis = 1)
X.head()
```

[23]:
```
    minutes n_steps n_ingredients calories total fat  sugar sodium protein  \
0        55      11             7     51.5       0.0   13.0    0.0     2.0
3        30       9             6    173.4      18.0    0.0   17.0    22.0
7       130       6            13    269.8      22.0   32.0   48.0    39.0
8        45      11            11    368.1      17.0   10.0    2.0    14.0
10      190       5             8    352.9       1.0  337.0   23.0     3.0

    saturated fat carbohydrates  … maybe   ok okay pretty sorry tasteless  \
0             0.0           4.0  …   0.0  0.0  0.0    0.0   0.0       0.0
3            35.0           1.0  …   0.0  0.0  0.0    0.0   0.0       0.0
7            27.0           5.0  …   0.0  0.0  0.0    0.0   0.0       0.0
8             8.0          20.0  …   0.0  0.0  0.0    0.0   0.0       0.0
10            0.0          28.0  …   0.0  0.0  0.0    0.0   0.0       0.0

    terrible think waste worst
0        0.0   0.0   0.0   0.0
3        0.0   0.0   0.0   0.0
7        0.0   0.0   0.0   0.0
8        0.0   0.0   0.0   0.0
10       0.0   0.0   0.0   0.0

[5 rows x 30 columns]
```

[24]:
```
y = final_df['rating']
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
rating_labels = [str(i) for i in le.classes_]
rating_labels
```

[24]:
```
['0', '1', '2', '3', '4', '5']
```

5.3.1 L1

[25]:
```
alphas = [0.0001,0.001,0.01,0.1,1,10,100,1000]
penalty = 'l1'
```

[26]:
```
%%time
l1_scores = []

for a in alphas:
    log  = make_pipeline(StandardScaler(),
                         SGDClassifier(loss = "log",
                                       penalty = penalty,
                                       alpha = a,
                                       #class_weight = "balanced",
                                       max_iter=1000,
                                       tol=1e-3,
```

```
                                        random_state = 1))
    print("Current Alpha:", a)
    avg_score = np.mean(cross_val_score(log, X, y,
                                        cv=5,scoring='accuracy',
                                        n_jobs=-1))
    print("Current Average Score:", avg_score)
    l1_scores.append(avg_score)
```

```
Current Alpha: 0.0001
Current Average Score: 0.6984826961026986
Current Alpha: 0.001
Current Average Score: 0.698646818623901
Current Alpha: 0.01
Current Average Score: 0.6948762365297446
Current Alpha: 0.1
Current Average Score: 0.6904836975207898
Current Alpha: 1
Current Average Score: 0.6904836975207898
Current Alpha: 10
Current Average Score: 0.6904836975207898
Current Alpha: 100
Current Average Score: 0.36735214015760376
Current Alpha: 1000
Current Average Score: 0.36735214015760376
CPU times: user 19.7 s, sys: 3.07 s, total: 22.8 s
Wall time: 2min 5s
```

5.3.2 L2

[27]:
```
penalty = 'l2'
```

[28]:
```
%%time
l2_scores = []

for a in alphas:
    log  = make_pipeline(StandardScaler(),
                         SGDClassifier(loss = "log",
                                       penalty = penalty,
                                       alpha = a,
                                       #class_weight = "balanced",
                                       max_iter=1000,
                                       tol=1e-3,
                                       random_state = 1))
    print("Current Alpha:", a)
    avg_score = np.mean(cross_val_score(log, X, y,
                                        cv=5,
```

```
                                        scoring='accuracy',
                                        n_jobs=-1))
    print("Current Average Score:", avg_score)
    l2_scores.append(avg_score)
```

```
Current Alpha: 0.0001
Current Average Score: 0.6980291835087925
Current Alpha: 0.001
Current Average Score: 0.6986727338451318
Current Alpha: 0.01
Current Average Score: 0.6986900094778036
Current Alpha: 0.1
Current Average Score: 0.6950921931311302
Current Alpha: 1
Current Average Score: 0.6904966548982179
Current Alpha: 10
Current Average Score: 0.6904923358035001
Current Alpha: 100
Current Average Score: 0.36734350196816823
Current Alpha: 1000
Current Average Score: 0.36734350196816823
CPU times: user 20.1 s, sys: 2.66 s, total: 22.8 s
Wall time: 1min 3s
```

5.3.3 Table Report

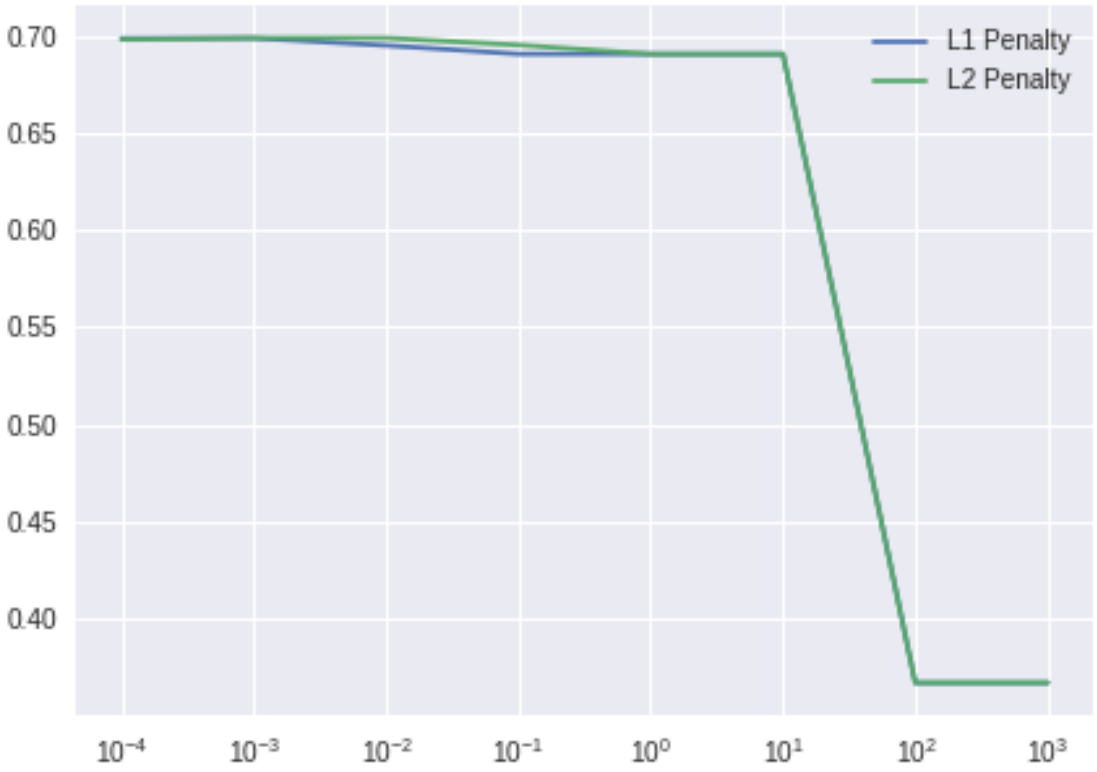
[29]:
```
table_report = {'Alphas':alphas,
                'L1 Penalty - Accuracy': l1_scores,
                'L2 Penalty - Accuracy': l2_scores}

table_df = pd.DataFrame(table_report)
table_df = table_df.set_index('Alphas')
table_df
```

[29]:

Alphas	L1 Penalty - Accuracy	L2 Penalty - Accuracy
0.0001	0.698483	0.698029
0.0010	0.698647	0.698673
0.0100	0.694876	0.698690
0.1000	0.690484	0.695092
1.0000	0.690484	0.690497
10.0000	0.690484	0.690492
100.0000	0.367352	0.367344
1000.0000	0.367352	0.367344

5.3.4 Regularization Accuracy Plot

[30]:
```
fig,ax=plt.subplots(figsize=(7,5))
ax.plot(table_df.index,
        table_df["L1 Penalty - Accuracy"],
       label = 'L1 Penalty')
ax.set(xscale="log")

ax.plot(table_df.index,
        table_df["L2 Penalty - Accuracy"],
       label = 'L2 Penalty')
ax.set(xscale="log")

plt.legend()
plt.show()
```



5.4 Final Model

[31]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
 ↪random_state = 1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

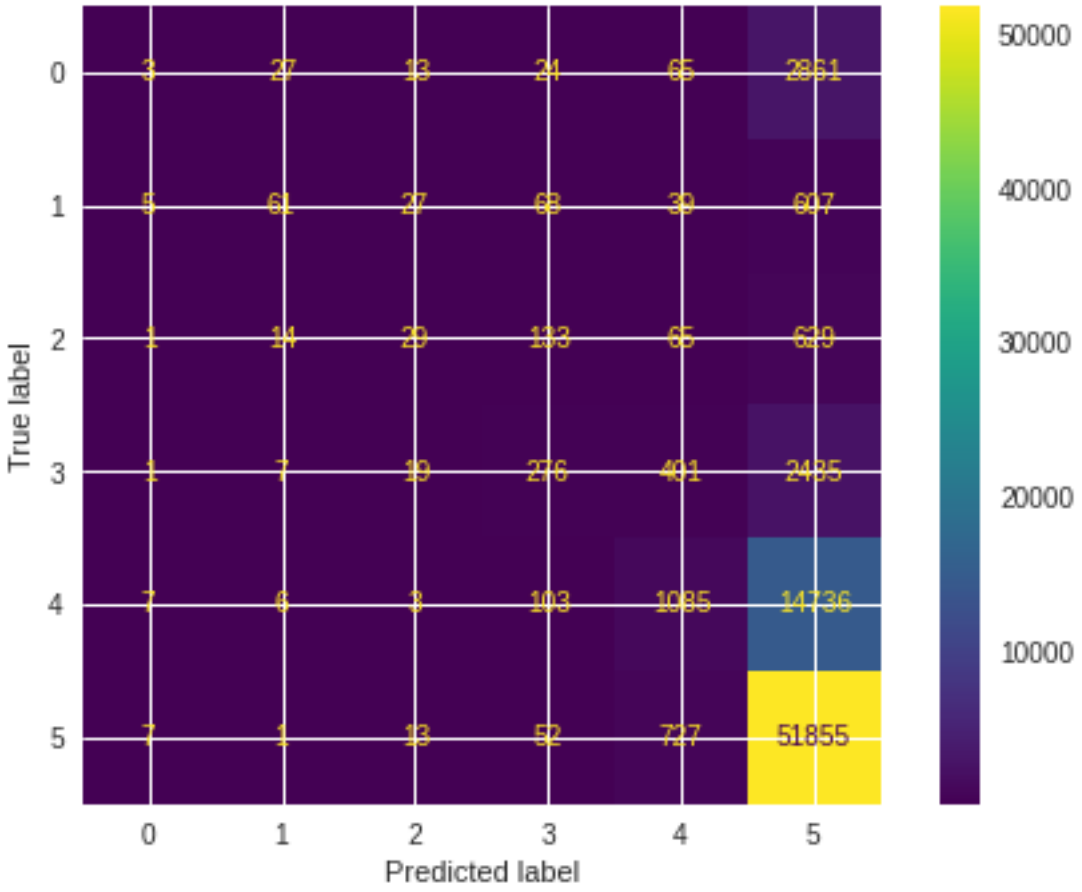
```
(155124, 30) (76405, 30) (155124,) (76405,)
```

[32]:
```
# Optimal Parameters
penalty = 'l1'
alpha = 0.0001

# Model
log  = make_pipeline(StandardScaler(),
                     SGDClassifier(loss = "log",
                                   penalty = penalty,
                                   alpha = alpha,
                                   max_iter=1000,
                                   tol=1e-3,
                                   random_state = 1))
# Fit Model
log.fit(X_train,y_train)

# make predictions
ypred = log.predict(X_test)
# evaluate predictions
acc = accuracy_score(y_test, ypred)
print('Accuracy: %.3f' % acc)
```

```
Accuracy: 0.698
```

5.5 Logistic Model Confusion Matrix

[33]:
```
plot_confusion_matrix(log, X_test, y_test)
plt.show()
```



5.6 Logistic Model Classification Report

[34]:
```
print(classification_report(y_test, ypred))
```

```
              precision    recall  f1-score   support

           0       0.12      0.00      0.00      2993
           1       0.53      0.08      0.13       807
           2       0.28      0.03      0.06       871
           3       0.42      0.09      0.15      3139
           4       0.46      0.07      0.12     15940
           5       0.71      0.98      0.82     52655

    accuracy                           0.70     76405
   macro avg       0.42      0.21      0.21     76405
weighted avg       0.61      0.70      0.60     76405
```

Neural Networks Model

August 14, 2021

---

1 Loading Data

[3]:
```
interact = pd.read_csv("data/RAW_interactions.csv")
interact.head()
```

[3]:
```
   user_id  recipe_id        date  rating  \
0    38094      40893  2003-02-17       4
1  1293707      40893  2011-12-21       5
2     8937      44394  2002-12-01       4
3   126440      85009  2010-02-27       5
4    57222      85009  2011-10-01       5

                                              review
0  Great with a salad. Cooked on top of stove for…
1  So simple, so delicious! Great for chilly fall…
2  This worked very well and is EASY.  I used not…
3  I made the Mexican topping and took it to bunk…
4  Made the cheddar bacon topping, adding a sprin…
```

1.1 Using only Reviews to Predict Ratings

[4]:
```
df = interact[['rating', 'review']]
```

1.2 Handling Null Values

[5]:
```
df.isnull().sum()
```

[5]:
```
rating      0
review    169
dtype: int64
```

[6]:
```
df = df.dropna()
df.head()
```

[6]:
```
   rating                                             review
0       4  Great with a salad. Cooked on top of stove for…
1       5  So simple, so delicious! Great for chilly fall…
2       4  This worked very well and is EASY.  I used not…
3       5  I made the Mexican topping and took it to bunk…
4       5  Made the cheddar bacon topping, adding a sprin…
```

[7]:
```
df.isnull().sum()
```

[7]:
```
rating    0
review    0
dtype: int64
```

1.3 Convert Review Text into Features

[8]:
```
review = df['review'].tolist()
review[:3]
```

[8]:
```
['Great with a salad. Cooked on top of stove for 15 minutes.Added a shake of
cayenne and a pinch of salt.  Used low fat sour cream.  Thanks.',
 "So simple, so delicious! Great for chilly fall evening. Should have doubled it
;)<br/><br/>Second time around, forgot the remaining cumin. We usually love
cumin, but didn't notice the missing 1/2 teaspoon!",
 'This worked very well and is EASY.  I used not quite a whole package (10oz) of
white chips.  Great!']
```

[9]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import scipy.sparse

vectorizer = TfidfVectorizer()
review = vectorizer.fit_transform(review)
review = pd.DataFrame.sparse.from_spmatrix(review, columns = vectorizer.
 ↪get_feature_names())
review.head()
```

[9]:
```
    00  000  0000  000000  0000001  0000laalaa  000170  000ft  000g  000mg  \
0  0.0  0.0   0.0     0.0      0.0          0.0     0.0    0.0   0.0    0.0
1  0.0  0.0   0.0     0.0      0.0          0.0     0.0    0.0   0.0    0.0
2  0.0  0.0   0.0     0.0      0.0          0.0     0.0    0.0   0.0    0.0
3  0.0  0.0   0.0     0.0      0.0          0.0     0.0    0.0   0.0    0.0
4  0.0  0.0   0.0     0.0      0.0          0.0     0.0    0.0   0.0    0.0

   …  œvolcano  œwasteâ  œwe  œwhat  œwhiteâ  œwow  œyes  œzipâ   šo  šopsky
0  …       0.0      0.0  0.0    0.0      0.0   0.0   0.0    0.0  0.0     0.0
1  …       0.0      0.0  0.0    0.0      0.0   0.0   0.0    0.0  0.0     0.0
2  …       0.0      0.0  0.0    0.0      0.0   0.0   0.0    0.0  0.0     0.0
3  …       0.0      0.0  0.0    0.0      0.0   0.0   0.0    0.0  0.0     0.0
```

```
4  …       0.0       0.0  0.0     0.0       0.0    0.0    0.0     0.0  0.0      0.0

[5 rows x 151459 columns]
```

1.4 Transform Ratings with labelEncoder

[10]:
```
y = df['rating']
y.value_counts()
```

[10]:
```
5    816229
4    187333
0     60847
3     40852
2     14122
1     12815
Name: rating, dtype: int64
```

[11]:
```
labels = df['rating']
y = labels
le = preprocessing.LabelEncoder()
le.fit(y)
y=le.transform(y)
u_labels = [str(i) for i in le.classes_]
u_labels
```

[11]:
```
['0', '1', '2', '3', '4', '5']
```

1.5 Feature Engineering - Top 50 Features

[106]:
```
selector = SelectKBest(chi2, k=50).fit(review, y)
```

[107]:
```
cols = selector.get_support(indices=True)
```

[108]:
```
top_50 = review.iloc[:,cols]
top_50 = pd.DataFrame(top_50.columns, columns = ["Top 50 Features"])
top_50.head()
```

[108]:
```
  Top 50 Features
0           awful
1             bad
2             bit
3           bland
4             but
```

[109]:
```
review = review.iloc[:,cols]
review.head()
```

[109]:
```
   awful  bad  bit  bland       but  care  delicious  disappointed  \
0    0.0  0.0  0.0    0.0  0.000000   0.0   0.000000           0.0
1    0.0  0.0  0.0    0.0  0.076732   0.0   0.101998           0.0
2    0.0  0.0  0.0    0.0  0.000000   0.0   0.000000           0.0
3    0.0  0.0  0.0    0.0  0.000000   0.0   0.000000           0.0
4    0.0  0.0  0.0    0.0  0.000000   0.0   0.000000           0.0

   disappointing  disgusting  …  thought  too  wasn  waste  wasted  \
0            0.0         0.0  …      0.0  0.0   0.0    0.0     0.0
1            0.0         0.0  …      0.0  0.0   0.0    0.0     0.0
2            0.0         0.0  …      0.0  0.0   0.0    0.0     0.0
3            0.0         0.0  …      0.0  0.0   0.0    0.0     0.0
4            0.0         0.0  …      0.0  0.0   0.0    0.0     0.0

   wonderful  worst  would  wrong  yuck
0        0.0    0.0    0.0    0.0   0.0
1        0.0    0.0    0.0    0.0   0.0
2        0.0    0.0    0.0    0.0   0.0
3        0.0    0.0    0.0    0.0   0.0
4        0.0    0.0    0.0    0.0   0.0

[5 rows x 50 columns]
```

[110]:
```
from sklearn.neural_network import MLPClassifier
```

[111]:
```
clf = MLPClassifier(hidden_layer_sizes=(10,5), max_iter=1000, alpha=1e-4,␣
↪solver='lbfgs',
                    verbose=10,random_state=42, activation = 'relu',␣
↪early_stopping = False,
                    n_iter_no_change=100)
clf.fit(review,y)


clf.score(review, y)
```

[111]:
```
0.7415955513081635
```

[112]:
```
rat_rec = interact[['rating', 'recipe_id']]
review = pd.concat([rat_rec, review], axis=1, ignore_index = True)
review = review.rename(columns = {0:'rating', 1:'id'})
review.head()
```

[112]:
```
   rating     id    2    3    4    5         6    7         8    9  …   42  \
0       4  40893  0.0  0.0  0.0  0.0  0.000000  0.0  0.000000  0.0  …  0.0
1       5  40893  0.0  0.0  0.0  0.0  0.076732  0.0  0.101998  0.0  …  0.0
2       4  44394  0.0  0.0  0.0  0.0  0.000000  0.0  0.000000  0.0  …  0.0
3       5  85009  0.0  0.0  0.0  0.0  0.000000  0.0  0.000000  0.0  …  0.0
```

```
4       5  85009  0.0  0.0  0.0  0.0  0.000000  0.0  0.000000  0.0  …  0.0

    43   44   45   46   47   48   49   50   51
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 52 columns]
```

---

2 Raw Recipes - Improving baseline classifier

[113]:
```
recipes = pd.read_csv("RAW_recipes.csv")
```

[114]:
```
recipes.isnull().sum()
```

[114]:
```
name                1
id                  0
minutes             0
contributor_id      0
submitted           0
tags                0
nutrition           0
n_steps             0
steps               0
description      4979
ingredients         0
n_ingredients       0
dtype: int64
```

[115]:
```
recipes = recipes.dropna()
```

[116]:
```
recipes.isnull().sum()
```

[116]:
```
name             0
id               0
minutes          0
contributor_id   0
submitted        0
tags             0
nutrition        0
n_steps          0
steps            0
```

```
description       0
ingredients       0
n_ingredients     0
dtype: int64
```

2.1 Separate Nutrition Array into Variables

[117]:
```
def tonutrition(n):
    n = n[1:-1]
    erase = n.split(', ')
    erase = [float(i) for i in erase]
    return erase

recipes['nutrition'] = recipes.nutrition.astype(str).apply(tonutrition)
```

[118]:
```
#Separate arrays into 7 columns
recipes[['calories','total_fat','sugar','sodium','protein','sat_fat','other']]␣
 ↪= \
pd.DataFrame(recipes.nutrition.tolist(), index= recipes.index)
```

2.2 Merge Dataset

[119]:
```
recipes = recipes.drop(['contributor_id','submitted','nutrition'], axis=1)
```

[120]:
```
merged_df = pd.merge(recipes, review, on = 'id')
```

2.3 Label Encoder for Target Variable

[123]:
```
y=merged_df['rating']
le = preprocessing.LabelEncoder()
le.fit(y)
y=le.transform(y)
u_labels = [str(i) for i in le.classes_]
```

[124]:
```
y = pd.DataFrame(y)
y = y.rename(columns = {0:'rating'})
```

[125]:
```
merged_df = merged_df.drop(['id', 'rating'], axis=1)
```

2.4 Tags Variable into Features

[126]:
```
def totags(n):
    n = n[1:-1]
    erase = n.split(', ')
    return ' '.join(erase)
```

```
tags = pd.DataFrame(merged_df['tags'])

tags['tags'] = tags.tags.astype(str).apply(totags)
tags.head()
```

[126]:
```
                                                tags
0  '60-minutes-or-less' 'time-to-make' 'course' '…
1  '60-minutes-or-less' 'time-to-make' 'course' '…
2  '60-minutes-or-less' 'time-to-make' 'course' '…
3  '30-minutes-or-less' 'time-to-make' 'course' '…
4  '30-minutes-or-less' 'time-to-make' 'course' '…
```

[127]:
```
vectorizer = TfidfVectorizer()
tags = tags['tags'].tolist()
tags = vectorizer.fit_transform(tags)
tags = pd.DataFrame.sparse.from_spmatrix(tags, columns = vectorizer.
 ↪get_feature_names())
tags.head()
```

[127]:
```
    15        30        60   a1  african  ahead  alcoholic  american  amish  \
0  0.0  0.000000  0.157328  0.0      0.0    0.0        0.0  0.170086    0.0
1  0.0  0.000000  0.157328  0.0      0.0    0.0        0.0  0.170086    0.0
2  0.0  0.000000  0.157328  0.0      0.0    0.0        0.0  0.170086    0.0
3  0.0  0.167489  0.000000  0.0      0.0    0.0        0.0  0.325059    0.0
4  0.0  0.167489  0.000000  0.0      0.0    0.0        0.0  0.325059    0.0

   and  …    winter  with  wrap  yams  year  years  yeast  yellow  zealand  \
0  0.0  …  0.298591   0.0   0.0   0.0   0.0    0.0    0.0     0.0      0.0
1  0.0  …  0.298591   0.0   0.0   0.0   0.0    0.0    0.0     0.0      0.0
2  0.0  …  0.298591   0.0   0.0   0.0   0.0    0.0    0.0     0.0      0.0
3  0.0  …  0.000000   0.0   0.0   0.0   0.0    0.0    0.0     0.0      0.0
4  0.0  …  0.000000   0.0   0.0   0.0   0.0    0.0    0.0     0.0      0.0

   zucchini
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0

[5 rows x 593 columns]
```

[128]:
```
selector_tag = SelectKBest(chi2, k=50).fit(tags, y)
```

[129]:
```
cols_tag = selector_tag.get_support(indices=True)
```

[130]:
```
top_tag_50 = tags.iloc[:,cols_tag]
top_tag_50 = pd.DataFrame(top_tag_50.columns, columns = ["Top 50 Features"])
top_tag_50.head()
```

[130]:
```
  Top 50 Features
0            bath
1           beans
2            beef
3       beverages
4        brownies
```

[131]:
```
tags = tags.iloc[:,cols_tag]
tags.head()
```

[131]:
```
   bath  beans  beef  beverages  brownies  cakes  candy  canning  casseroles  \
0   0.0    0.0   0.0        0.0       0.0    0.0    0.0      0.0         0.0
1   0.0    0.0   0.0        0.0       0.0    0.0    0.0      0.0         0.0
2   0.0    0.0   0.0        0.0       0.0    0.0    0.0      0.0         0.0
3   0.0    0.0   0.0        0.0       0.0    0.0    0.0      0.0         0.0
4   0.0    0.0   0.0        0.0       0.0    0.0    0.0      0.0         0.0

   chicken  …  salads  sauces      side  simply  slow  valley  vegetables  \
0      0.0  …     0.0     0.0  0.223228     0.0   0.0     0.0    0.177391
1      0.0  …     0.0     0.0  0.223228     0.0   0.0     0.0    0.177391
2      0.0  …     0.0     0.0  0.223228     0.0   0.0     0.0    0.177391
3      0.0  …     0.0     0.0  0.000000     0.0   0.0     0.0    0.000000
4      0.0  …     0.0     0.0  0.000000     0.0   0.0     0.0    0.000000

   waffles  water  yeast
0      0.0    0.0    0.0
1      0.0    0.0    0.0
2      0.0    0.0    0.0
3      0.0    0.0    0.0
4      0.0    0.0    0.0

[5 rows x 50 columns]
```

[132]:
```
df3 = pd.concat([merged_df, tags], axis=1, ignore_index=True)
df3.head()
```

[132]:
```
                                        0    1    \
0  arriba   baked winter squash mexican style   55
1  arriba   baked winter squash mexican style   55
2  arriba   baked winter squash mexican style   55
3           a bit different  breakfast pizza   30
4           a bit different  breakfast pizza   30
```

```
                                                   2    3    \
0  ['60-minutes-or-less', 'time-to-make', 'course…   11
1  ['60-minutes-or-less', 'time-to-make', 'course…   11
2  ['60-minutes-or-less', 'time-to-make', 'course…   11
3  ['30-minutes-or-less', 'time-to-make', 'course…    9
4  ['30-minutes-or-less', 'time-to-make', 'course…    9

                                                   4    \
0  ['make a choice and proceed with recipe', 'dep…
1  ['make a choice and proceed with recipe', 'dep…
2  ['make a choice and proceed with recipe', 'dep…
3  ['preheat oven to 425 degrees f', 'press dough…
4  ['preheat oven to 425 degrees f', 'press dough…

                                                   5    \
0  autumn is my favorite time of year to cook! th…
1  autumn is my favorite time of year to cook! th…
2  autumn is my favorite time of year to cook! th…
3  this recipe calls for the crust to be prebaked…
4  this recipe calls for the crust to be prebaked…

                                                   6    7      8     9    …  \
0  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
1  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
2  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
3  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …
4  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …

   105  106       107  108  109  110       111  112  113  114
0  0.0  0.0  0.223228  0.0  0.0  0.0  0.177391  0.0  0.0  0.0
1  0.0  0.0  0.223228  0.0  0.0  0.0  0.177391  0.0  0.0  0.0
2  0.0  0.0  0.223228  0.0  0.0  0.0  0.177391  0.0  0.0  0.0
3  0.0  0.0  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.0
4  0.0  0.0  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.0

[5 rows x 115 columns]
```

[133]:
```
df3.isnull().sum().sum()
```

[133]:
```
8450
```

2.5 Steps Variable into Features

[134]:
```
def tosteps(n):
    n = n[1:-1]
    erase = n.split(', ')
    return ' '.join(erase)
```

```
steps = pd.DataFrame(merged_df['steps'])

steps['steps'] = steps.steps.astype(str).apply(tosteps)
steps.head()
```

[134]:

	steps
0	'make a choice and proceed with recipe' 'depen…
1	'make a choice and proceed with recipe' 'depen…
2	'make a choice and proceed with recipe' 'depen…
3	'preheat oven to 425 degrees f' 'press dough i…
4	'preheat oven to 425 degrees f' 'press dough i…

[135]:
```
vectorizer = TfidfVectorizer()
steps = steps['steps'].tolist()
steps = vectorizer.fit_transform(steps)
steps = pd.DataFrame.sparse.from_spmatrix(steps, columns = vectorizer.
 ↪get_feature_names())
steps.head()
```

[135]:

	00	000	000ft	001	0016	008	00am	01	02	04	…	zup	zuppa	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	

	zuppainglese	zuzu	zweiback	zwetschgen	zwetschgendatschi	\
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	

	zwetschgenkuchen	zwieback	zyliss
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0

[5 rows x 42799 columns]

[136]:
```
selector_steps = SelectKBest(chi2, k=50).fit(steps, y)
cols_steps = selector_steps.get_support(indices=True)
```

[137]:
```
top_steps_50 = steps.iloc[:,cols_steps]
top_steps_50 = pd.DataFrame(top_steps_50.columns, columns = ["Top 50 Features"])
top_steps_50.head()
```

[137]:
```
  Top 50 Features
0          batter
1            beat
2            beef
3        broccoli
4            cake
```

[138]:
```
steps = steps.iloc[:,cols_steps]
steps.head()
```

[138]:
```
   batter  beat  beef  broccoli  cake  casserole  chicken  chops  cook  \
0     0.0   0.0   0.0       0.0   0.0        0.0      0.0    0.0   0.0
1     0.0   0.0   0.0       0.0   0.0        0.0      0.0    0.0   0.0
2     0.0   0.0   0.0       0.0   0.0        0.0      0.0    0.0   0.0
3     0.0   0.0   0.0       0.0   0.0        0.0      0.0    0.0   0.0
4     0.0   0.0   0.0       0.0   0.0        0.0      0.0    0.0   0.0

   cooker  …  tofu  toss  uglier  vanilla  vinegarfor  whites  will  worth  \
0     0.0  …   0.0   0.0     0.0      0.0         0.0     0.0   0.0    0.0
1     0.0  …   0.0   0.0     0.0      0.0         0.0     0.0   0.0    0.0
2     0.0  …   0.0   0.0     0.0      0.0         0.0     0.0   0.0    0.0
3     0.0  …   0.0   0.0     0.0      0.0         0.0     0.0   0.0    0.0
4     0.0  …   0.0   0.0     0.0      0.0         0.0     0.0   0.0    0.0

        you  your
0  0.081335   0.0
1  0.081335   0.0
2  0.081335   0.0
3  0.000000   0.0
4  0.000000   0.0

[5 rows x 50 columns]
```

[139]:
```
df4 = pd.concat([df3, steps], axis=1, ignore_index=True)
df4.head()
```

[139]:
```
                                        0    1    \
0  arriba   baked winter squash mexican style   55
1  arriba   baked winter squash mexican style   55
2  arriba   baked winter squash mexican style   55
3           a bit different  breakfast pizza   30
4           a bit different  breakfast pizza   30
```

```
                                                   2    3    \
0  ['60-minutes-or-less', 'time-to-make', 'course…   11
1  ['60-minutes-or-less', 'time-to-make', 'course…   11
2  ['60-minutes-or-less', 'time-to-make', 'course…   11
3  ['30-minutes-or-less', 'time-to-make', 'course…    9
4  ['30-minutes-or-less', 'time-to-make', 'course…    9

                                                   4    \
0  ['make a choice and proceed with recipe', 'dep…
1  ['make a choice and proceed with recipe', 'dep…
2  ['make a choice and proceed with recipe', 'dep…
3  ['preheat oven to 425 degrees f', 'press dough…
4  ['preheat oven to 425 degrees f', 'press dough…

                                                   5    \
0  autumn is my favorite time of year to cook! th…
1  autumn is my favorite time of year to cook! th…
2  autumn is my favorite time of year to cook! th…
3  this recipe calls for the crust to be prebaked…
4  this recipe calls for the crust to be prebaked…

                                                   6    7      8     9    … \
0  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
1  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
2  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
3  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …
4  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …

   155  156  157  158  159  160  161  162       163  164
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.081335  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.081335  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.081335  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0

[5 rows x 165 columns]
```

[140]:
```
df4.isnull().sum().sum()
```

[140]:
```
8450
```

2.6 Ingredients Variable into Features

[141]:
```
def toingredients(n):
    n = n[1:-1]
    erase = n.split(', ')
    return ' '.join(erase)
```

```
ingredients = pd.DataFrame(merged_df['ingredients'])

ingredients['ingredients'] = ingredients.ingredients.astype(str).
 ↪apply(toingredients)
ingredients.head()
```

[141]:
```
                                          ingredients
0  'winter squash' 'mexican seasoning' 'mixed spi…
1  'winter squash' 'mexican seasoning' 'mixed spi…
2  'winter squash' 'mexican seasoning' 'mixed spi…
3  'prepared pizza crust' 'sausage patty' 'eggs' …
4  'prepared pizza crust' 'sausage patty' 'eggs' …
```

[142]:
```
vectorizer = TfidfVectorizer()
ingredients = ingredients['ingredients'].tolist()
ingredients = vectorizer.fit_transform(ingredients)
ingredients = pd.DataFrame.sparse.from_spmatrix(ingredients, columns =␣
 ↪vectorizer.get_feature_names())
ingredients.head()
```

[142]:
```
    10  100  10x   12   13   15  151   16   18   21  …  zero  zest  zesty  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0    0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0    0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0    0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0    0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0    0.0

   zinfandel  zinger  ziploc  ziti  zoom  zucchini  zwieback
0        0.0     0.0     0.0   0.0   0.0       0.0       0.0
1        0.0     0.0     0.0   0.0   0.0       0.0       0.0
2        0.0     0.0     0.0   0.0   0.0       0.0       0.0
3        0.0     0.0     0.0   0.0   0.0       0.0       0.0
4        0.0     0.0     0.0   0.0   0.0       0.0       0.0

[5 rows x 4160 columns]
```

[143]:
```
selector_ingredients = SelectKBest(chi2, k=50).fit(ingredients, y)
cols_ingredients = selector_ingredients.get_support(indices=True)
```

[144]:
```
top_ingredients_50 = ingredients.iloc[:,cols_ingredients]
top_ingredients_50 = pd.DataFrame(top_ingredients_50.columns, columns = ["Top␣
 ↪50 Features"])
top_ingredients_50.head()
```

[144]:
```
  Top 50 Features
0          active
```

```
1            all
2      applesauce
3          baking
4            beef
```

[145]:
```
ingredients = ingredients.iloc[:,cols_ingredients]
ingredients.head()
```

[145]:
```
   active  all  applesauce  baking  beef  boneless  bread  breasts  broccoli  \
0     0.0  0.0         0.0     0.0   0.0       0.0    0.0      0.0       0.0
1     0.0  0.0         0.0     0.0   0.0       0.0    0.0      0.0       0.0
2     0.0  0.0         0.0     0.0   0.0       0.0    0.0      0.0       0.0
3     0.0  0.0         0.0     0.0   0.0       0.0    0.0      0.0       0.0
4     0.0  0.0         0.0     0.0   0.0       0.0    0.0      0.0       0.0

   chicken  …  splenda  substitute  sugar  tartar  tofu  vanilla  water  \
0      0.0  …      0.0         0.0    0.0     0.0   0.0      0.0    0.0
1      0.0  …      0.0         0.0    0.0     0.0   0.0      0.0    0.0
2      0.0  …      0.0         0.0    0.0     0.0   0.0      0.0    0.0
3      0.0  …      0.0         0.0    0.0     0.0   0.0      0.0    0.0
4      0.0  …      0.0         0.0    0.0     0.0   0.0      0.0    0.0

   whipping  whites  yeast
0       0.0     0.0    0.0
1       0.0     0.0    0.0
2       0.0     0.0    0.0
3       0.0     0.0    0.0
4       0.0     0.0    0.0

[5 rows x 50 columns]
```

[146]:
```
df5 = pd.concat([df4, ingredients], axis=1, ignore_index=True)
df5.head()
```

[146]:
```
                                          0    1    \
0  arriba   baked winter squash mexican style   55
1  arriba   baked winter squash mexican style   55
2  arriba   baked winter squash mexican style   55
3           a bit different  breakfast pizza   30
4           a bit different  breakfast pizza   30

                                                 2    3    \
0  ['60-minutes-or-less', 'time-to-make', 'course…   11
1  ['60-minutes-or-less', 'time-to-make', 'course…   11
2  ['60-minutes-or-less', 'time-to-make', 'course…   11
3  ['30-minutes-or-less', 'time-to-make', 'course…    9
4  ['30-minutes-or-less', 'time-to-make', 'course…    9
```

```
                                                   4    \
0  ['make a choice and proceed with recipe', 'dep…
1  ['make a choice and proceed with recipe', 'dep…
2  ['make a choice and proceed with recipe', 'dep…
3  ['preheat oven to 425 degrees f', 'press dough…
4  ['preheat oven to 425 degrees f', 'press dough…

                                                   5    \
0  autumn is my favorite time of year to cook! th…
1  autumn is my favorite time of year to cook! th…
2  autumn is my favorite time of year to cook! th…
3  this recipe calls for the crust to be prebaked…
4  this recipe calls for the crust to be prebaked…

                                                   6    7      8     9    …  \
0  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
1  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
2  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
3  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …
4  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …

   205  206  207  208  209  210  211  212  213  214
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 215 columns]
```

[147]:
```
df5.isnull().sum().sum()
```

[147]:
```
8450
```

2.7 Description Variable into Features

[148]:
```
description = pd.DataFrame(merged_df['description'])

description['description'] = description.description.astype(str)
description.head()
```

[148]:
```
                                         description
0  autumn is my favorite time of year to cook! th…
1  autumn is my favorite time of year to cook! th…
2  autumn is my favorite time of year to cook! th…
3  this recipe calls for the crust to be prebaked…
```

```
4  this recipe calls for the crust to be prebaked…
```

[149]:
```
vectorizer = TfidfVectorizer()
description = description['description'].tolist()
description = vectorizer.fit_transform(description)
description = pd.DataFrame.sparse.from_spmatrix(description, columns =␣
 ↪vectorizer.get_feature_names())
description.head()
```

[149]:
```
    00  000  000037moms_roast_turkey  000186  000ft  000th  001  \
0  0.0  0.0                      0.0     0.0    0.0    0.0  0.0
1  0.0  0.0                      0.0     0.0    0.0    0.0  0.0
2  0.0  0.0                      0.0     0.0    0.0    0.0  0.0
3  0.0  0.0                      0.0     0.0    0.0    0.0  0.0
4  0.0  0.0                      0.0     0.0    0.0    0.0  0.0

   001712roasted_garlic  005178mexican_red_chili_sauce  0060586141  …  \
0                   0.0                            0.0         0.0  …
1                   0.0                            0.0         0.0  …
2                   0.0                            0.0         0.0  …
3                   0.0                            0.0         0.0  …
4                   0.0                            0.0         0.0  …

   çilbir  èšª  épices  érable  évora  órexi  örebro  über  œuvre  šaltiena
0     0.0  0.0     0.0     0.0    0.0    0.0     0.0   0.0    0.0       0.0
1     0.0  0.0     0.0     0.0    0.0    0.0     0.0   0.0    0.0       0.0
2     0.0  0.0     0.0     0.0    0.0    0.0     0.0   0.0    0.0       0.0
3     0.0  0.0     0.0     0.0    0.0    0.0     0.0   0.0    0.0       0.0
4     0.0  0.0     0.0     0.0    0.0    0.0     0.0   0.0    0.0       0.0

[5 rows x 69418 columns]
```

[150]:
```
selector_description = SelectKBest(chi2, k=50).fit(description, y)
cols_description = selector_description.get_support(indices=True)
```

[151]:
```
top_description_50 = description.iloc[:,cols_description]
top_description_50 = pd.DataFrame(top_description_50.columns, columns = ["Top␣
 ↪50 Features"])
top_description_50.head()
```

[151]:
```
  Top 50 Features
0          289860
1        absoutly
2           again
3       attacking
4       biography
```

[152]:
```
description = description.iloc[:,cols_description]
description.head()
```

[152]:
```
   289860  absoutly  again  attacking  biography  bouilion  burst  cafemon  \
0     0.0       0.0    0.0        0.0        0.0       0.0    0.0      0.0
1     0.0       0.0    0.0        0.0        0.0       0.0    0.0      0.0
2     0.0       0.0    0.0        0.0        0.0       0.0    0.0      0.0
3     0.0       0.0    0.0        0.0        0.0       0.0    0.0      0.0
4     0.0       0.0    0.0        0.0        0.0       0.0    0.0      0.0

   cake  churns  …  snackie  spookiest  stumbled  susi  teaaspoon  \
0   0.0     0.0  …      0.0        0.0       0.0   0.0        0.0
1   0.0     0.0  …      0.0        0.0       0.0   0.0        0.0
2   0.0     0.0  …      0.0        0.0       0.0   0.0        0.0
3   0.0     0.0  …      0.0        0.0       0.0   0.0        0.0
4   0.0     0.0  …      0.0        0.0       0.0   0.0        0.0

   unbelieveable  unpluged  vegetarian  vegweb  yummyyy
0            0.0       0.0         0.0     0.0      0.0
1            0.0       0.0         0.0     0.0      0.0
2            0.0       0.0         0.0     0.0      0.0
3            0.0       0.0         0.0     0.0      0.0
4            0.0       0.0         0.0     0.0      0.0

[5 rows x 50 columns]
```

[153]:
```
df6 = pd.concat([df5, description], axis=1, ignore_index=True)
df6.head()
```

[153]:
```
                                          0    1    \
0  arriba   baked winter squash mexican style   55
1  arriba   baked winter squash mexican style   55
2  arriba   baked winter squash mexican style   55
3           a bit different  breakfast pizza   30
4           a bit different  breakfast pizza   30

                                                 2    3    \
0  ['60-minutes-or-less', 'time-to-make', 'course…   11
1  ['60-minutes-or-less', 'time-to-make', 'course…   11
2  ['60-minutes-or-less', 'time-to-make', 'course…   11
3  ['30-minutes-or-less', 'time-to-make', 'course…    9
4  ['30-minutes-or-less', 'time-to-make', 'course…    9

                                                 4    \
0  ['make a choice and proceed with recipe', 'dep…
1  ['make a choice and proceed with recipe', 'dep…
2  ['make a choice and proceed with recipe', 'dep…
```

```
3  ['preheat oven to 425 degrees f', 'press dough…
4  ['preheat oven to 425 degrees f', 'press dough…

                                                5    \
0  autumn is my favorite time of year to cook! th…
1  autumn is my favorite time of year to cook! th…
2  autumn is my favorite time of year to cook! th…
3  this recipe calls for the crust to be prebaked…
4  this recipe calls for the crust to be prebaked…

                                                6    7      8     9    …  \
0  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
1  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
2  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
3  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …
4  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …

   255  256  257  258  259  260  261  262  263  264
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 265 columns]
```

[154]:
```
df6.isnull().sum().sum()
```

[154]:
```
8450
```

2.8 Name Variable into Features

[155]:
```
name = pd.DataFrame(merged_df['name'])

name['name'] = name.name.astype(str)
name.head()
```

[155]:
```
                                        name
0  arriba   baked winter squash mexican style
1  arriba   baked winter squash mexican style
2  arriba   baked winter squash mexican style
3            a bit different  breakfast pizza
4            a bit different  breakfast pizza
```

[156]:
```
vectorizer = TfidfVectorizer()
name = name['name'].tolist()
name = vectorizer.fit_transform(name)
```

```
name = pd.DataFrame.sparse.from_spmatrix(name, columns = vectorizer.
 ↪get_feature_names())
name.head()
```

[156]:

	00	000	001	007	00pm	07	08	09	10	100	…	zwiebelsosse	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	

	zwiebelsuppe	zwiebelwhe	zwina	zwt	zwt3	zwt6	zwtiii	zydeco	zzzingers
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 28442 columns]

[157]:
```
selector_name = SelectKBest(chi2, k=50).fit(name, y)
cols_name = selector_name.get_support(indices=True)
```

[158]:
```
top_name_25 = name.iloc[:,cols_name]
top_name_25 = pd.DataFrame(top_name_25.columns, columns = ["Top 25 Features"])
top_name_25.head()
```

[158]:

	Top 25 Features
0	and
1	athens
2	authentic
3	best
4	bread

[159]:
```
name = name.iloc[:,cols_name]
name.head()
```

[159]:

	and	athens	authentic	best	bread	cake	canning	carb	casserole	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	chicken	…	skillet	slow	strietzel	sugar	tofu	torrone	vegan	with	\
0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

```
2      0.0  …      0.0   0.0        0.0    0.0   0.0      0.0    0.0   0.0
3      0.0  …      0.0   0.0        0.0    0.0   0.0      0.0    0.0   0.0
4      0.0  …      0.0   0.0        0.0    0.0   0.0      0.0    0.0   0.0

   yeast  yum
0    0.0  0.0
1    0.0  0.0
2    0.0  0.0
3    0.0  0.0
4    0.0  0.0

[5 rows x 50 columns]
```

[160]:
```
df6 = pd.concat([df5, name], axis=1, ignore_index=True)
df6.head()
```

[160]:
```
                                          0    1    \
0  arriba   baked winter squash mexican style   55
1  arriba   baked winter squash mexican style   55
2  arriba   baked winter squash mexican style   55
3           a bit different  breakfast pizza   30
4           a bit different  breakfast pizza   30

                                                 2    3    \
0  ['60-minutes-or-less', 'time-to-make', 'course…   11
1  ['60-minutes-or-less', 'time-to-make', 'course…   11
2  ['60-minutes-or-less', 'time-to-make', 'course…   11
3  ['30-minutes-or-less', 'time-to-make', 'course…    9
4  ['30-minutes-or-less', 'time-to-make', 'course…    9

                                                 4    \
0  ['make a choice and proceed with recipe', 'dep…
1  ['make a choice and proceed with recipe', 'dep…
2  ['make a choice and proceed with recipe', 'dep…
3  ['preheat oven to 425 degrees f', 'press dough…
4  ['preheat oven to 425 degrees f', 'press dough…

                                                 5    \
0  autumn is my favorite time of year to cook! th…
1  autumn is my favorite time of year to cook! th…
2  autumn is my favorite time of year to cook! th…
3  this recipe calls for the crust to be prebaked…
4  this recipe calls for the crust to be prebaked…

                                                 6    7      8     9    …  \
0  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
1  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
```

```
2  ['winter squash', 'mexican seasoning', 'mixed …    7   51.5   0.0  …
3  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …
4  ['prepared pizza crust', 'sausage patty', 'egg…    6  173.4  18.0  …

   255  256  257  258  259  260  261  262  263  264
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 265 columns]
```

[161]:
```
df_final = df6.drop([0,2,4,5,6], axis=1)
```

[162]:
```
df_final.isnull().sum().sum()
```

[162]:
```
8450
```

2.8.1 Neural Network Model

[164]:
```
rows_with_nan = [index for index, row in df_final.iterrows() if row.isnull().
 ↪any()]
len(rows_with_nan)
```

[164]:
```
169
```

[165]:
```
df_final = df_final.drop(rows_with_nan)
y = y.drop(rows_with_nan)
```

[166]:
```
df_final.isnull().sum().sum()
```

[166]:
```
0
```

[167]:
```
selector_final = SelectKBest(chi2, k=50).fit(df_final, y)
cols_final = selector_final.get_support(indices=True)
```

[168]:
```
top_final = df_final.iloc[:,cols_final]
top_final = pd.DataFrame(top_final.columns, columns = ["Top 50 Features"])
top_final.head()
```

[168]:
```
   Top 50 Features
0                1
1                3
2                8
3                9
4               10
```

[169]:
```
df_final = df_final.iloc[:,cols_final]
df_final.head()
```

[169]:

	1	3	8	9	10	11	12	13	14	69	…	226	227	\
0	55	11	51.5	0.0	13.0	0.0	2.0	0.0	4.0	0.0	…	0.0	0.0	
1	55	11	51.5	0.0	13.0	0.0	2.0	0.0	4.0	0.0	…	0.0	0.0	
2	55	11	51.5	0.0	13.0	0.0	2.0	0.0	4.0	0.0	…	0.0	0.0	
3	30	9	173.4	18.0	0.0	17.0	22.0	35.0	1.0	0.0	…	0.0	0.0	
4	30	9	173.4	18.0	0.0	17.0	22.0	35.0	1.0	0.0	…	0.0	0.0	

	228	229	232	238	241	248	254	261
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 50 columns]

[170]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit_transform(df_final)
```

[170]:
```
array([[-0.00414643,  0.23341534, -0.39376813, …, -0.18327358,
        -0.21401863, -0.0716367 ],
       [-0.00414643,  0.23341534, -0.39376813, …, -0.18327358,
        -0.21401863, -0.0716367 ],
       [-0.00414643,  0.23341534, -0.39376813, …, -0.18327358,
        -0.21401863, -0.0716367 ],
       …,
       [-0.00415037, -0.79660913, -0.27455361, …, -0.18327358,
        -0.21401863, -0.0716367 ],
       [-0.00415037, -0.79660913, -0.27455361, …, -0.18327358,
        -0.21401863, -0.0716367 ],
       [-0.00415037, -0.79660913, -0.27455361, …, -0.18327358,
        -0.21401863, -0.0716367 ]])
```

[171]:
```
X_train, X_test, y_train, y_test = train_test_split(df_final, y, test_size=0.2,␣
↪random_state=42)
```

[172]:
```
y_train = pd.DataFrame(y_train)
y_train.head()
```

[172]:

	rating
17687	5
179130	5
350840	0

```
758804       3
644346       5
```

[178]:
```
from sklearn.neural_network import MLPClassifier
```

[186]:
```
clf = MLPClassifier(hidden_layer_sizes=(10,5), max_iter=1000, alpha=1e-4,␣
 ↪solver='sgd',
                    verbose=10,random_state=42, activation = 'relu',␣
 ↪early_stopping = False,
                    n_iter_no_change=100)
clf.fit(X_train,y_train)
clf.predict(X_test)

clf.score(X_train, y_train)
clf.score(X_test, y_test)
```

```
Iteration 1, loss = 0.96893517
Iteration 2, loss = 0.91494147
Iteration 3, loss = 0.91485595
Iteration 4, loss = 0.91476388
Iteration 5, loss = 0.91472802
Iteration 6, loss = 0.91468460
Iteration 7, loss = 0.91465727
Iteration 8, loss = 0.91458451
Iteration 9, loss = 0.91448973
Iteration 10, loss = 0.91447950
Iteration 11, loss = 0.91447592
Iteration 12, loss = 0.91447313
Iteration 13, loss = 0.91447649
Iteration 14, loss = 0.91446966
Iteration 15, loss = 0.91446923
Iteration 16, loss = 0.91447120
Iteration 17, loss = 0.91447487
Iteration 18, loss = 0.91447199
Iteration 19, loss = 0.91446831
Iteration 20, loss = 0.91447290
Iteration 21, loss = 0.91446965
Iteration 22, loss = 0.91446798
Iteration 23, loss = 0.91447612
Iteration 24, loss = 0.91446937
Iteration 25, loss = 0.91446573
Iteration 26, loss = 0.91447282
Iteration 27, loss = 0.91447030
Iteration 28, loss = 0.91448432
Iteration 29, loss = 0.91447048
Iteration 30, loss = 0.91446931
Iteration 31, loss = 0.91447192
```

```
Iteration 32, loss = 0.91447109
Iteration 33, loss = 0.91446717
Iteration 34, loss = 0.91446263
Iteration 35, loss = 0.91447109
Iteration 36, loss = 0.91447373
Iteration 37, loss = 0.91446672
Iteration 38, loss = 0.91446741
Iteration 39, loss = 0.91447228
Iteration 40, loss = 0.91446714
Iteration 41, loss = 0.91446649
Iteration 42, loss = 0.91446876
Iteration 43, loss = 0.91446704
Iteration 44, loss = 0.91446739
Iteration 45, loss = 0.91446426
Iteration 46, loss = 0.91446705
Iteration 47, loss = 0.91446664
Iteration 48, loss = 0.91446878
Iteration 49, loss = 0.91446894
Iteration 50, loss = 0.91447025
Iteration 51, loss = 0.91447155
Iteration 52, loss = 0.91446508
Iteration 53, loss = 0.91446563
Iteration 54, loss = 0.91446717
Iteration 55, loss = 0.91446775
Iteration 56, loss = 0.91446962
Iteration 57, loss = 0.91446740
Iteration 58, loss = 0.91446847
Iteration 59, loss = 0.91446415
Iteration 60, loss = 0.91446698
Iteration 61, loss = 0.91447036
Iteration 62, loss = 0.91446857
Iteration 63, loss = 0.91446393
Iteration 64, loss = 0.91446585
Iteration 65, loss = 0.91447243
Iteration 66, loss = 0.91446901
Iteration 67, loss = 0.91445710
Iteration 68, loss = 0.91446418
Iteration 69, loss = 0.91446683
Iteration 70, loss = 0.91446584
Iteration 71, loss = 0.91446952
Iteration 72, loss = 0.91446838
Iteration 73, loss = 0.91446817
Iteration 74, loss = 0.91446600
Iteration 75, loss = 0.91446282
Iteration 76, loss = 0.91446310
Iteration 77, loss = 0.91447197
Iteration 78, loss = 0.91446872
Iteration 79, loss = 0.91446890
```

```
Iteration 80, loss = 0.91446618
Iteration 81, loss = 0.91446278
Iteration 82, loss = 0.91446227
Iteration 83, loss = 0.91446419
Iteration 84, loss = 0.91446643
Iteration 85, loss = 0.91446678
Iteration 86, loss = 0.91446275
Iteration 87, loss = 0.91446395
Iteration 88, loss = 0.91446247
Iteration 89, loss = 0.91446636
Iteration 90, loss = 0.91446551
Iteration 91, loss = 0.91446206
Iteration 92, loss = 0.91445876
Iteration 93, loss = 0.91445855
Iteration 94, loss = 0.91446180
Iteration 95, loss = 0.91446392
Iteration 96, loss = 0.91446836
Iteration 97, loss = 0.91445828
Iteration 98, loss = 0.91446071
Iteration 99, loss = 0.91446443
Iteration 100, loss = 0.91446219
Iteration 101, loss = 0.91446466
Iteration 102, loss = 0.91445287
Iteration 103, loss = 0.91446253
Training loss did not improve more than tol=0.000100 for 100 consecutive epochs.
Stopping.
```

[186]:
```
0.7214189719398569
```

[182]:
```
clf = MLPClassifier(hidden_layer_sizes=(10,5), max_iter=1000, alpha=1e-4,␣
↪solver='adam',
                    verbose=10,random_state=42, activation = 'relu',␣
↪early_stopping = False,
                    n_iter_no_change=100)
clf.fit(X_train,y_train)
clf.predict(X_test)

clf.score(X_train, y_train)
clf.score(X_test, y_test)
```

```
Iteration 1, loss = 1.50126534
Iteration 2, loss = 0.93801335
Iteration 3, loss = 0.92553468
Iteration 4, loss = 0.91768900
Iteration 5, loss = 0.91450310
Iteration 6, loss = 0.91448309
Iteration 7, loss = 0.91447967
Iteration 8, loss = 0.91447218
```

```
Iteration 9, loss = 0.91446882
Iteration 10, loss = 0.91446851
Iteration 11, loss = 0.91450462
Iteration 12, loss = 0.91446278
Iteration 13, loss = 0.91446427
Iteration 14, loss = 0.91446037
Iteration 15, loss = 0.91445867
Iteration 16, loss = 0.91448284
Iteration 17, loss = 0.91446605
Iteration 18, loss = 0.91446364
Iteration 19, loss = 0.91447499
Iteration 20, loss = 0.91446497
Iteration 21, loss = 0.91446072
Iteration 22, loss = 0.91445626
Iteration 23, loss = 0.91446626
Iteration 24, loss = 0.91446134
Iteration 25, loss = 0.91445693
Iteration 26, loss = 0.91446553
Iteration 27, loss = 0.91446204
Iteration 28, loss = 0.91446488
Iteration 29, loss = 0.91446048
Iteration 30, loss = 0.91446261
Iteration 31, loss = 0.91446417
Iteration 32, loss = 0.91446319
Iteration 33, loss = 0.91445998
Iteration 34, loss = 0.91445748
Iteration 35, loss = 0.91446534
Iteration 36, loss = 0.91446771
Iteration 37, loss = 0.91445978
Iteration 38, loss = 0.91446053
Iteration 39, loss = 0.91446394
Iteration 40, loss = 0.91446191
Iteration 41, loss = 0.91446139
Iteration 42, loss = 0.91446140
Iteration 43, loss = 0.91445977
Iteration 44, loss = 0.91446303
Iteration 45, loss = 0.91445845
Iteration 46, loss = 0.91446076
Iteration 47, loss = 0.91445921
Iteration 48, loss = 0.91446226
Iteration 49, loss = 0.91446247
Iteration 50, loss = 0.91446480
Iteration 51, loss = 0.91446781
Iteration 52, loss = 0.91446106
Iteration 53, loss = 0.91445983
Iteration 54, loss = 0.91446075
Iteration 55, loss = 0.91446311
Iteration 56, loss = 0.91446527
```

```
Iteration 57, loss = 0.91446258
Iteration 58, loss = 0.91446316
Iteration 59, loss = 0.91445844
Iteration 60, loss = 0.91446274
Iteration 61, loss = 0.91446544
Iteration 62, loss = 0.91446487
Iteration 63, loss = 0.91445715
Iteration 64, loss = 0.91445988
Iteration 65, loss = 0.91446525
Iteration 66, loss = 0.91446353
Iteration 67, loss = 0.91445503
Iteration 68, loss = 0.91446055
Iteration 69, loss = 0.91446144
Iteration 70, loss = 0.91446201
Iteration 71, loss = 0.91446458
Iteration 72, loss = 0.91446155
Iteration 73, loss = 0.91446261
Iteration 74, loss = 0.91446157
Iteration 75, loss = 0.91445816
Iteration 76, loss = 0.91445904
Iteration 77, loss = 0.91446917
Iteration 78, loss = 0.91446821
Iteration 79, loss = 0.91446450
Iteration 80, loss = 0.91445987
Iteration 81, loss = 0.91445956
Iteration 82, loss = 0.91445661
Iteration 83, loss = 0.91446180
Iteration 84, loss = 0.91446207
Iteration 85, loss = 0.91446244
Iteration 86, loss = 0.91445635
Iteration 87, loss = 0.91446041
Iteration 88, loss = 0.91446134
Iteration 89, loss = 0.91446552
Iteration 90, loss = 0.91446481
Iteration 91, loss = 0.91445858
Iteration 92, loss = 0.91445750
Iteration 93, loss = 0.91445756
Iteration 94, loss = 0.91445949
Iteration 95, loss = 0.91446121
Iteration 96, loss = 0.91446656
Iteration 97, loss = 0.91445670
Iteration 98, loss = 0.91445844
Iteration 99, loss = 0.91446439
Iteration 100, loss = 0.91445936
Iteration 101, loss = 0.91446227
Iteration 102, loss = 0.91445074
Iteration 103, loss = 0.91445746
Iteration 104, loss = 0.91446657
```

```
Iteration 105, loss = 0.91446335
Iteration 106, loss = 0.91446500
Training loss did not improve more than tol=0.000100 for 100 consecutive epochs.
Stopping.
```

[182]: 0.7214189719398569

[183]:
```
clf.score(X_train, y_train)
```

[183]: 0.7218092584804763

[184]:
```
clf.score(X_test, y_test)
```

[184]: 0.7214189719398569

Random Forest Model

August 14, 2021

0.0.1 Random forest

[114]:
```
from sklearn.ensemble import RandomForestClassifier
```

[115]:
```
random_clf = RandomForestClassifier(max_depth=2, random_state=42)
```

[116]:
```
random_score = np.mean(cross_val_score(random_clf, df_final, y,
                                       cv=5,scoring='accuracy',
                                       n_jobs=-1))
```

[117]:
```
random_score
```

[117]:
```
0.7217312009630328
```

[122]:
```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
```

[123]:
```
y_pred = cross_val_predict(random_clf, df_final, y, cv=10)
conf_mat = confusion_matrix(y, y_pred)
```

[124]:
```
conf_mat
```

[124]:
```
array([[     0,      0,      0,      0,     91,  59602],
       [     0,      0,      0,      0,     30,  12428],
       [     0,      0,      0,      0,      0,  13727],
       [     0,      0,      0,      0,      0,  39810],
       [     0,      0,      0,      0,    150, 182675],
       [     0,      0,      0,      0,   1005, 799169]], dtype=int64)
```