UID: 406312710

Econ 425: Machine Learning

Project Title: Flavorful Insights

A customer sentiment analysis project for bakery items from Amazon Fine Foods Reviews 1

Summary:
This project aims to analyze customer sentiment towards bakery products from Amazon Fine Foods reviews and compare the performance of three sentiment analysis models:

- VADER
- roBERTa
- Hugging Face Pipeline (Transformers)

The analysis tries to address the question: **"How can sentiment analysis of customer reviews inform market strategies, product development, and pricing decisions for both large e-commerce platforms like Amazon and small businesses in the food industry?"**

References:

Anvil Works. (n.d.). Using Hugging Face transformers on the Anvil server. Retrieved March 18, 2025, from https://anvil.works/forum/t/using-hugging-face-transformers-on-the-anvil-server/17264

Analytics Vidhya. (2022, October). Sentiment analysis using VADER. Retrieved March 18, 2025, from https://www.analyticsvidhya.com/blog/2022/10/sentiment-analysis-using-vader/

Barbieri, F., Camacho-Collados, J., Espinosa Anke, L., & Neves, L. (2020). TweetEval: Unified benchmark and comparative evaluation for tweet classification. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1644–1650. https://doi.org/10.18653/v1/2020.findings-emnlp.148

GeeksforGeeks. (n.d.). How to create a pie chart in Seaborn? Retrieved March 18, 2025, from https://www.geeksforgeeks.org/how-to-create-a-pie-chart-in-seaborn/

Hugging Face. (n.d.). CardiffNLP/twitter-roberta-base-sentiment. Retrieved March 18, 2025, from https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

Kaggle. (n.d.). Amazon Fine Food Reviews dataset. Retrieved March 18, 2025, from https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

Matplotlib. (n.d.). Matplotlib.pyplot.suptitle. Retrieved March 18, 2025, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.suptitle.html

Mulla, R. (2021). Sentiment analysis Python YouTube tutorial. Kaggle. Retrieved March 18, 2025, from https://www.kaggle.com/code/robikscube/sentiment-analysis-python-youtube-tutorial

PrettyTable. (n.d.). PrettyTable (version 3.15.1) [Python package]. PyPI. Retrieved March 21, 2025, from https://pypi.org/project/prettytable/

Scikit-learn. (n.d.). *TfidfVectorizer*. Retrieved March 18, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Stanford SNAP. (n.d.). Web-FineFoods dataset. Retrieved March 18, 2025, from https://snap.stanford.edu/data/web-FineFoods.html

Stack Overflow. (2018, March 1). How to use tqdm with pandas in a Jupyter notebook? Retrieved March 18, 2025, from https://stackoverflow.com/questions/40476680/how-to-use-tqdm-with-pandas-in-a-jupyter-notebook

Stack Overflow. (2019, April 15). How to add extra stop words in addition to default stopwords in WordCloud? Retrieved March 18, 2025, from https://stackoverflow.com/questions/53997443/how-to-add-extra-stop-words-in-addition-to-default-stopwords-in-wordcloud

Table of Contents:

In [1]:
```
#!pip install wordcloud
#!pip install torch torchvision torchaudio
#!pip install prettytable
```

In [2]:
```
#import necessary libs
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from wordcloud import WordCloud, STOPWORDS
from nltk.tokenize import word_tokenize
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
from tqdm import tqdm
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
#from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
plt.style.use('ggplot')
from prettytable import PrettyTable
from transformers import pipeline
```

In [3]:
```
# Downloading specific NLTKresources
#nltk.download('punkt')  # For tokenization
#nltk.download('averaged_perceptron_tagger')  # For POS tagging
#nltk.download('maxent_ne_chunker')  # For named entity recognition
#nltk.download('words')  # For the word corpus used in chunking
nltk.download('vader_lexicon') # VADER assigns sentiment scores (positive, negative
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\kar_o\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[3]: True

Data

Source

- For text file: https://snap.stanford.edu/data/web-FineFoods.html
- For .csv file: https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

Description:

This dataset comprises approximately 500,000 reviews of fine foods from Amazon from 10/1999 to 10/2012

Features:

- **Id** - Unique identifier for each review.
- **ProductId** - Unique identifier for the product being reviewed.
- **UserId** - Unique identifier for the user who wrote the review.
- **ProfileName** - Display name of the user who submitted the review.
- **HelpfulnessNumerator** - How many users found the review helpful.
- **HelpfulnessDenominator** - Total users who voted on the helpfulness of the review.
- **Score** - Rating given by the user, typically on a scale of 1 to 5.
- **Time** - Timestamp of when the review was submitted.
- **Summary** - Short summary of the review.
- **Text** - Elaborate review provided by the user.

Data Preprocessing

- Loading the Data and assessing the dataset size
- Renaming Columns for better readability
- Handling Missing Values
- Checking for Duplicates
- Filtering for Bakery-Related Products
- Displaying the final dataframe

In [4]:
```
# Upload data
df = pd.read_csv("Reviews.csv")
df.shape
```

Out[4]: (568454, 10)

In [5]:
```
df.columns #checking col names
```

Out[5]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [6]:
```
df = df.rename(columns={"Score":"Rating"}) # this is the primary column for my anal
df.columns
```

Out[6]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Rating', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype 
---  ------                  --------------   ----- 
 0   Id                      568454 non-null  int64 
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568428 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64 
 5   HelpfulnessDenominator  568454 non-null  int64 
 6   Rating                  568454 non-null  int64 
 7   Time                    568454 non-null  int64 
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

In [8]:
```
#Handling missing values
print(df.isna().sum()) #find how many?

na_cols = df.columns[df.isna().any()] #na_cols are ProfileName and Summary

#print(df[na_cols].dtypes) #finding datatype of those cols
#rows_with_na_summary = df[df['ProfileName'].isna()][['ProfileName', 'Summary', "Te
#rows_with_na_summary
```

```
Id                         0
ProductId                  0
UserId                     0
ProfileName               26
HelpfulnessNumerator       0
HelpfulnessDenominator     0
Rating                     0
Time                       0
Summary                   27
Text                       0
dtype: int64
```

In [9]:
```
df['ProfileName']=df['ProfileName'].fillna('Unknown') #filling missing values
df['Summary']= df['Summary'].fillna('No Summary') #filling missing values
df.isna().any() #check again
```

Out[9]:
```
Id                      False
ProductId               False
UserId                  False
ProfileName             False
HelpfulnessNumerator    False
HelpfulnessDenominator  False
Rating                  False
Time                    False
Summary                 False
Text                    False
dtype: bool
```

In [10]:
```
num_duplicates = df.duplicated().sum() #checking for duplicate rows
print(f"Number of duplicate rows: {num_duplicates}")
```

```
Number of duplicate rows: 0
```

In [11]:
```
# Making a list of bakery-related keywords
list_of_baking_words = ["cake", "bread", "cookie", "muffin", "pastry", "croissant",

# Filtering rows where 'Summary' col contains the above words
bakery_df= df[df["Summary"].str.contains('|'.join(list_of_baking_words), case=False

# Results
print(bakery_df.head())
```

```
      Id   ProductId          UserId     ProfileName  HelpfulnessNumerator  \
116  117  B0026Y3YBK  A3P60QLFDDCHOY   Giordano "GB"                     2
117  118  B0026Y3YBK  A38BUM0OXH38VK    singlewinder                     0
183  184  B001KUUNP6  A262Z0S6PT9U16    Lee Thombley                     3
375  376  B0087HW5E2  A139RTDNMU3WY5    blanket lady                     2
394  395  B001ELL6O8  A13T2G4T8LR8XA  First Time Mom                     2

     HelpfulnessDenominator  Rating        Time  \
116                       2       5  1304899200
117                       0       5  1347667200
183                       3       5  1292716800
375                       2       5  1339977600
394                       2       5  1189296000

                                      Summary  \
116                                Great cookies
117                          Best everyday cookie!
183  Perfect for gluten-free chocolate chip cookies
375          Greatest Oil since slice bread !!!!!!!
394                              Delisious Pancakes

                                                  Text
116  I'm Italian and I lived in Italy for years. I ...
117  In the 1980s I spent several summers in Italy....
183  We made chocolate chip cookies with BRM Garban...
375  I have used this oil for several years and it ...
394  Before I discover this mix on Amazon I always ...
```

EDA

- **Vectorization** (this is not a necessary step for this project but can be helpful)
  - Vectorization helps in identifying relevant bakery keywords in the text data, useful for interpreting sentiment scores. By matching these keywords, it can be assessed if the presence of certain terms correlates with positive or negative sentiments, enhancing model interpretability.
- **Statistical summary**
- **Pie Chart**
- **Count Plot**

Vectorization

In [12]:
```
# Vectorizing the Summary column of the bakery-related reviews to find if the keywo
vectorizer = TfidfVectorizer(stop_words="english") #initializing a vectorizer objec
baking_words = vectorizer.fit_transform(bakery_df["Summary"]) #running the vectoriz
baking_features = vectorizer.get_feature_names_out() # Extracting top features from

#Function to find the matching words in the list of baking words and the baking fea
for word in list_of_baking_words :
    if word in baking_features:
        print(f"{word} is present in Summary")
```

```
cake is present in Summary
bread is present in Summary
cookie is present in Summary
muffin is present in Summary
pastry is present in Summary
croissant is present in Summary
bagel is present in Summary
biscuit is present in Summary
donut is present in Summary
brownie is present in Summary
```

Statistical summary

In [13]:
```
bakery_df.describe()
```

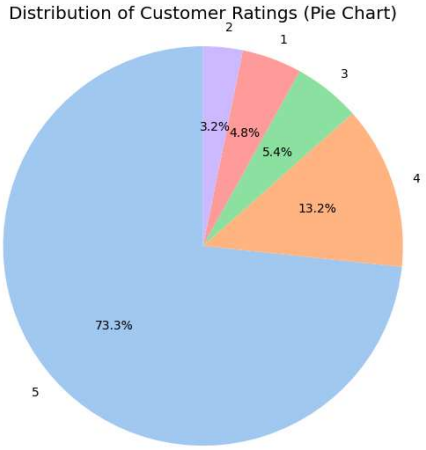
Out[13]:

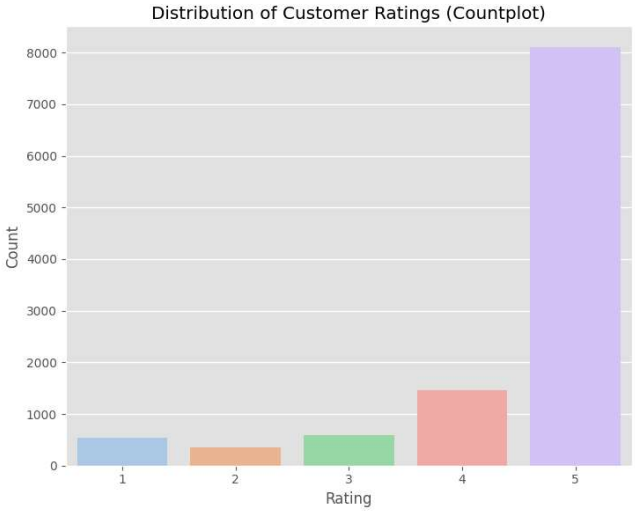
	Id	HelpfulnessNumerator	HelpfulnessDenominator	Rating	
count	11048.000000	11048.000000	11048.000000	11048.000000	1.1048
mean	276918.953295	1.421705	1.747556	4.470221	1.2817
std	165211.115639	3.706097	4.155602	1.062806	5.2400
min	117.000000	0.000000	0.000000	1.000000	9.8280
25%	143946.000000	0.000000	0.000000	4.000000	1.2441
50%	263833.000000	0.000000	0.000000	5.000000	1.2944
75%	420714.750000	2.000000	2.000000	5.000000	1.3256
max	567573.000000	96.000000	96.000000	5.000000	1.3512

Pie Chart and Count Plot

In [14]:
```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

rating_counts = bakery_df['Rating'].value_counts()
axes[0].pie(rating_counts, labels=rating_counts.index, colors=sns.color_palette('pa
axes[0].set_title("Distribution of Customer Ratings (Pie Chart)")
axes[0].axis('equal')

sns.countplot(data=bakery_df, x='Rating', palette='pastel', ax=axes[1])
axes[1].set_title("Distribution of Customer Ratings (Countplot)")
axes[1].set_xlabel("Rating")
axes[1].set_ylabel("Count")

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

```
C:\Users\kar_o\AppData\Local\Temp\ipykernel_7936\3470386022.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=bakery_df, x='Rating', palette='pastel', ax=axes[1])
```





Inference

Mean Rating: The average rating of 4.47 indicates a strong inclination towards positive reviews. The standard deviation of 1.06 indicates a moderate variation in ratings, suggesting that while most products receive high ratings, there are still some products with low ratings (near the minimum of 1).

Pie Chart: About 73.3%, received 5-star ratings, signifying high levels of satisfaction.Lower ratings, particularly 1-star and 2-star reviews, were minimal, reflecting a smaller proportion of dissatisfied customers.

Count Plot: The count plot shows the stark contrast between the number of 5-star ratings and those rated lower. The prevalence of positive reviews suggests a strong product market fit and generally favorable customer experience.

VADER model

The first model implemented is VADER (Valence Aware Dictionary and sEntiment Reasoner).

I'm using NLTK's SentimentIntensityAnalyzer (by creating an object sia) to get the following sentiment scores of text in the summary col:

- negative
- neutral
- positive
- compound

In [15]:
```
# Making a VADER sentiment score function
def apply_vader_sentiment(df, text_column='Summary'):
    # Sentiment Analysis object
    sia = SentimentIntensityAnalyzer()
```

```
    # Run sentiment analysis to get sentiment scores for each row
    sentiment_df = df.copy()
    sentiment_df[['vader_neg', 'vader_neu', 'vader_pos', 'vader_compound']] = senti
        lambda x: pd.Series(sia.polarity_scores(str(x)))
    )

    return sentiment_df
```

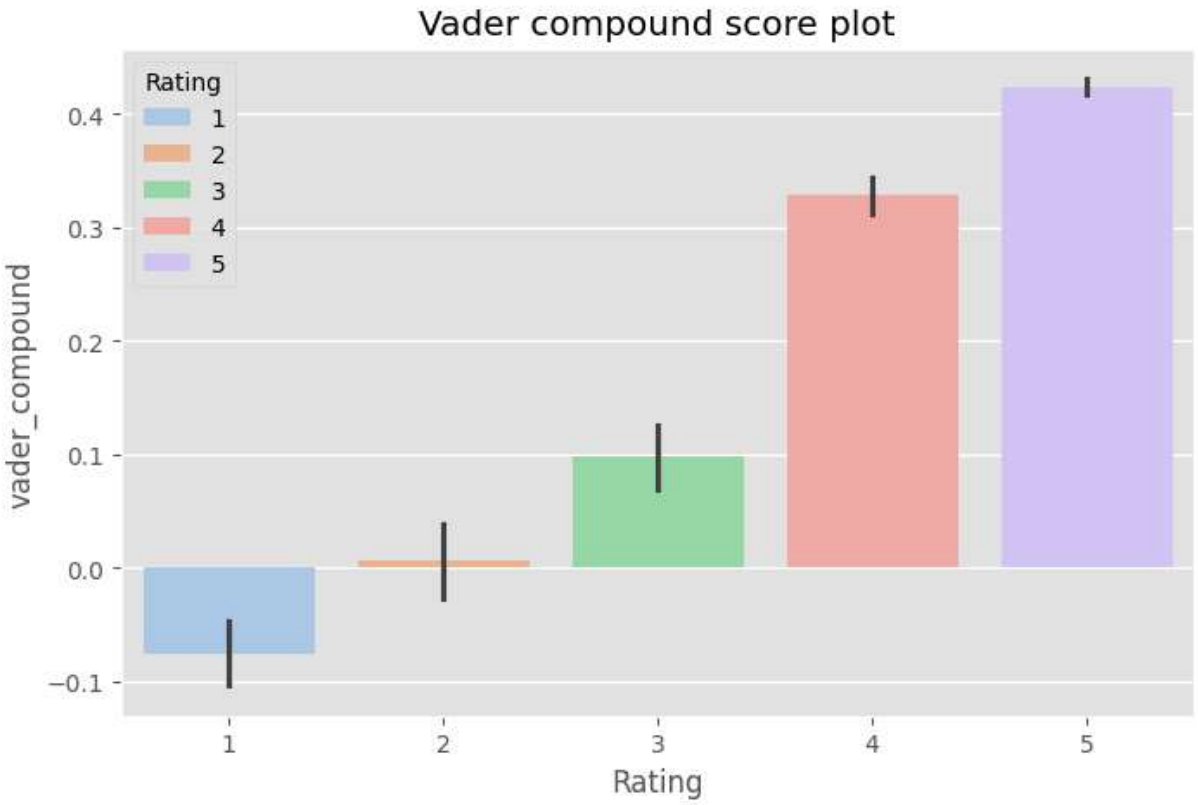
In [16]:
```
# Applying the function on Summary col
sentiment_df = apply_vader_sentiment(bakery_df, text_column='Summary')
print(sentiment_df.head(3))
```

```
     Id   ProductId          UserId    ProfileName  HelpfulnessNumerator  \
116  117  B0026Y3YBK  A3P60QLFDDCHOY  Giordano "GB"                     2
117  118  B0026Y3YBK  A38BUM0OXH38VK   singlewinder                     0
183  184  B001KUUNP6  A262Z0S6PT9U16   Lee Thombley                     3

     HelpfulnessDenominator  Rating        Time  \
116                       2       5  1304899200
117                       0       5  1347667200
183                       3       5  1292716800

                                      Summary  \
116                               Great cookies
117                         Best everyday cookie!
183  Perfect for gluten-free chocolate chip cookies

                                                  Text  vader_neg  vader_neu  \
116  I'm Italian and I lived in Italy for years. I ...        0.0      0.196
117  In the 1980s I spent several summers in Italy....        0.0      0.308
183  We made chocolate chip cookies with BRM Garban...        0.0      0.575

     vader_pos  vader_compound
116      0.804          0.6249
117      0.692          0.6696
183      0.425          0.5719
```

Plot of compound score for the Vader Model

The plot of compund rating shows the relationship between `Rating` and the `vader_compound` sentiment score columns for the sentiment analysis.

In [17]:
```
plt.figure(figsize = (8, 5))
ax = sns.barplot(data=sentiment_df, x='Rating', y='vader_compound', hue='Rating', p
ax.set_title('Vader compound score plot')
plt.show()
```



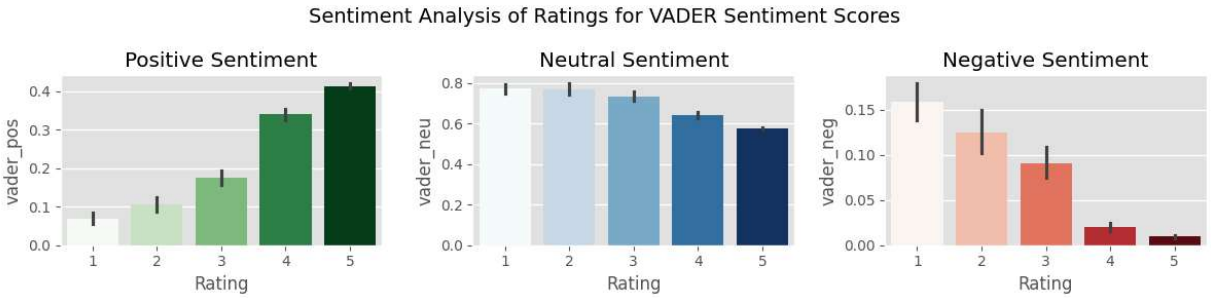
Inference:

- **Positive Sentiment**: Higher ratings (4 and 5 star ratings) have higher `vader_compound` scores -> customer feedback has more positive sentiment.
- **Negative Sentiment**: Lower ratings (1, 2 and 3 stars) have lower `vader_compound` scores -> customer feedback has neutral or more negative sentiment.
- **Trend**: The scores gradually increase with ratings -> higher ratings correlate with more positive sentiment.
- **Error Bars**: The black vertical lines on top of each bar represent the confidence intervals or standard errors -> variability in estimation.

Plot of Sentiment Analysis of Ratings for VADER Sentiment Scores

In [18]:
```
positive = 'Greens'  # For positive sentiment
neutral = 'Blues'  # For neutral sentiment
negative = 'Reds'   # For negative sentiment

fig, axs = plt.subplots(1, 3, figsize=(12, 3))

# Making bar plots for each sentiment
sns.barplot(data=sentiment_df, x='Rating', y='vader_pos', ax=axs[0], hue='Rating',
sns.barplot(data=sentiment_df, x='Rating', y='vader_neu', ax=axs[1], hue='Rating',
sns.barplot(data=sentiment_df, x='Rating', y='vader_neg', ax=axs[2], hue='Rating',

# Titles
axs[0].set_title('Positive Sentiment')
```

```
axs[1].set_title('Neutral Sentiment')
axs[2].set_title('Negative Sentiment')
fig.suptitle('Sentiment Analysis of Ratings for VADER Sentiment Scores', fontsize=1

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



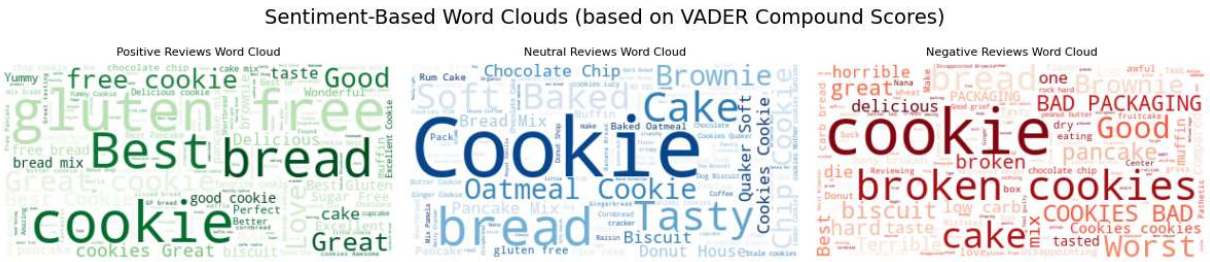
Inference:

- **Positive Sentiment:** Higher ratings (4 and 5) show higher positive sentiment scores indicating customer satisfaction. Darker shades of green indicate more positive sentiment and lighter shades of green indicate lesser postive sentiment. As customer ratings increase, the average positive sentiment score also increases. This indicates that higher-rated reviews tend to have stronger positive sentiment. The gradual shift from light green to dark green further emphasizes this trend. The error bars are relatively small, suggesting that the average sentiment scores are consistent within each rating group.
- **Neutral Sentiment:** Mid-range ratings (3) are neither very positive nor very negative. So they indicate highest neutral sentiment. Darker shades of blue indicate more neutral sentiment and lighter shades of blue indicate lesser neutral sentiment.
- **Negative Sentiment:** Lower ratings (1 and 2) show higher negative sentiment scores, indicating customer dissatisfaction. Darker shades of red indicate more negative sentiment and lighter shades of red indicate lesser negative sentiment.As customer ratings increase, the average negative sentiment score slightly decreases. This indicates that higher-rated reviews tend to have slightly less negative sentiment. The shift from light pink to dark red is subtle. The error bars are again relatively small.

Word Cloud for Vader Model

In [19]:
```
# Based on compund score threshold, categorizing sentiment of the summary col
pos = sentiment_df[sentiment_df['vader_compound'] > 0.05]['Summary']
neg = sentiment_df[sentiment_df['vader_compound'] < -0.05]['Summary']
neu= sentiment_df[(sentiment_df['vader_compound'] >= -0.05) & (sentiment_df['vader_

# Making a single text string by combining summaries of same sentiment
pos_text = " ".join(pos)
neg_text = " ".join(neg)
neu_text = " ".join(neu)

stopwords = set(STOPWORDS) #for pre-defined list of irrelevant words for analysis l
```

```
# Making the word clouds for each sentiment
positive_wordcloud = WordCloud(stopwords = stopwords, background_color='white', col
neutral_wordcloud = WordCloud(stopwords = stopwords, background_color='white', colo
negative_wordcloud = WordCloud(stopwords = stopwords, background_color='white', col

# Plot
fig, ax = plt.subplots(1, 3, figsize = (12,12))


ax[0].imshow(positive_wordcloud, interpolation='bilinear')
ax[0].set_title("Positive Reviews Word Cloud", fontsize=8)
ax[0].axis("off")

ax[1].imshow(neutral_wordcloud, interpolation='bilinear')
ax[1].set_title("Neutral Reviews Word Cloud", fontsize=8)
ax[1].axis("off")

ax[2].imshow(negative_wordcloud, interpolation='bilinear')
ax[2].set_title("Negative Reviews Word Cloud", fontsize=8)
ax[2].axis("off")


fig.subplots_adjust(wspace=0, hspace= 0.1)
fig.suptitle('Sentiment-Based Word Clouds (based on VADER Compound Scores)', fontsi
fig.tight_layout()
plt.show()
```



Inference:

- **Positive Sentiment Word Cloud**: Prominent words in this cloud show positive reception towards taste and dietary preferences.
- **Neutral Sentiment Word Cloud**: Prominent words in this cloud show lack strong emotion showing neutral sentiment. This cloud mainly shows description of the variety of products.
- **Negative Sentiment Word Cloud**: Prominent words in this cloud show negative reception and dissatisfaction.

roBERTa Model

The second model implemented is roBERTa (Robustly optimized BERT approach).

I'm using Hugging Face's transformers library to load a pre-trained roBERTa model to get the following sentiment scores of text in the summary col:

- negative
- neutral
- positive
- compound (calculated manually)

In [20]:
```
task='sentiment'
MODEL = f"cardiffnlp/twitter-roberta-base-{task}"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

In [21]:
```
# Making a RoBERTa sentiment score function
def apply_roBERTa_sentiment(text_col):
    text = tokenizer(text_col, return_tensors='pt')
    output = model(**text)
    score = output[0][0].detach().numpy()
    score = softmax(score)
    dictionary = {
        'roberta_neg': score[0],
        'roberta_neu': score[1],
        'roberta_pos': score[2]
    }
    return dictionary
```

In [22]:
```
# Applying the function on Summary col
tqdm.pandas()  # Enabling a progress bar
sentiment_df['roberta_sentiment_scores'] = sentiment_df['Summary'].progress_apply(a

# Drop unnecessary cols and separate the RoBERTa sentiment scores into three other
sentiment_df = pd.concat([sentiment_df.drop(columns=['roberta_sentiment_scores']),
                          sentiment_df['roberta_sentiment_scores'].apply(pd.Series)
sentiment_df.columns
```

```
100%|██████████████████████████████████████████████████████████████████████████| 1
1048/11048 [07:42<00:00, 23.91it/s]
```

Out[22]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Rating', 'Time', 'Summary', 'Text',
       'vader_neg', 'vader_neu', 'vader_pos', 'vader_compound', 'roberta_neg',
       'roberta_neu', 'roberta_pos'],
      dtype='object')
```

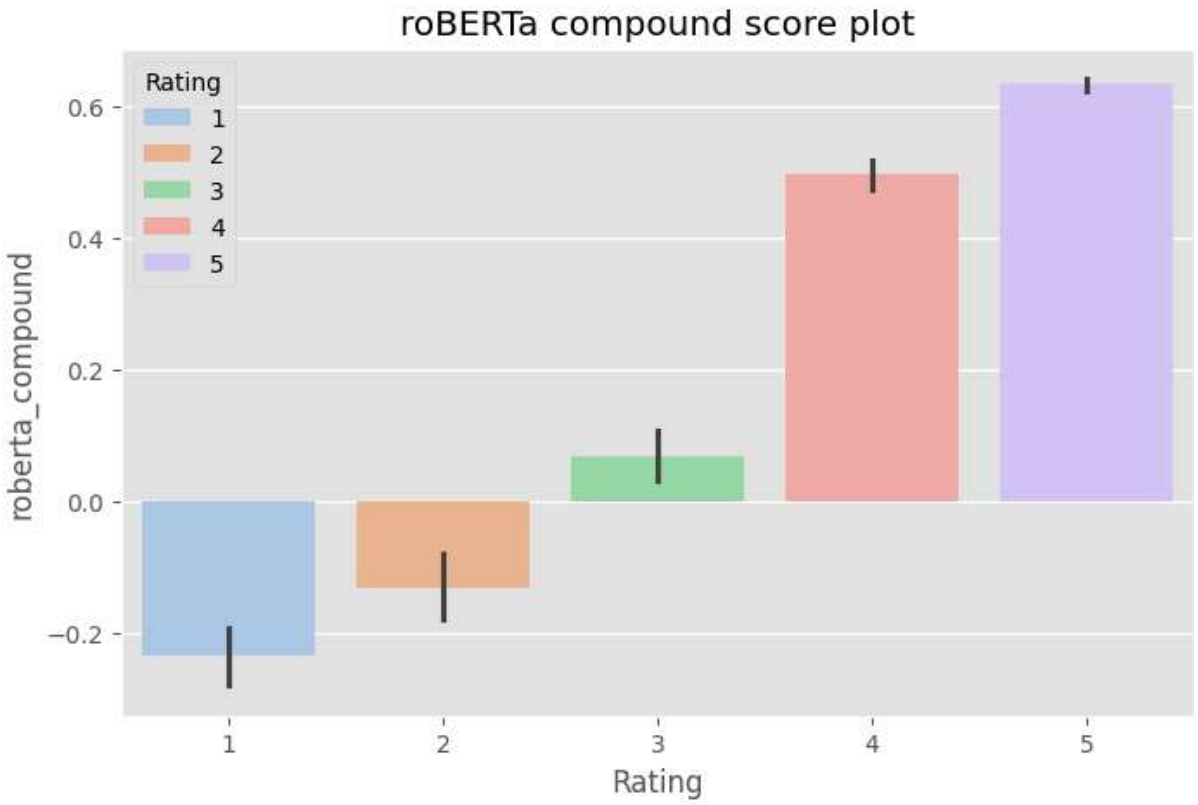
In [23]:
```
# RoBERTa Compound Score
sentiment_df['roberta_compound'] = (sentiment_df['roberta_pos'] - sentiment_df['rob

# Display the first few rows
print(sentiment_df.head())
```

```
      Id   ProductId          UserId     ProfileName  HelpfulnessNumerator  \
116  117  B0026Y3YBK  A3P60QLFDDCHOY   Giordano "GB"                     2
117  118  B0026Y3YBK  A38BUM0OXH38VK    singlewinder                     0
183  184  B001KUUNP6  A262Z0S6PT9U16    Lee Thombley                     3
375  376  B0087HW5E2  A139RTDNMU3WY5    blanket lady                     2
394  395  B001ELL6O8  A13T2G4T8LR8XA  First Time Mom                     2

     HelpfulnessDenominator  Rating        Time  \
116                       2       5  1304899200
117                       0       5  1347667200
183                       3       5  1292716800
375                       2       5  1339977600
394                       2       5  1189296000

                                      Summary  \
116                                Great cookies
117                            Best everyday cookie!
183  Perfect for gluten-free chocolate chip cookies
375          Greatest Oil since slice bread !!!!!!!
394                             Delisious Pancakes

                                                  Text  vader_neg  vader_neu  \
116  I'm Italian and I lived in Italy for years. I ...        0.0      0.196
117  In the 1980s I spent several summers in Italy....        0.0      0.308
183  We made chocolate chip cookies with BRM Garban...        0.0      0.575
375  I have used this oil for several years and it ...        0.0      0.482
394  Before I discover this mix on Amazon I always ...        0.0      1.000

     vader_pos  vader_compound  roberta_neg  roberta_neu  roberta_pos  \
116      0.804          0.6249     0.004493     0.070814     0.924693
117      0.692          0.6696     0.001362     0.034527     0.964110
183      0.425          0.5719     0.001589     0.059519     0.938892
375      0.518          0.7482     0.013173     0.099402     0.887424
394      0.000          0.0000     0.004090     0.254064     0.741847

     roberta_compound
116          0.920200
117          0.962748
183          0.937303
375          0.874251
394          0.737757
```

Plot of compound score for the roBERTa Model

The plot of compund rating shows the relationship between `Rating` and the `roberta_compound` sentiment score columns for the sentiment analysis.

In [24]:
```
plt.figure(figsize = (8, 5))
ax = sns.barplot(data=sentiment_df, x='Rating', y='roberta_compound', hue='Rating',
ax.set_title('roBERTa compound score plot')
plt.show()
```



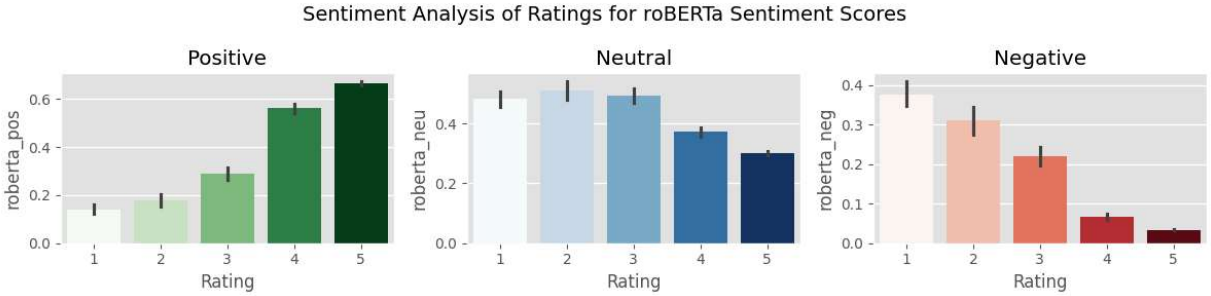
Inference:

- **Positive Sentiment**: Higher ratings (4 and 5 star ratings) have higher `roberta_compound` scores -> customer feedback has more positive sentiment.
- **Negative Sentiment**: Lower ratings (1, 2 and 3 stars) have lower `roberta_compound` scores -> customer feedback has neutral or more negative sentiment.
- **Trend**: The scores graudlly increase with ratings -> higher ratings correlate with more positive sentiment.
- **Error Bars**: The black vertical lines on top of each bar represent the confidence intervals or standard errors -> variability in estimation.

Plot of Sentiment Analysis of Ratings for roBERTa Sentiment Scores

In [25]:
```
positive = 'Greens'  # For positive sentiment
neutral = 'Blues'  # For neutral sentiment
negative = 'Reds'   # For negative sentiment

fig, axs = plt.subplots(1, 3, figsize=(12, 3))

# Making bar plots for each sentiment
sns.barplot(data=sentiment_df, x='Rating', y='roberta_pos', ax=axs[0], hue='Rating'
sns.barplot(data=sentiment_df, x='Rating', y='roberta_neu', ax=axs[1], hue='Rating'
sns.barplot(data=sentiment_df, x='Rating', y='roberta_neg', ax=axs[2], hue='Rating'

# Set titles for each subplot
axs[0].set_title('Positive')
```

```
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
fig.suptitle('Sentiment Analysis of Ratings for roBERTa Sentiment Scores', fontsize

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



Inference:

- **Positive Sentiment:** Higher ratings (4 and 5) show higher positive sentiment scores indicating customer satisfaction. Darker shades of green indicate more positive sentiment and lighter shades of green indicate lesser postive sentiment. As customer ratings increase, the average positive sentiment score also increases. This indicates that higher-rated reviews tend to have stronger positive sentiment. The gradual shift from light green to dark green further emphasizes this trend. The error bars are relatively small, suggesting that the average sentiment scores are consistent within each rating group.
- **Neutral Sentiment:** Mid-range ratings (3) are neither very positive nor very negative. So they indicate highest neutral sentiment. Darker shades of blue indicate more neutral sentiment and lighter shades of blue indicate lesser neutral sentiment.
- **Negative Sentiment:** Lower ratings (1 and 2) show higher negative sentiment scores, indicating customer dissatisfaction. Darker shades of red indicate more negative sentiment and lighter shades of red indicate lesser negative sentiment.As customer ratings increase, the average negative sentiment score slightly decreases. This indicates that higher-rated reviews tend to have slightly less negative sentiment. The shift from light pink to dark red is subtle. The error bars are again relatively small.

Word Cloud for roBERTa Model

In [26]:
```
# Based on compund score threshold, categorizing sentiment of the summary col
pos = sentiment_df[sentiment_df['roberta_compound'] > 0.05]['Summary']
neg = sentiment_df[sentiment_df['roberta_compound'] < -0.05]['Summary']
neu= sentiment_df[(sentiment_df['roberta_compound'] >= -0.05) & (sentiment_df['vade

# Making a single text string by combining summaries of same sentiment
pos_text = " ".join(pos)
neg_text = " ".join(neg)
neu_text = " ".join(neu)

stopwords = set(STOPWORDS)
```

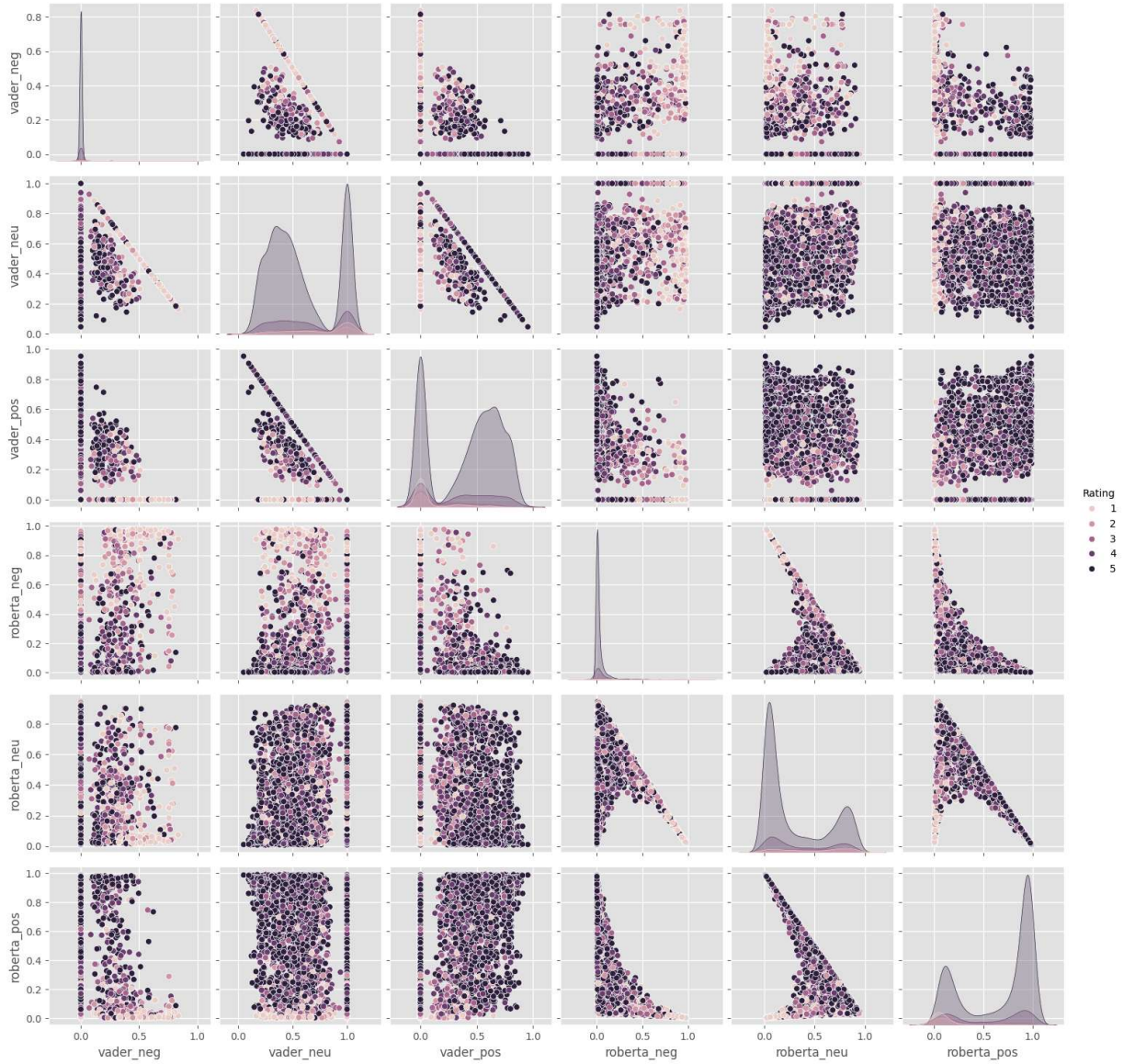
```
# Making the word clouds for each sentiment
positive_wordcloud = WordCloud(stopwords = stopwords, background_color='white', col
neutral_wordcloud = WordCloud(stopwords = stopwords, background_color='white', colo
negative_wordcloud = WordCloud(stopwords = stopwords, background_color='white', col


# Plot
fig, ax = plt.subplots(1, 3, figsize = (12,12))


ax[0].imshow(positive_wordcloud, interpolation='bilinear')
ax[0].set_title("Positive Reviews Word Cloud", fontsize=8)
ax[0].axis("off")

ax[1].imshow(neutral_wordcloud, interpolation='bilinear')
ax[1].set_title("Neutral Reviews Word Cloud", fontsize=8)
ax[1].axis("off")

ax[2].imshow(negative_wordcloud, interpolation='bilinear')
ax[2].set_title("Negative Reviews Word Cloud", fontsize=8)
ax[2].axis("off")


fig.subplots_adjust(wspace=0, hspace= 0.1)
fig.suptitle('Sentiment-Based Word Clouds (based on roBERTa Compound Scores)', font
fig.tight_layout()
plt.show()
```

Inference:

- **Positive Sentiment Word Cloud**: Prominent words in this cloud show positive reception towards taste and dietary preferences.
- **Neutral Sentiment Word Cloud**: Prominent words in this cloud show lack strong emotion showing neutral sentiment. This cloud mainly shows description of the variety of products.
- **Negative Sentiment Word Cloud**: Prominent words in this cloud show negative reception and dissatisfaction.

Pair plot for comparison of VADER and roBERTa models

In [27]:
```
sns.pairplot(sentiment_df, hue='Rating', vars=['vader_neg', 'vader_neu', 'vader_pos
plt.savefig('myimage.png', format='png', dpi= 300)
plt.show()
```



Inference: VADER vs. RoBERTa

- RoBERTa's positive and negative scores are more distinctly separated between different rating groups in contrast to VADER.
- So, RoBERTa is slightly better at judging rating based sentiment.
- Irrespective of rating, Vader scores large number of reviews as neutral.
- Roberta detects negative sentiment stronger than vader.

Hugging Face Pipeline (Transformers) Model

The third model implemented is the Hugging Face Pipeline using pre-trained transformer models (such as RoBERTa or BERT) for sentiment analysis.

I'm using Hugging Face `pipeline` for sentiment analysis to get the following based on the text in the summary col:

- sentiment label (POSITIVE/NEGATIVE)

- sentiment score

In [28]:
```
# Making a Huggingface pipeline (Transformers) sentiment score function
def apply_transformers_sentiment(df, text_column):
    # Sentiment Analysis object
    sent_pipeline = pipeline("sentiment-analysis")

    # Run sentiment analysis to get sentiment scores for each row (with a progress
    df['sentiment'] = [sent_pipeline(x)[0]['label'] for x in tqdm(df[text_column],
    df['sentiment_score'] = [sent_pipeline(x)[0]['score'] for x in tqdm(df[text_col

    return df
```

In [29]:
```
# Applying the function on Summary col
sentiment_df = apply_transformers_sentiment(sentiment_df, 'Summary')
print(sentiment_df.head())
```

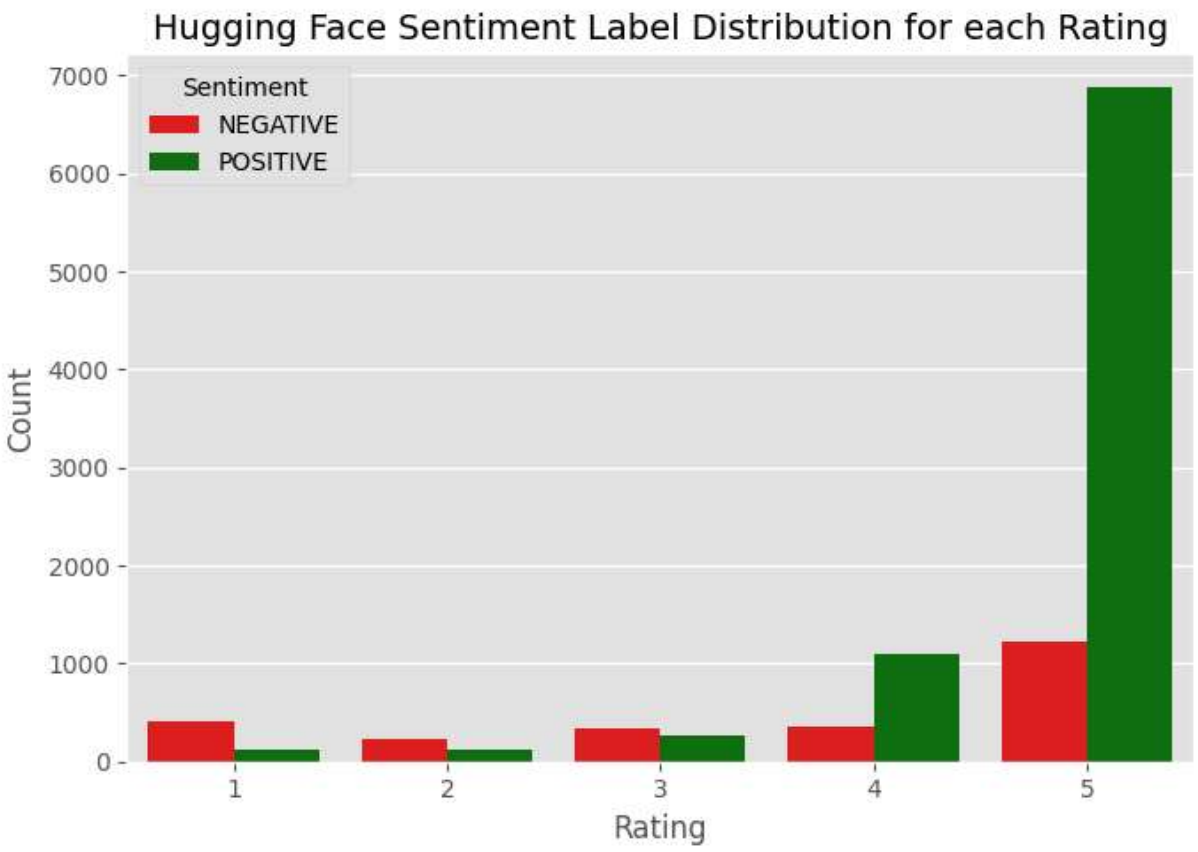
```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst
-2-english and revision 714eb0f (https://huggingface.co/distilbert/distilbert-base-u
ncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not r
ecommended.
Device set to use cpu
Sentiment Analysis: 100%|████████████████████████████████████████████████████████| 11
048/11048 [03:31<00:00, 52.33row/s]
Sentiment Score: 100%|███████████████████████████████████████████████████████████| 11
048/11048 [03:47<00:00, 48.59row/s]
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Rating	Time	Summary	Text	vader_neg	vader_neu	vader_pos	vader_compound	roberta_neg	roberta_neu	roberta_pos	roberta_compound	sentiment	sentiment_score
116	117	B0026Y3YBK	A3P60QLFDDCHOY	Giordano "GB"	2	2	5	1304899200	Great cookies	I'm Italian and I lived in Italy for years. I ...	0.0	0.196	0.804	0.6249	0.004493	0.070814	0.924693	0.920200	POSITIVE	0.999866
117	118	B0026Y3YBK	A38BUM0OXH38VK	singlewinder	0	0	5	1347667200	Best everyday cookie!	In the 1980s I spent several summers in Italy....	0.0	0.308	0.692	0.6696	0.001362	0.034527	0.964110	0.962748	POSITIVE	0.999811
183	184	B001KUUNP6	A262Z0S6PT9U16	Lee Thombley	3	3	5	1292716800	Perfect for gluten-free chocolate chip cookies	We made chocolate chip cookies with BRM Garban...	0.0	0.575	0.425	0.5719	0.001589	0.059519	0.938892	0.937303	POSITIVE	0.994876
375	376	B0087HW5E2	A139RTDNMU3WY5	blanket lady	2	2	5	1339977600	Greatest Oil since slice bread !!!!!!!	I have used this oil for several years and it ...	0.0	0.482	0.518	0.7482	0.013173	0.099402	0.887424	0.874251	POSITIVE	0.999745
394	395	B001ELL6O8	A13T2G4T8LR8XA	First Time Mom	2	2	5	1189296000	Delisious Pancakes	Before I discover this mix on Amazon I always ...	0.0	1.000	0.000	0.0000	0.004090	0.254064	0.741847	0.737757	NEGATIVE	0.994825

Plot of sentiment distribution across ratings for the Huggingface Pipeline (Transformers) Model

In [30]:
```
plt.figure(figsize=(7, 5))

# Making count plots for each sentiment
sns.countplot(data=sentiment_df, x='Rating', hue='sentiment', palette={'POSITIVE':

# Set titles
plt.title('Hugging Face Sentiment Label Distribution for each Rating', fontsize=14)
plt.xlabel('Rating')
plt.ylabel('Count')
plt.legend(title='Sentiment')
```

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



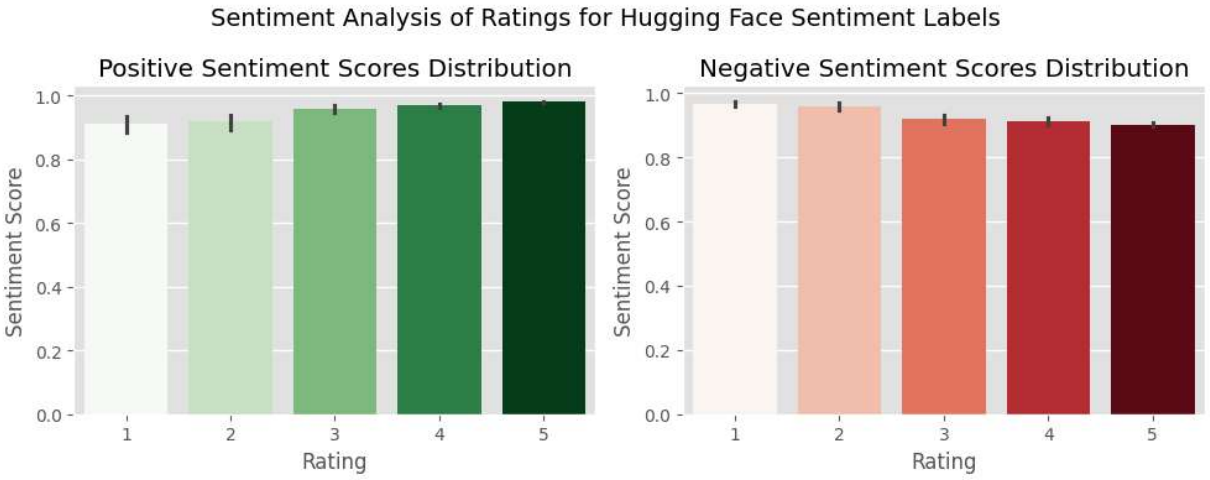
Inference:

With an increase in ratings, there is a an increase in the proportion of positive sentiment. This shows that the model effectively captures a strong positive reception from the customers as seen from the massive amount of 5 stars ratings.

Plot of Sentiment Analysis of Ratings for Huggingface Pipeline (Transformers) Sentiment

In [31]:
```
positive = 'Greens'  # For positive sentiment
negative = 'Reds'   # For negative sentiment


fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Making bar plots for each sentiment
sns.barplot(data=sentiment_df[sentiment_df['sentiment'] == 'POSITIVE'], x='Rating',
sns.barplot(data=sentiment_df[sentiment_df['sentiment'] == 'NEGATIVE'], x='Rating',


# Set titles for each subplot
axs[0].set_title('Positive Sentiment Scores Distribution')
axs[0].set_ylabel('Sentiment Score')
```

```
axs[1].set_title('Negative Sentiment Scores Distribution')
axs[1].set_ylabel('Sentiment Score')

fig.suptitle('Sentiment Analysis of Ratings for Hugging Face Sentiment Labels', fon

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



Inference:

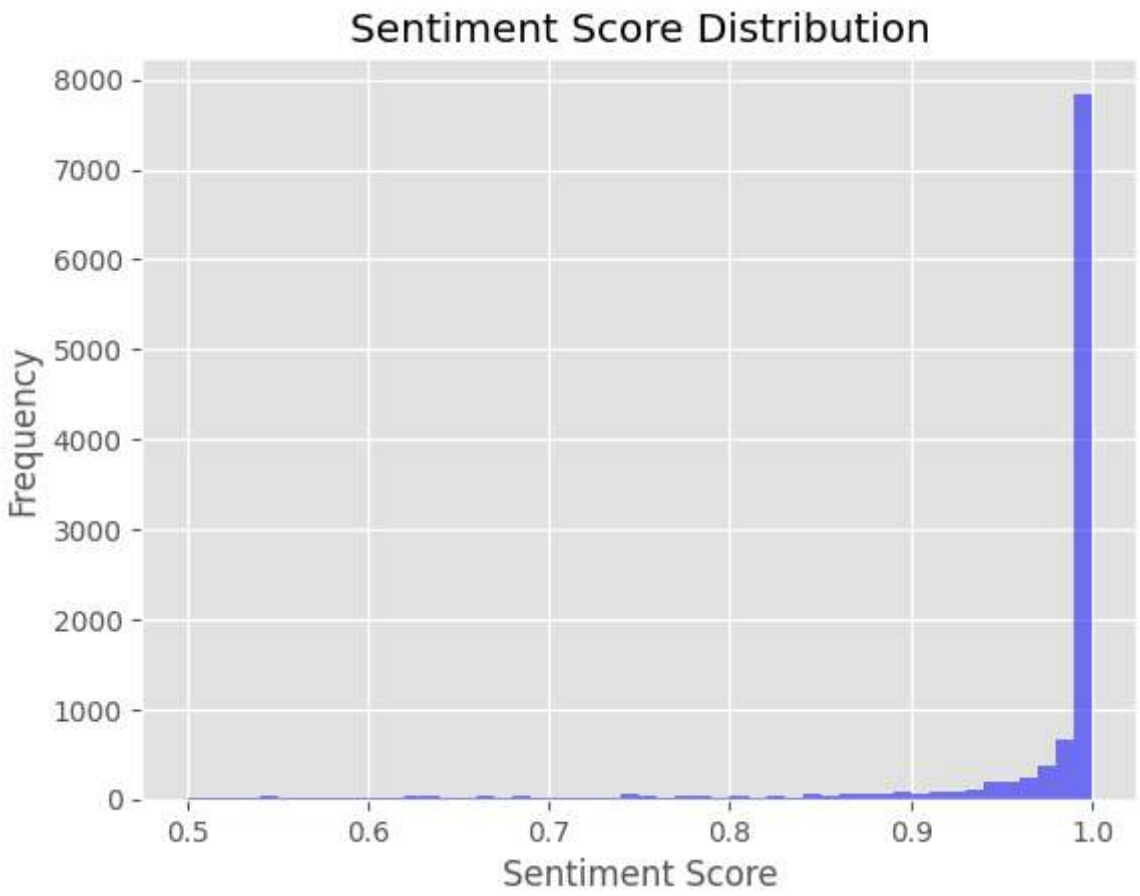
- **Positive Sentiment:** Higher ratings (4 and 5) show higher positive sentiment scores indicating customer satisfaction. Darker shades of green indicate more positive sentiment and lighter shades of green indicate lesser postive sentiment. As customer ratings increase, the average positive sentiment score also increases. This indicates that higher-rated reviews tend to have stronger positive sentiment. The gradual shift from light green to dark green further emphasizes this trend. The error bars are relatively small, suggesting that the average sentiment scores are consistent within each rating group.
- **Negative Sentiment:** Lower ratings (1 and 2) show higher negative sentiment scores, indicating customer dissatisfaction. Darker shades of red indicate more negative sentiment and lighter shades of red indicate lesser negative sentiment. As customer ratings increase, the average negative sentiment score slightly decreases. This indicates that higher-rated reviews tend to have slightly less negative sentiment. The shift from light pink to dark red is subtle. The error bars are again relatively small.

In [32]:
```
# Checking number of positive and negative summaries based on 'sentiment' column
positive_sentiments = sentiment_df[sentiment_df['sentiment'] == 'POSITIVE']
negative_sentiments = sentiment_df[sentiment_df['sentiment'] == 'NEGATIVE']

print(f"Positive Sentiments Count: {len(positive_sentiments)}")
print(f"Negative Sentiments Count: {len(negative_sentiments)}")

# Check sentiment score distribution for the 'sentiment' column
sentiment_df['sentiment_score'].describe()
```

```
Positive Sentiments Count: 8467
Negative Sentiments Count: 2581
```

Out[32]:
```
count    11048.000000
mean         0.964097
std          0.089119
min          0.500198
25%          0.984166
50%          0.998952
75%          0.999817
max          0.999892
Name: sentiment_score, dtype: float64
```

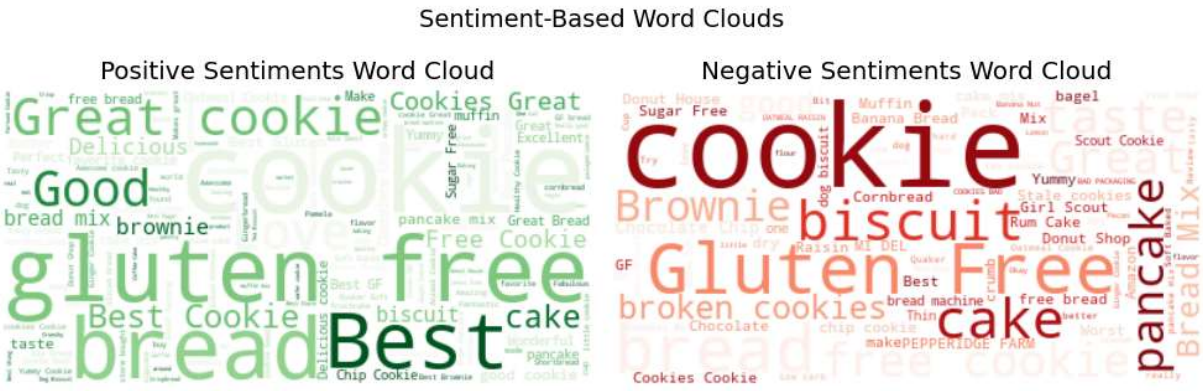
In [33]:
```
plt.hist(sentiment_df['sentiment_score'], bins=50, color='blue', alpha=0.5)
plt.title("Sentiment Score Distribution")
plt.xlabel("Sentiment Score")
plt.ylabel("Frequency")
plt.show()
```



Word Cloud for Hugging Face Pipeline Model

In [34]:
```
# Grouping the data by sentiment label
pos_reviews = sentiment_df[sentiment_df['sentiment'] == 'POSITIVE']['Summary']
neg_reviews = sentiment_df[sentiment_df['sentiment'] == 'NEGATIVE']['Summary']

# Join summaries for each sentiment
pos_text = " ".join(pos_reviews)
neg_text = " ".join(neg_reviews)

# Creating the word clouds
```

```
stopwords = set(STOPWORDS)
positive_wordcloud = WordCloud(stopwords=stopwords, background_color='white', color
negative_wordcloud = WordCloud(stopwords=stopwords, background_color='white', color

# Plot
fig, ax = plt.subplots(1, 2, figsize = (10,8))

plt.subplot(1, 2, 1)
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.title("Positive Sentiments Word Cloud")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(negative_wordcloud, interpolation='bilinear')
plt.title("Negative Sentiments Word Cloud")
plt.axis("off")

fig.suptitle('Sentiment-Based Word Clouds', fontsize=14, y=0.71)


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



Inference:

- **Positive Sentiment Word Cloud**: Prominent words in this cloud show positive reception towards taste and dietary preferences.
- **Negative Sentiment Word Cloud**: Prominent words in this cloud show negative reception and dissatisfaction.

Manually assigning Actual Sentiment Scores

To evaluate the model I will manually assign sentiment scores (true sentiment) to the ratings given and based on that later evaluate if the models have predicted the sentiment correctly. If rating is more than 3 then it is a positive sentiment (1) else negative (0). In simpler words, we check to see if the model can classify a particular rating as positive/neutral/negative as appropirately as a human.

In [35]:
```
# if ratings above 3 are positive sentiment and 3 or below are negative sentiment
sentiment_df['TS'] = sentiment_df['Rating'].apply(lambda x: 1 if x > 3 else 0)
print(sentiment_df.head(3))
```

```
     Id   ProductId          UserId    ProfileName  HelpfulnessNumerator  \
116  117  B0026Y3YBK  A3P60QLFDDCHOY  Giordano "GB"                     2
117  118  B0026Y3YBK  A38BUM0OXH38VK   singlewinder                     0
183  184  B001KUUNP6  A262Z0S6PT9U16   Lee Thombley                     3

     HelpfulnessDenominator  Rating        Time  \
116                       2       5  1304899200
117                       0       5  1347667200
183                       3       5  1292716800

                                       Summary  \
116                                Great cookies
117                          Best everyday cookie!
183  Perfect for gluten-free chocolate chip cookies

                                                  Text  ...  vader_neu  \
116  I'm Italian and I lived in Italy for years. I ...  ...      0.196
117  In the 1980s I spent several summers in Italy....  ...      0.308
183  We made chocolate chip cookies with BRM Garban...  ...      0.575

     vader_pos  vader_compound  roberta_neg  roberta_neu  roberta_pos  \
116      0.804          0.6249     0.004493     0.070814     0.924693
117      0.692          0.6696     0.001362     0.034527     0.964110
183      0.425          0.5719     0.001589     0.059519     0.938892

     roberta_compound  sentiment sentiment_score  TS
116          0.920200   POSITIVE        0.999866   1
117          0.962748   POSITIVE        0.999811   1
183          0.937303   POSITIVE        0.994876   1

[3 rows x 21 columns]
```

Performance Evaluation

Now I will create the following columns:

- `VADER_Prediction` - 1 for positive where vader_compound > 0, 0 for negative where vader_compound <= 0
- `RoBERTa_Prediction` - 1 for positive where roberta_compound > 0, 0 for negative where roberta_compound <= 0
- `HuggingFace_Prediction` - : 1 for positive where sentiment_score > 0.5, 0 for negative where sentiment_score <= 0.5

Reason: These predictions are necessary for evaluation metrics like accuracy, precision, recall, and F1 score to gauge the model performance. The continuous or probabilistic outputs (vader_compound, roberta_compound, sentiment_score) are converted into binary

predictions so that they can be compared with the true sentiment of ratings (assigned earlier).

In [36]:
```
# VADER Prediction
sentiment_df['VADER_Prediction'] = sentiment_df['vader_compound'].apply(lambda x: 1

# RoBERTa Prediction
sentiment_df['RoBERTa_Prediction'] = sentiment_df['roberta_compound'].apply(lambda

# HuggingFace Prediction
sentiment_df['HuggingFace_Prediction'] = sentiment_df['sentiment_score'].apply(lamb
```

In [37]:
```
print(sentiment_df.head())
```

```
      Id   ProductId          UserId     ProfileName  HelpfulnessNumerator  \
116  117  B0026Y3YBK  A3P60QLFDDCHOY   Giordano "GB"                     2
117  118  B0026Y3YBK  A38BUM0OXH38VK    singlewinder                     0
183  184  B001KUUNP6  A262Z0S6PT9U16    Lee Thombley                     3
375  376  B0087HW5E2  A139RTDNMU3WY5    blanket lady                     2
394  395  B001ELL6O8  A13T2G4T8LR8XA  First Time Mom                     2

     HelpfulnessDenominator  Rating        Time  \
116                       2       5  1304899200
117                       0       5  1347667200
183                       3       5  1292716800
375                       2       5  1339977600
394                       2       5  1189296000

                                      Summary  \
116                                Great cookies
117                            Best everyday cookie!
183  Perfect for gluten-free chocolate chip cookies
375          Greatest Oil since slice bread !!!!!!!
394                             Delisious Pancakes

                                                  Text  ...  roberta_neg  \
116  I'm Italian and I lived in Italy for years. I ...  ...     0.004493
117  In the 1980s I spent several summers in Italy....  ...     0.001362
183  We made chocolate chip cookies with BRM Garban...  ...     0.001589
375  I have used this oil for several years and it ...  ...     0.013173
394  Before I discover this mix on Amazon I always ...  ...     0.004090

     roberta_neu  roberta_pos  roberta_compound  sentiment  sentiment_score  \
116     0.070814     0.924693          0.920200   POSITIVE         0.999866
117     0.034527     0.964110          0.962748   POSITIVE         0.999811
183     0.059519     0.938892          0.937303   POSITIVE         0.994876
375     0.099402     0.887424          0.874251   POSITIVE         0.999745
394     0.254064     0.741847          0.737757   NEGATIVE         0.994825

     TS  VADER_Prediction RoBERTa_Prediction  HuggingFace_Prediction
116   1                 1                 1                       1
117   1                 1                 1                       1
183   1                 1                 1                       1
375   1                 1                 1                       1
394   1                 0                 1                       1

[5 rows x 24 columns]
```

Calculating Accuracy, Precision, Recall, F1 score for all three models

In [38]:
```
# Metrics for VADER
vader_accuracy = accuracy_score(sentiment_df['TS'], sentiment_df['VADER_Prediction'
vader_precision = precision_score(sentiment_df['TS'], sentiment_df['VADER_Predictio
vader_recall = recall_score(sentiment_df['TS'], sentiment_df['VADER_Prediction'])
vader_f1 = f1_score(sentiment_df['TS'], sentiment_df['VADER_Prediction'])

# Print the results for all three models
```

```
print("VADER Metrics: Accuracy, Precision, Recall, F1")
print(vader_accuracy, vader_precision, vader_recall, vader_f1)
```

```
VADER Metrics: Accuracy, Precision, Recall, F1
0.6908942795076032 0.9415397874174087 0.685454355327826 0.7933434190620272
```

In [39]:
```
# Metrics for RoBERTa
roberta_accuracy = accuracy_score(sentiment_df['TS'], sentiment_df['RoBERTa_Predict
roberta_precision = precision_score(sentiment_df['TS'], sentiment_df['RoBERTa_Predi
roberta_recall = recall_score(sentiment_df['TS'], sentiment_df['RoBERTa_Prediction'
roberta_f1 = f1_score(sentiment_df['TS'], sentiment_df['RoBERTa_Prediction'])

print("RoBERTa Metrics: Accuracy, Precision, Recall, F1")
print(roberta_accuracy, roberta_precision, roberta_recall, roberta_f1)
```

```
RoBERTa Metrics: Accuracy, Precision, Recall, F1
0.8715604634322954 0.9245204336947456 0.9273240614869811 0.925920125293657
```

In [40]:
```
# Metrics for Hugging Face Pipeline (Transformers)
hfp_accuracy = accuracy_score(sentiment_df['TS'], sentiment_df['HuggingFace_Predict
hfp_precision = precision_score(sentiment_df['TS'], sentiment_df['HuggingFace_Predi
hfp_recall = recall_score(sentiment_df['TS'], sentiment_df['HuggingFace_Prediction'
hfp_f1 = f1_score(sentiment_df['TS'], sentiment_df['HuggingFace_Prediction'])

print("HuggingFace Pipeline Metrics: Accuracy, Precision, Recall, F1")
print(hfp_accuracy, hfp_precision, hfp_recall, hfp_f1)
```

```
HuggingFace Pipeline Metrics: Accuracy, Precision, Recall, F1
0.8655865314989138 0.8655865314989138 1.0 0.9279510940759789
```

Collating the results

In [41]:
```
table = PrettyTable()

# Set column names
table.field_names = ["Model", "Accuracy", "Precision", "Recall", "F1 Score"]

# Add rows for each model's metrics, format as percentages with 2 decimal places
table.add_row(["VADER", f"{vader_accuracy * 100:.2f}%", f"{vader_precision * 100:.2
table.add_row(["RoBERTa", f"{roberta_accuracy * 100:.2f}%", f"{roberta_precision *
table.add_row(["HuggingFace Pipeline", f"{hfp_accuracy * 100:.2f}%", f"{hfp_precisi

# Print the table
print(table)
```

```
+----------------------+----------+-----------+---------+----------+
|        Model         | Accuracy | Precision |  Recall | F1 Score |
+----------------------+----------+-----------+---------+----------+
|        VADER         |  69.09%  |   94.15%  |  68.55% |  79.33%  |
|       RoBERTa        |  87.16%  |   92.45%  |  92.73% |  92.59%  |
| HuggingFace Pipeline |  86.56%  |   86.56%  | 100.00% |  92.80%  |
+----------------------+----------+-----------+---------+----------+
```

Inference:

RoBERTa Stands Out: RoBERTa demonstrates the highest overall performance, with an accuracy of 87.16% and an F1 score of 92.59%

It exhibits a strong balance between precision (92.45%) and recall (92.73%), indicating it is good at both identifying positive sentiments correctly and capturing most of the actual positive instances.

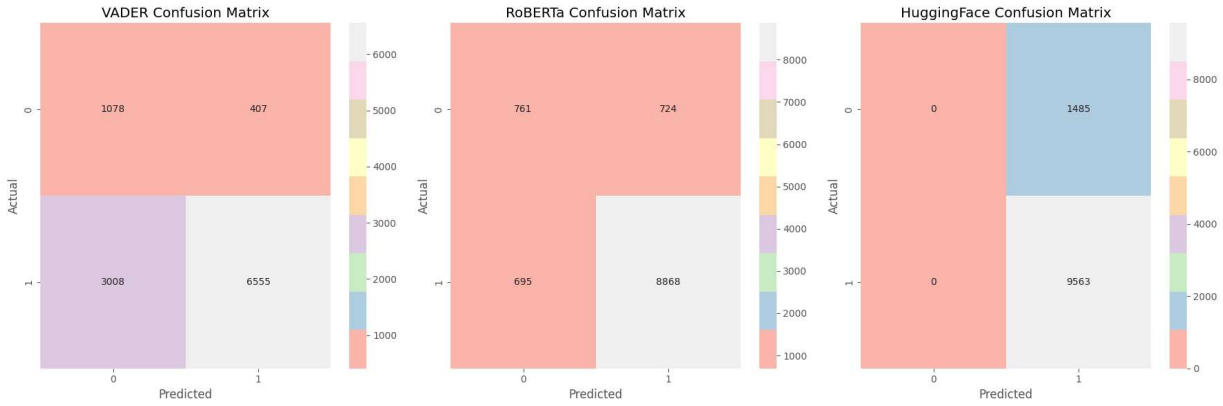
HuggingFace Pipeline Strong Recall: The HuggingFace Pipeline also performs well, with an accuracy of 86.56% and an F1 score of 92.80%. It achieves a perfect recall of 100.00%, meaning it successfully identified all actual positive sentiments in the dataset. However it has the lowest precision of the three models, meaning that while it found all the positive results, it also labeled a lot of negative results as positive.

VADER's Limitations: VADER shows the lowest performance among the three models. While it has high precision (94.15%), its recall is significantly lower (68.55%), resulting in a lower F1 score (79.33%). This suggests that VADER is conservative in its positive sentiment predictions, missing many actual positive instances. It also has the lowest accuracy of the three models.

Confusion Matrix

In [42]:
```
# Confusion matrices for each model
vader_conf_matrix = confusion_matrix(sentiment_df['TS'], sentiment_df['VADER_Predic
roberta_conf_matrix = confusion_matrix(sentiment_df['TS'], sentiment_df['RoBERTa_Pr
hf_conf_matrix = confusion_matrix(sentiment_df['TS'], sentiment_df['HuggingFace_Pre
```

In [43]:
```
# Create a figure with 3 subplots
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# List of confusion matrices and corresponding titles
conf_matrices = [vader_conf_matrix, roberta_conf_matrix, hf_conf_matrix]
titles = ['VADER Confusion Matrix', 'RoBERTa Confusion Matrix', 'HuggingFace Confus

# Plot each confusion matrix using i as index
for i in range(3):
    sns.heatmap(conf_matrices[i], annot=True, fmt ='d', cmap='Pastel1', ax=axes[i])

    axes[i].set_title(titles[i])
    axes[i].set_xlabel("Predicted")
    axes[i].set_ylabel("Actual")

# Adjust layout and display
plt.tight_layout()
plt.show()
```



Inference

- Transformer-based models (RoBERTa and HuggingFace Pipeline) **significantly outperform** the lexicon-based VADER model in sentiment analysis accuracy and F1 score.
- RoBERTa provides a well-balanced performance, while the HuggingFace Pipeline excels in recall, potentially at the cost of precision.

For this data set, if it is more important to find every positive sentiment, even if some negative sentiments are incorrectly labeled, then the hugging face pipeline would be the best choice. If it is more important to have an overall balanced result, then Roberta is the better choice.