In [ ]:
```
import re
import spacy
import nltk
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from textblob import TextBlob
import calendar
import contractions
from nltk.corpus import wordnet, stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from sklearn.utils import shuffle
from wordcloud import WordCloud

import warnings
warnings.filterwarnings('ignore')
```

## Read Data

In [ ]:
```
df = pd.read_csv('cook-book-data\Cookbook Reviews.csv')
df.head(2)
```

Out[ ]:

| | rec_no | rec_cd | rec_nm | cmt_id |
|---|---|---|---|---|
| **0** | 1 | 14299 | Creamy White Chili | sp_aUSaElGf_14299_c_2G3aneMRgRMZwXqlHmSdXSG1hEM | u_9iFLIhM |
| **1** | 1 | 14299 | Creamy White Chili | sp_aUSaElGf_14299_c_2FsPC83HtzCsQAtOxlbL6RcaPbY | u_Lu6p25 |

## Preprocessing

In [ ]:
```
datapoint = df[~(df['ratings']==0)].reset_index(drop=True)
renamed_cols = {
    'rec_nm':'recipe',
    'user_reput': 'user_behavior',
    'response_no': 'comment_response'
}
datapoint.rename(columns=renamed_cols, inplace=True)
```

```
datapoint['datetime'] = pd.to_datetime(datapoint['timestamp'], unit='s')
datapoint['year'] =datapoint['datetime'].dt.year

month_dict = {i: calendar.month_name[i][:3] for i in range(1, 13)}
datapoint['month'] = datapoint['datetime'].dt.month#.map(month_dict)

drop_cols = ['rec_cd', 'rec_no', 'cmt_id', 'user_id', 'user_nm', 'timestamp', 'date
datapoint.drop(columns=drop_cols, inplace=True)
datapoint.drop_duplicates(inplace=True)
datapoint.dropna(inplace=True)
datapoint.head()
```

Out[ ]:

| | recipe | user_behavior | comment_response | upvotes | downvotes | ratings | max_rating | |
|---|---|---|---|---|---|---|---|---|
| **0** | Creamy White Chili | 1 | 0 | 0 | 0 | 5 | 527 | |
| **1** | Creamy White Chili | 50 | 0 | 7 | 0 | 5 | 724 | |
| **2** | Creamy White Chili | 10 | 0 | 3 | 0 | 5 | 710 | |
| **3** | Creamy White Chili | 1 | 0 | 3 | 1 | 5 | 518 | |
| **4** | Creamy White Chili | 1 | 0 | 11 | 0 | 5 | 833 | |

Text polarity

In [ ]:

```
def text_polarity(text):
    return TextBlob(text).sentiment.polarity

def text_sentiment(text):
    score = text_polarity(text)
    if score<0:
        return 'negative'
```

```
    elif score==0:
        return 'neutral'
    else:
        return 'positive'


datapoint['comment'] = datapoint['comment'].astype(str)
datapoint['sentiment'] = datapoint['comment'].apply(text_sentiment)
```

Data filtering

In [ ]:
```
def filter_votes(datapoint, threshold=3)-> pd.DataFrame:
    up_voted = shuffle(datapoint[datapoint.upvotes>datapoint.downvotes], random_sta
    eq_voted = shuffle(datapoint[datapoint.upvotes==datapoint.downvotes], random_st
    down_voted = shuffle(datapoint[datapoint.upvotes<datapoint.downvotes], random_s

    return shuffle(pd.concat(
        [down_voted[down_voted.ratings<=threshold],
         up_voted[up_voted.ratings>threshold],
         eq_voted], axis=0), random_state=3)

def filter_sentiments(datapoint, threshold=3) -> pd.DataFrame:
    lt_threshold = datapoint[datapoint.ratings<=threshold]
    gt_threshold = datapoint[datapoint.ratings>threshold]
    pos_sentiment = shuffle(gt_threshold[~(gt_threshold.sentiment=='negative')], ra
    neg_sentiment = shuffle(lt_threshold[~(lt_threshold.sentiment=='positive')], ra

    data_frame = shuffle(pd.concat([neg_sentiment, pos_sentiment], axis=0), random_
    return data_frame

modelpoint = filter_votes(datapoint)
modelpoint = filter_sentiments(modelpoint)
modelpoint = shuffle(modelpoint)
# modelpoint.head()
```

## Collective Data Analysis

In [ ]:
```
def compute_distrited_values(colname):
    ratings = sorted(modelpoint[colname].value_counts().to_dict().items(), key=lamb
    values = [w for _, w in ratings]
    labels = [w for w, _ in ratings]
    tot = sum(values)
    annot_labels =['{} - {:,.2f}%'.format(v, (w/tot)*100) for w,v in zip(values, la
    return values, annot_labels
```

In [ ]:
```
#Upvotes & Downvotes plot
def up_down_vote_plot(ax):
    up_votes = modelpoint.groupby('ratings')['upvotes'].sum()
    down_votes = modelpoint.groupby('ratings')['downvotes'].sum()
    res_comments = modelpoint.groupby('ratings')['comment_response'].sum()
    res_votes = pd.merge(up_votes, down_votes, how='inner', on='ratings').merge(res
    res_votes= res_votes.sort_values('ratings', ascending=False)
    res_votes['downvotes'] = res_votes['downvotes'].map(lambda value: -value)
```

```
    sns.barplot(res_votes, x='ratings', y='downvotes', color='blue', gap=0.4,ax=ax,
    sns.barplot(res_votes, x='ratings', y='upvotes', color='red', gap=0.4, ax=ax, l

    ax.set_ylabel('total votes', fontdict=dict(size=10))
    ax.set_xlabel('ratings', fontdict=dict(size=10))
    ax.set_title('Up/Down votes v. rating score', fontdict=dict(size=10))
```

In [ ]:
```
# Pie charts - distributions
def distribution_chart(ax, colname, title):
    values, labels = compute_distrited_values(colname)

    colors={'ratings':'bright', 'sentiment': 'dark6'}
    sns.set_theme()
    plt.style.use('ggplot')
    ax.pie(x=values,colors=sns.color_palette(colors.get(colname)), counterclock=Fal
        'width':0.5, 'edgecolor':'white', 'linewidth': 2}, textprops=dict(size=12))
    ax.set_title(f'{title} distributions', fontsize='11')
    ax.legend(labels=labels, loc='center left', fontsize='8.5', bbox_to_anchor=(-0.
```

In [ ]:
```
# Recipes v. total ratings
def recipe_rating_count_plot(ax, topn=20):
    recipe_df = modelpoint.groupby(['recipe'], as_index=False)['max_rating'].sum()
    recipe_df = recipe_df.sort_values('max_rating', ascending=False).reset_index(dr

    sns.barplot(recipe_df.iloc[:topn], x='max_rating', y='recipe', hue='recipe', ax
    ax.set_ylabel('Recipe')
    ax.set_xlabel('total ratings')
    ax.set_title('Recipes v. total ratings', fontdict=dict(size=10))
```

In [ ]:
```
# Average ratings v. sentiment
def ratings_sentiment_plot(ax):
    rating_sent = modelpoint.groupby(['recipe', 'sentiment'], as_index=False)['rati
    sns.barplot(rating_sent, y='ratings', x='sentiment', palette='plasma', hue='sen
    ax.set_ylabel('ratings')
    ax.set_xlabel('')
    ax.set_title('Average ratings v. sentiment', fontdict=dict(size=10))
```

In [ ]:
```
# Recipe rating v. month
def ratings_month_plot(ax):
    df4 = modelpoint.groupby(['month'],as_index=False)['ratings'].mean().sort_value
    df4.month = df4.month.map(month_dict)
    sns.lineplot(df4, x='month', y='ratings', ax=ax, errorbar=None, color='red')
    ax.set_xticks(ticks=list(range(df4.month.nunique())))
    ax.set_xticklabels(labels=df4.month.unique(),rotation=-60)
    ax.set_ylabel('rating')
    ax.set_xlabel('month')
    ax.set_title('Recipe rating v. month', fontdict=dict(size=10))
```

In [ ]:
```
# Accumulated ratings v. rating scores by sentiment
def max_ratings_plot(ax):
    sns.stripplot(modelpoint, x='ratings', y='user_behavior',palette='plasma', hue=
    ax.legend(loc='center left', fontsize='8', bbox_to_anchor=(0,0.8))
    ax.set_title('Accumulated ratings v. rating scores by sentiment', fontdict=dict
```

```
    ax.set_ylabel('total')
    ax.set_xlabel('score')
```

In [ ]:

```
plt.figure(figsize=(12,9))
sns.set_style('darkgrid')
nrow = 3
ncol = 3

ax1 = plt.subplot2grid((nrow,ncol),(0,0))
ax2 = plt.subplot2grid((nrow,ncol),(0,1))
ax3 = plt.subplot2grid((nrow,ncol),(0,2))
ax4 = plt.subplot2grid((nrow,ncol),(1,0), rowspan=2)
ax5 = plt.subplot2grid((nrow,ncol),(1,1), colspan=2)
# ax6 & ax7 - Are not added to template; their space is covered by spanning rows/co
ax8 = plt.subplot2grid((nrow,ncol),(2,1), rowspan=1)
ax9 = plt.subplot2grid((nrow,ncol),(2,2))

distribution_chart(ax1, 'ratings', 'rating')
distribution_chart(ax2, 'sentiment', 'sentiment')
up_down_vote_plot(ax3)
recipe_rating_count_plot(ax4,topn=20)
ratings_month_plot(ax5)
ratings_sentiment_plot(ax8)
recipe_rating_count_plot(ax4,topn=20)
max_ratings_plot(ax9)

plt.tight_layout()

plt.show()
```


## Text Level Analysis

### Fields selection

In [ ]:
```
pd.set_option('display.max_colwidth', 200)
model_point= modelpoint[['ratings', 'sentiment','comment']].dropna().copy()
model_point.head()
```

Out[ ]:

| | ratings | sentiment | comment |
|---|---|---|---|
| **10781** | 5 | positive | So Delicious and Tender! |
| **1757** | 5 | positive | I have been making this for years and I love i... |
| **11216** | 4 | positive | These are simple to make and tasty. A family f... |
| **12984** | 5 | positive | I think that white icing is WAAAYY too sugary ... |
| **8436** | 5 | positive | This was very good! I placed this in a 9 x 13... |

### Text-preprocessing

In [ ]:
```
# text-preprocessing methods
stop_words = np.unique(list(ENGLISH_STOP_WORDS) + stopwords.words('english'))
lemma = WordNetLemmatizer()

def get_wordnet_pos(word):
```

```
    tag = nltk.pos_tag([word])[0][1][1].upper()
    tag_list = {
        'J': wordnet.ADJ,
        'N': wordnet.NOUN,
        'V': wordnet.VERB,
        'R': wordnet.ADV
    }
    return tag_list.get(tag, wordnet.NOUN)

nlp_model = spacy.load('en_core_web_sm', disable=['ner', 'parser'])

def text_processor(s: str) -> str:
    text = s.lower()
    text = contractions.fix(text)
    text = ' '.join([word.strip() for word in re.findall(r'\w+',text)])
    text = re.sub(r'\d+', '', text)
    text = ' '.join([word for word in word_tokenize(text) if word not in stop_words
    # text = ' '.join([lemma.lemmatize(word, pos=get_wordnet_pos(word)) for word in
    text = ' '.join([token.lemma_ for token in nlp_model(text)])
    text = ' '.join([word for word in text.split() if len(word)>2])
    return text

model_point['comment'] = model_point['comment'].astype(str)
model_point['clean_comment'] = model_point['comment'].map(text_processor)
model_point['tokens'] = model_point['clean_comment'].map(lambda s: s.split())
model_point.head()
```

Out[ ]:

| | ratings | sentiment | comment | clean_comment | tokens |
|---|---|---|---|---|---|
| **10781** | 5 | positive | So Delicious and Tender! | delicious tender | [delicious, tender] |
| **1757** | 5 | positive | I have been making this for years and I love i... | make year love time substitute sour cream evap... | [make, year, love, time, substitute, sour, cre... |
| **11216** | 4 | positive | These are simple to make and tasty. A family f... | simple make tasty family favorite | [simple, make, tasty, family, favorite] |
| **12984** | 5 | positive | I think that white icing is WAAAYY too sugary ... | think white ice waaayy sugary sweet look decad... | [think, white, ice, waaayy, sugary, sweet, loo... |
| **8436** | 5 | positive | This was very good! I placed this in a 9 x 13... | good place pan need crush oreos crust pan norm... | [good, place, pan, need, crush, oreos, crust, ... |

### Text Visualization

In [ ]:

```
from collections import Counter

total_words = Counter([w for tokens in model_point['tokens'] for w in tokens])

stars_4_to_5 = model_point[(model_point['ratings']==4) | (model_point['ratings']==5
```

```
stars_1_to_3 = model_point[(model_point['ratings']==1) |
                           (model_point['ratings']==3) |
                           (model_point['ratings']==3)]


sorted_4_to_5 = sorted({word: index+1 for tokens in stars_4_to_5['tokens']
                        for index, word in enumerate(tokens)}.items(), key=lambda i
sorted_1_to_3 = sorted({word: index+1 for tokens in stars_1_to_3['tokens']
                        for index, word in enumerate(tokens)}.items(), key=lambda i
```

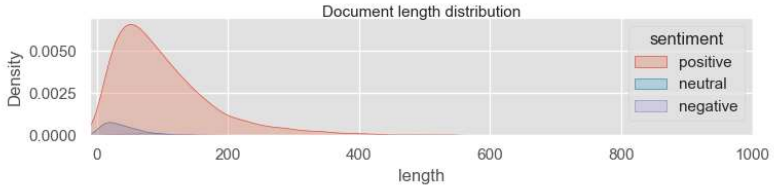
In [ ]:

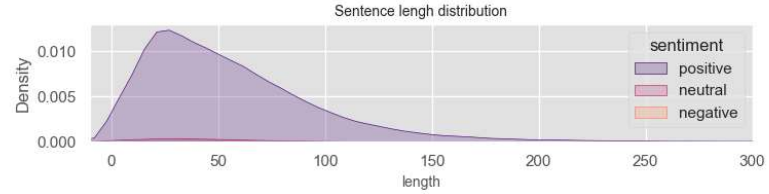
```
# Words Distribution
def doc_length_plot(ax):
    review_length = [(y, len(x)) for x,y in zip(model_point['clean_comment'], model
    doc_size = pd.DataFrame(review_length, columns=['sentiment', 'doc_length'])
    review_length = [(y, len(x)) for x,y in zip(model_point['clean_comment'], model
    doc_size = pd.DataFrame(review_length, columns=['sentiment', 'doc_length'])
    sns.kdeplot(doc_size, x='doc_length', hue='sentiment',ax=ax, fill=True)
    ax.set_title('Document length distribution', fontdict=dict(size=11, weight='lig
    ax.set_xlim(-10, 1000)
    ax.set_xlabel('length')

def sentence_length_plot(ax):
    sentence_level = [(doc[1], sentence) for doc in zip(model_point['comment'], mod
                      for sentence in sent_tokenize(doc[0],'english')]
    sentence_length = [(k, len(s)) for k, s in sentence_level]
    sentence_size = pd.DataFrame(sentence_length, columns=['sentiment', 's_length']

    sns.kdeplot(sentence_size, x='s_length', palette='magma', hue='sentiment', fill
    ax.set_title('Sentence lengh distribution', fontdict=dict(size=10, weight='ligh
    ax.set_xlim(-10, 300)
    ax.set_xlabel('length', fontsize='10')

# Words Frequency v. Sentiments
pos_modelpoint = model_point[model_point['sentiment']=='positive']
ne_modelpoint = model_point[model_point['sentiment']=='neutral']
neg_modelpoint = model_point[model_point['sentiment']=='negative']
sorted_pos = sorted({word: index+1 for tokens in pos_modelpoint['tokens']

                     for index, word in enumerate(tokens)}.items(), key=lambda item

sorted_neg = sorted({word: index+1 for tokens in neg_modelpoint['tokens']
                     for index, word in enumerate(tokens)}.items(), key=lambda item

sorted_ne = sorted({word: index+1 for tokens in ne_modelpoint['tokens']
                    for index, word in enumerate(tokens)}.items(), key=lambda item:

def plot_sentiment_words(ax, words: list[str],title: str, topn=10):
    x = [i for i, _ in words[:topn]]
    y = [i for _,i in words[:topn]]

    colors = {
        'positive': 'flare',
        'negative': 'plasma',
        'neutral': 'hsv'
    }
```

```
    sns.barplot(x=y, y=x,palette=colors.get(title), hue=x, ax=ax)
    ax.set_xlabel('count')
    ax.set_ylabel('words')
    ax.set_title(f'Top {topn} words v. {title}', fontdict=dict(size=11, weight='lig

# Recipe WordCloud
def cloud_plot(ax, sorted_words, title):
    word_cloud = WordCloud(
        stopwords=stop_words,
        background_color='white',
        max_font_size=100,
        max_words=50,
        colormap='plasma',
        height=500,
        random_state=42,
        prefer_horizontal=1,
        min_font_size=10
    )

    word_cloud.generate_from_frequencies(dict(sorted_words))
    ax.imshow(word_cloud)
    title = ax.set_title(f'{title}', fontdict=dict(size=12, weight='light', color='
```

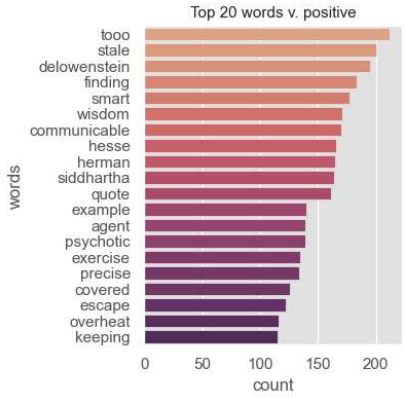
In [ ]:

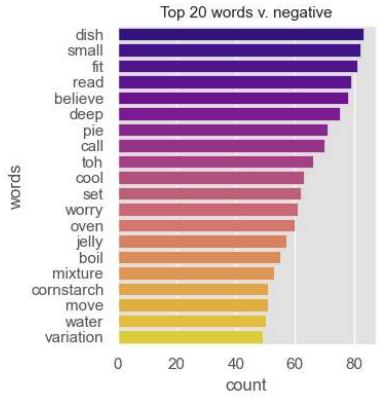
```
nrow = 4
ncol = 3
plt.figure(figsize=(12,8))

axes1 = plt.subplot2grid((nrow, ncol), loc=(0,0), colspan=2)
axes3 = plt.subplot2grid((nrow, ncol), loc=(0,2), rowspan=2)
axes4 = plt.subplot2grid((nrow, ncol), loc=(1,0), colspan=2)
axes7 = plt.subplot2grid((nrow, ncol), loc=(2,0), rowspan=2)
axes8 = plt.subplot2grid((nrow, ncol), loc=(2,1), rowspan=2)
axes9 = plt.subplot2grid((nrow, ncol), loc=(2,2), rowspan=2)

doc_length_plot(axes1)
sentence_length_plot(axes4)

cloud_plot(axes3, total_words, 'Recipe Words')
axes3.axis('off')

plot_sentiment_words(axes7, sorted_pos, 'positive', topn=20)
plot_sentiment_words(axes8, sorted_neg, 'negative', topn=20)
plot_sentiment_words(axes9, sorted_ne, 'neutral', topn=20)

plt.tight_layout()
plt.show()
```

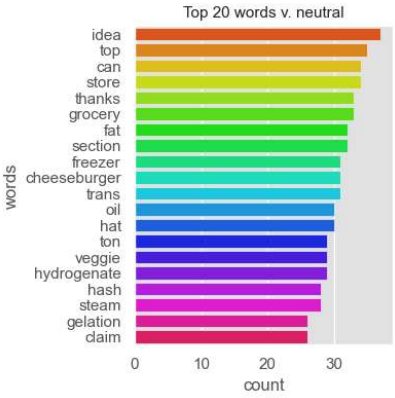

Document length distribution
Density
0.0050
0.0025
0.0000
0
200
400
600
800
1000
length
sentiment
positive
neutral
negative


Sentence lengh distribution
Density
0.010
0.005
0.000
0
50
100
150
200
250
300
length
sentiment
positive
neutral
negative


Recipe Words
recipe
bread
cheese
instead
really
great
easy
make
sugar
soup
love
sauce
like
time
family
cup
delicious
try
cake
add
good
use
taste
flavor
husband


Top 20 words v. positive
words
tooo
stale
delowenstein
finding
smart
wisdom
communicable
hesse
herman
siddhartha
quote
example
agent
psychotic
exercise
precise
covered
escape
overheat
keeping
0
50
100
150
200
count


Top 20 words v. negative
words
dish
small
fit
read
believe
deep
pie
call
toh
cool
set
worry
oven
jelly
boil
mixture
cornstarch
move
water
variation
0
20
40
60
80
count


Top 20 words v. neutral
words
idea
top
can
store
thanks
grocery
fat
section
freezer
cheeseburger
trans
oil
hat
ton
veggie
hydrogenate
hash
steam
gelation
claim
0
10
20
30
count

In [ ]: