In [1]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\shubh\OneDrive\Desktop\guvi session\healthcare_reviews.csv.csv")
```

In [2]:
```
df.head()
```

Out[2]:

	Review_Text	Rating
0	I have mixed feelings about my experience.	4
1	The staff was caring and attentive. I couldn't...	5
2	I have mixed feelings about my experience.	5
3	I have mixed feelings about my experience.	5
4	The healthcare provider was excellent. I had a...	3

In [3]:
```
import numpy as np

def create_sentiment(rating):

    if rating==1 or rating==2:
        return -1 # negative sentiment
    elif rating==4 or rating==5:
        return 1 # positive sentiment
    else:
        return 0 # neutral sentiment

df['Sentiment'] = df['Rating'].apply(create_sentiment)
```

In [4]:
```
df.head(10)
```

Out[4]:

	Review_Text	Rating	Sentiment
0	I have mixed feelings about my experience.	4	1
1	The staff was caring and attentive. I couldn't...	5	1
2	I have mixed feelings about my experience.	5	1
3	I have mixed feelings about my experience.	5	1
4	The healthcare provider was excellent. I had a...	3	0
5	The staff was caring and attentive. I couldn't...	4	1
6	NaN	2	-1
7	I had a bad experience with this healthcare pr...	2	-1
8	I have mixed feelings about my experience.	3	0
9	I have mixed feelings about my experience.	5	1

In [5]:
```
df['Review_Text'][5]
```

Out[5]:
```
"The staff was caring and attentive. I couldn't be happier. "
```

Handling missing data

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Review_Text  900 non-null    object
 1   Rating       1000 non-null   int64 
 2   Sentiment    1000 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 23.6+ KB
```

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

Out[7]:
```
Review_Text    100
Rating           0
Sentiment        0
dtype: int64
```

In [8]:
```
df['Review_Text'].value_counts()
```

Out[8]:
```
Review_Text
I'm very satisfied with the service I received. Highly recommended.          117
The service was disappointing. I won't be coming back.                       106
The staff was caring and attentive. I couldn't be happier.                   104
The healthcare provider was excellent. I had a great experience.             103
I have mixed feelings about my experience.                                    98
My experience was terrible. I would not recommend this provider.              97
It was an average experience. Neither good nor bad.                           94
The service was okay, but nothing exceptional.                                91
I had a bad experience with this healthcare provider. Avoid if possible.      90
Name: count, dtype: int64
```

In [9]:
```
df['Review_Text']=df['Review_Text'].fillna('i have mixed feelings about my experience.')
```

In [10]:
```
df.head(10)
```

Out[10]:

	Review_Text	Rating	Sentiment
0	I have mixed feelings about my experience.	4	1
1	The staff was caring and attentive. I couldn't...	5	1
2	I have mixed feelings about my experience.	5	1
3	I have mixed feelings about my experience.	5	1
4	The healthcare provider was excellent. I had a...	3	0
5	The staff was caring and attentive. I couldn't...	4	1
6	i have mixed feelings about my experience.	2	-1
7	I had a bad experience with this healthcare pr...	2	-1
8	I have mixed feelings about my experience.	3	0
9	I have mixed feelings about my experience.	5	1

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Review_Text    0
Rating         0
Sentiment      0
dtype: int64
```

lower case conversion

In [12]:
```
import string
```

In [13]:
```
df['Review_Text']=df['Review_Text'].str.lower()
```

In [14]:
```
df.head(10)
```

Out[14]:

	Review_Text	Rating	Sentiment
0	i have mixed feelings about my experience.	4	1
1	the staff was caring and attentive. i couldn't...	5	1
2	i have mixed feelings about my experience.	5	1
3	i have mixed feelings about my experience.	5	1
4	the healthcare provider was excellent. i had a...	3	0
5	the staff was caring and attentive. i couldn't...	4	1
6	i have mixed feelings about my experience.	2	-1
7	i had a bad experience with this healthcare pr...	2	-1
8	i have mixed feelings about my experience.	3	0
9	i have mixed feelings about my experience.	5	1

Removal punctuation

In [15]:
```
punct=string.punctuation
```

In [16]:
```
def remove_punct(x):
    return x.translate(str.maketrans("","", punct))

df['Review_Text']=df['Review_Text'].apply(lambda x: remove_punct(x))
```

In [17]:
```
df.head(10)
```

Out[17]:

	Review_Text	Rating	Sentiment
0	i have mixed feelings about my experience	4	1
1	the staff was caring and attentive i couldnt b...	5	1
2	i have mixed feelings about my experience	5	1
3	i have mixed feelings about my experience	5	1
4	the healthcare provider was excellent i had a ...	3	0
5	the staff was caring and attentive i couldnt b...	4	1
6	i have mixed feelings about my experience	2	-1
7	i had a bad experience with this healthcare pr...	2	-1
8	i have mixed feelings about my experience	3	0
9	i have mixed feelings about my experience	5	1

Tokenized data

In [18]:
```
import re

def tokenize(x):
    tokens_txt = re.split('\W+', x)
    return tokens_txt

df['Review_Text']=df['Review_Text'].apply(lambda x: tokenize(x))

df.head(10)
```

Out[18]:

	Review_Text	Rating	Sentiment
0	[i, have, mixed, feelings, about, my, experien...	4	1
1	[the, staff, was, caring, and, attentive, i, c...	5	1
2	[i, have, mixed, feelings, about, my, experien...	5	1
3	[i, have, mixed, feelings, about, my, experien...	5	1
4	[the, healthcare, provider, was, excellent, i,...	3	0
5	[the, staff, was, caring, and, attentive, i, c...	4	1
6	[i, have, mixed, feelings, about, my, experience]	2	-1
7	[i, had, a, bad, experience, with, this, healt...	2	-1
8	[i, have, mixed, feelings, about, my, experien...	3	0
9	[i, have, mixed, feelings, about, my, experien...	5	1

Remove stopwords

In [19]:
```
import nltk
stopwords = nltk.corpus.stopwords.words('english')
stopwords[0:10]
```

Out[19]:
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

In [20]:
```
def remove_stopwords(txt):
    txt_clean = [word  for word in txt if word not in stopwords]
    return txt_clean

df['Review_Text']=df['Review_Text'].apply(lambda x: remove_stopwords(x))
df.head(10)
```

Out[20]:

	Review_Text	Rating	Sentiment
0	[mixed, feelings, experience,]	4	1
1	[staff, caring, attentive, couldnt, happier,]	5	1
2	[mixed, feelings, experience,]	5	1
3	[mixed, feelings, experience,]	5	1
4	[healthcare, provider, excellent, great, exper...	3	0
5	[staff, caring, attentive, couldnt, happier,]	4	1
6	[mixed, feelings, experience]	2	-1
7	[bad, experience, healthcare, provider, avoid,...	2	-1
8	[mixed, feelings, experience,]	3	0
9	[mixed, feelings, experience,]	5	1

Split into test and train examples

In [62]:
```
import sklearn
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df['Review_Text'], df.Sentiment)
```

In [64]:
```
x_test
```

Out[64]:
```
455                       [mixed, feelings, experience, ]
65          [service, disappointing, wont, coming, back, ]
275         [staff, caring, attentive, couldnt, happier, ]
926     [experience, terrible, would, recommend, provi...
22          [service, disappointing, wont, coming, back, ]
                              ...
89                          [mixed, feelings, experience]
835                       [mixed, feelings, experience, ]
637     [im, satisfied, service, received, highly, rec...
667               [service, okay, nothing, exceptional, ]
764         [staff, caring, attentive, couldnt, happier, ]
Name: Review_Text, Length: 250, dtype: object
```

In [65]:
```
y_test
```

Out[65]:
```
455    0
65     1
275    0
926   -1
22     1
      ..
89     1
835    1
637    1
667    1
764    1
Name: Sentiment, Length: 250, dtype: int64
```

In [22]:
```
y_train.isnull()
```

Out[22]:
```
864    False
917    False
112    False
293    False
430    False
       ...
4      False
859    False
438    False
41     False
623    False
Name: Sentiment, Length: 750, dtype: bool
```

In [23]:
```
y_train.value_counts()
```

Out[23]:
```
Sentiment
 1    321
-1    302
 0    127
Name: count, dtype: int64
```

In [55]:
```
y_train = df['Sentiment'].fillna("1")
```

In [34]:
```
x_train = df['Review_Text']
x_train = [" ".join(doc) for doc in x_train]
x_train = [str(doc) for doc in x_train if isinstance(doc, str) and len(doc) > 0]
```

In [57]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

```
# Create an instance of CountVectorizer
v = CountVectorizer()

# Fit and transform the training data
x_train_vec = v.fit_transform(x_train)

# Transform the test data using the fitted vectorizer
x_test_strings = [' '.join(words) for words in x_test]
x_test_vec = v.transform(x_test_strings)
```

In [27]:
```
print("Number of samples in x_train_vec:", x_train_vec.shape[0])
print("Number of samples in y_train:", len(y_train))
print("Number of samples in x_test_vec:", x_test_vec.shape[0])
print("Number of samples in y_test:", len(y_test))
```

```
Number of samples in x_train_vec: 1000
Number of samples in y_train: 1000
Number of samples in x_test_vec: 250
Number of samples in y_test: 250
```

Use clasification model

In [61]:
```
from sklearn import svm
clf_svm = svm.SVC(kernel = "linear")
clf_svm.fit(x_train_vec, y_train)
```

Out[61]:
```
SVC(kernel='linear')
```

Test Accuracy

In [29]:
```
clf_svm.score(x_test_vec, y_test)
```

Out[29]:
```
0.492
```

In [30]:
```
from sklearn.metrics import f1_score
f1_score(y_test, clf_svm.predict(x_test_vec), average = None)
```

Out[30]:
```
array([0.46857143, 0.        , 0.57746479])
```

In [37]:
```
rev = ["My experience was good. I would recommend this provider."]
rev_vec = v.transform(rev)
clf_svm.predict(rev_vec)
```

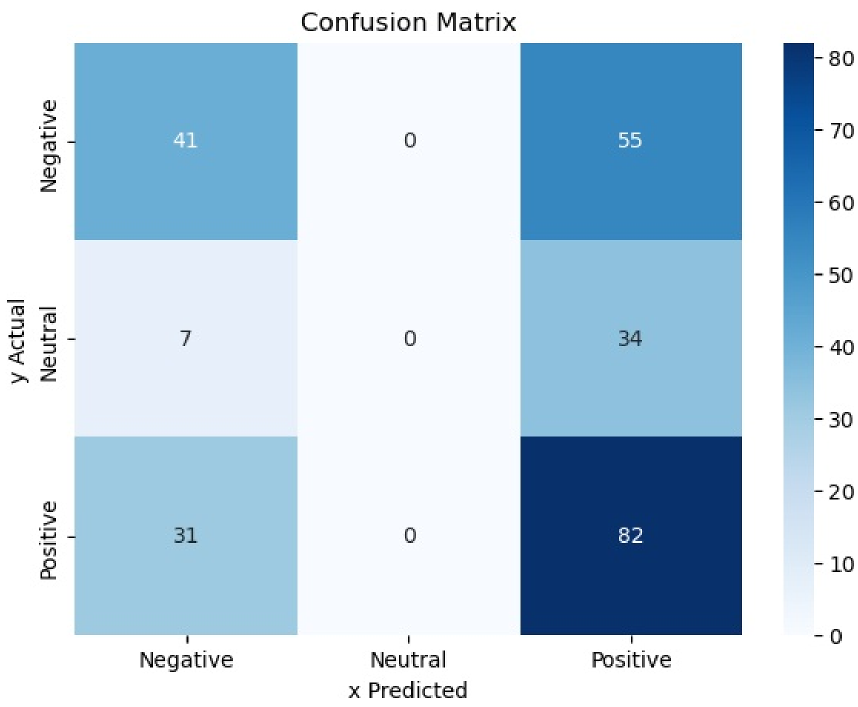
Out[37]:
```
array([1], dtype=int64)
```

Data Visualization

In [32]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
```

In [43]:
```
# Generate confusion matrix
cm = confusion_matrix(y_test, clf_svm.predict(x_test_vec))

# Create a heatmap for the confusion matrix
plt.figure(figsize=(7, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Neutral', 'Positive'], yticklabels
plt.title('Confusion Matrix')
plt.xlabel('x Predicted')
plt.ylabel('y Actual')
plt.show()
```



Report

In [46]:

```
# Generate classification report
report = classification_report(y_test, clf_svm.predict(x_test_vec), target_names=['Negative', 'Neutral', 'Posit

# Print the classification report
print(report)
```

```
              precision    recall  f1-score   support

    Negative       0.52      0.43      0.47        96
     Neutral       0.00      0.00      0.00        41
    Positive       0.48      0.73      0.58       113

    accuracy                           0.49       250
   macro avg       0.33      0.38      0.35       250
weighted avg       0.42      0.49      0.44       250
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division
` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division
` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division
` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Precision

-For the 'Negative' class: Precision = 0.52 (52% of instances predicted as negative were actually negative). -For the 'Neutral' class: Precision = 0.00 (Precision is undefined since there are no True Positives for 'Neutral'). -For the 'Positive' class: Precision = 0.48 (48% of instances predicted as positive were actually positive).

Recall (Sensitivity)

Recall is a measure of how many actual positive instances were correctly predicted by the model. Formula: True Positives/ True Positives + False Negatives

In your report:

For the 'Negative' class: Recall = 0.43 (43% of actual negative instances were correctly predicted). For the 'Neutral' class: Recall = 0.00 (Recall is undefined since there are no True Positives for 'Neutral'). For the 'Positive' class: Recall = 0.73 (73% of actual positive instances were correctly predicted)

F1-Score

The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. Formula: 2 · Precision · Recall Precision + Recall

In your report:

For the 'Negative' class: F1-Score = 0.47. For the 'Neutral' class: F1-Score = 0.00 (F1-Score is undefined since there are no True Positives for 'Neutral'). For the 'Positive' class: F1-Score = 0.58. Support:

Support is the number of actual occurrences of each class in the test dataset. In your report:

For the 'Negative' class: Support = 96. For the 'Neutral' class: Support = 41. For the 'Positive' class: Support = 113.

In [53]:

```
# Plot the distribution of sentiment classes
plt.figure(figsize=(6, 4))
sns.countplot(x=y_train, palette='viridis')
plt.title('Distribution of Sentiment Classes in Training Data')
plt.xlabel('x-axis Sentiment Class')
plt.ylabel('y-axis Count')
plt.show()
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is dep
recated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shubh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is dep
recated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shubh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is dep
recated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

In [45]:

```
from wordcloud import WordCloud

# Combine all reviews into a single string
all_reviews = ' '.join([' '.join(words) for words in x_train])

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_reviews)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Reviews')
plt.show()
```

In [49]:
```
# Save figures
plt.savefig('confusion_matrix.png')
```

```
<Figure size 640x480 with 0 Axes>
```

In [51]:
```
# Save classification report to a text file
with open('classification_report.txt', 'w') as file:
    file.write(report)
```

In [ ]: