In [335]:

```
import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from transformers import pipeline
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
[nltk_data] Error loading punkt: <urlopen error [Errno 8] nodename no
r
[nltk_data]     servname provided, or not known>
[nltk_data] Error loading stopwords: <urlopen error [Errno 8] nodenam
e
[nltk_data]     nor servname provided, or not known>
[nltk_data] Error loading wordnet: <urlopen error [Errno 8] nodename
[nltk_data]     nor servname provided, or not known>
```

In [336]:

```
file_path = '/Users/jannyvelazquez/Downloads/Store_Metrics.csv'
```

In [337]:

```
df = pd.read_csv(file_path)
```

In [338]:

```
#reading dataset
df.head()
```

Out[338]:

	Date	OSAT	Cleanliness	Friendliness	TTRO	Accuracy	OSAT Comments < 4	OSAT Comments = 4	OS Commer :
0	3/31/24 9:21	4	NaN	NaN	3.0	5.0	NaN	Food always tastes good! While the mobile orde...	N
1	3/30/24 11:42	5	5.0	5.0	5.0	5.0	NaN	NaN	N
2	3/30/24 11:35	5	5.0	5.0	NaN	NaN	NaN	NaN	Cle gr selecti frienc easy a tc
3	3/29/24 22:11	5	5.0	5.0	4.0	5.0	NaN	NaN	I alwa come this Wa and serv is gre
4	3/29/24 18:52	5	5.0	5.0	5.0	5.0	NaN	NaN	N

In [339]:

```
#combining
OSAT_Comments = df["OSAT Comments < 4"].combine_first(df["OSAT Comments

df['Combined OSAT Comments'] = OSAT_Comments

df.head()
```

Out[339]:

	Date	OSAT	Cleanliness	Friendliness	TTRO	Accuracy	OSAT Comments < 4	OSAT Comments = 4	OS Commen
0	3/31/24 9:21	4	NaN	NaN	3.0	5.0	NaN	Food always tastes good! While the mobile orde...	N
1	3/30/24 11:42	5	5.0	5.0	5.0	5.0	NaN	NaN	N
2	3/30/24 11:35	5	5.0	5.0	NaN	NaN	NaN	NaN	Cle gr selecti friend easy a tc
3	3/29/24 22:11	5	5.0	5.0	4.0	5.0	NaN	NaN	I alwa come this Wa and serv is gre
4	3/29/24 18:52	5	5.0	5.0	5.0	5.0	NaN	NaN	N

In [340]:

```
#droping 'OSAT Comments < 4', 'OSAT Comments = 4', and 'OSAT Comments
df2 = df.drop(['OSAT Comments < 4', 'OSAT Comments = 4', 'OSAT Comment

df2.head()
```

Out[340]:

	Date	OSAT	Cleanliness	Friendliness	TTRO	Accuracy	Combined OSAT Comments
0	3/31/24 9:21	4	NaN	NaN	3.0	5.0	Food always tastes good! While the mobile orde...
1	3/30/24 11:42	5	5.0	5.0	5.0	5.0	NaN
2	3/30/24 11:35	5	5.0	5.0	NaN	NaN	Clean, great selection, friendly, easy app to ...
3	3/29/24 22:11	5	5.0	5.0	4.0	5.0	I always come to this Wawa and service is grea...
4	3/29/24 18:52	5	5.0	5.0	5.0	5.0	NaN

In [341]:

```
#droping NaNs
df2 = df2.dropna()
```

In [342]:

```
#making dataset report
from ydata_profiling import ProfileReport
ProfileReport(df2)
```

```
Summarize dataset: 100%                16/16 [00:00<00:00, 44.83it/s, Completed]

Generate report structure: 100%        1/1 [00:00<00:00, 1.21it/s]

Render HTML: 100%                      1/1 [00:00<00:00, 11.52it/s]
```

Average record size in memory	64.0 B

Variable types

DateTime	1
Categorical	5
Text	1

Alerts

`Friendliness` is highly overall correlated with `OSAT`	High correlation
`OSAT` is highly overall correlated with `Friendliness`	High correlation
`OSAT` is highly imbalanced (69.6%)	Imbalance
`Cleanliness` is highly imbalanced (69.5%)	Imbalance
`Friendliness` is highly imbalanced (73.3%)	Imbalance
`TTRO` is highly imbalanced (58.7%)	Imbalance
`Accuracy` is highly imbalanced (69.6%)	Imbalance

Reproduction

Analysis started	2024-04-15 22:05:33.191518
Analysis finished	2024-04-15 22:05:33.463320
Duration	0.27 seconds
Software version	ydata-profiling vv4.6.4 (https://github.com/ydataai/ydata-profiling)
Download configuration	config.json (data:text/plain;charset=utf-8,%7B%22title%22%3A%20%22Pandas%20Profiling%20Report%2

Out[342]:

Models

1. Sentiment Analysis Model for Emotion Detection

predict the emotional tone of customer comments and checks if their feelings match the ratings they gave.

In [284]:

```
def preprocess_text(text):
    # Check if the text is a string
    if not isinstance(text, str):
        return ""  # Or return some placeholder text like "missingdata
    text = text.lower()  # Lowercase text
    text = re.sub(r'\W', ' ', text)  # Remove all special characters
    tokens = word_tokenize(text)  # Tokenize
    # Assuming stopwords and lemmatizer have been downloaded and impor
    tokens = [word for word in tokens if word not in stopwords.words('
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Lemmat
    return ' '.join(tokens)

# Applying the preprocessing function to your comments column again
# Assuming your DataFrame is named df and has been loaded correctly
df['processed_comments'] = df['Combined OSAT Comments'].apply(preproce
```

In [285]:

```
# Initialize the TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the preprocessed comments to create a feature matr
X = vectorizer.fit_transform(df['processed_comments'])

# Assuming your ratings are stored in a column called 'Overall satisfa
y = df['OSAT']
```

In [286]:

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Initialize and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the testing set
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, predictions))
```

```
Accuracy: 0.8701298701298701
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         8
           2       0.00      0.00      0.00         6
           3       0.00      0.00      0.00         2
           4       1.00      0.06      0.11        36
           5       0.87      1.00      0.93       333

    accuracy                           0.87       385
   macro avg       0.37      0.21      0.21       385
weighted avg       0.85      0.87      0.81       385


/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/sklearn/
metrics/_classification.py:1469: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior
.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/sklearn/
metrics/_classification.py:1469: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior
.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/sklearn/
metrics/_classification.py:1469: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior
.
  _warn_prf(average, modifier, msg_start, len(result))
```

The classification report showes model performs well in predicting class 5 (highest satisfaction rating), with high precision, recall, and F1-score. However, it struggles with classes 1, 2, and 3, where the precision, recall, and F1-score are all very low.

2. Sentiment Analysis Model for Trend Analysis

identify broader trends within customer feedback, cluster similar reviews or identify specific products/services mentioned.

In [287]:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

In [292]:

```
# Text preprocessing
def preprocess_text(text):
    if isinstance(text, str):
        text = text.lower()  # Lowercase text
        text = re.sub(r'\W', ' ', text)  # Remove special characters
        tokens = word_tokenize(text)  # Tokenize
        tokens = [word for word in tokens if word not in stopwords.wor
        lemmatizer = WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Le
        return ' '.join(tokens)
    else:
        return ''

# Apply text preprocessing to the 'Combined OSAT Comments' column
df2['processed_text'] = df2['Combined OSAT Comments'].apply(preprocess

# Display the first few rows of the processed dataset
print(df2.head())
```

```
              Date  OSAT  Cleanliness  Friendliness  TTRO  Accuracy  \
3   3/29/24 22:11     5          5.0           5.0   4.0       5.0   
8    3/27/24 9:40     5          5.0           5.0   5.0       5.0   
10  3/25/24 19:20     5          5.0           5.0   5.0       5.0   
15  3/21/24 16:36     4          5.0           5.0   5.0       5.0   
17  3/20/24 11:54     5          5.0           5.0   5.0       4.0   

                               Combined OSAT Comments  \
3   I always come to this Wawa and service is grea...   
8   Food is good quality. Staff is friendly. Place...   
10                                  Awesome treatment   
15                                  Ricardo was amazing   
17  Because the food and the staff are wonders. Al...   

                                       processed_text  
3   always come wawa service great came grab quick...  
8   food good quality staff friendly place clean c...  
10                                  awesome treatment  
15                                     ricardo amazing  
17  food staff wonder also forget awesome everythi...  
```

In [293]:

```
# Feature extraction
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df2['processed_text'])

# Display the shape of the feature matrix
print("Shape of feature matrix:", X.shape)
```

```
Shape of feature matrix: (821, 1815)
```

In [294]:

```
# Clustering (K-means)
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X)

# Assign cluster labels to the dataset
df2['cluster'] = kmeans.labels_

# Display the first few rows of the dataset with cluster labels
print(df2.head())
```

```
             Date  OSAT  Cleanliness  Friendliness  TTRO  Accuracy  \
3   3/29/24 22:11     5          5.0           5.0   4.0       5.0   
8    3/27/24 9:40     5          5.0           5.0   5.0       5.0   
10  3/25/24 19:20     5          5.0           5.0   5.0       5.0   
15  3/21/24 16:36     4          5.0           5.0   5.0       5.0   
17  3/20/24 11:54     5          5.0           5.0   5.0       4.0   

                                Combined OSAT Comments  \
3   I always come to this Wawa and service is grea...   
8   Food is good quality. Staff is friendly. Place...   
10                                  Awesome treatment   
15                                 Ricardo was amazing   
17  Because the food and the staff are wonders. Al...   

                                       processed_text  cluster  
3   always come wawa service great came grab quick...        1  
8   food good quality staff friendly place clean c...        3  
10                                  awesome treatment        0  
15                                    ricardo amazing        0  
17  food staff wonder also forget awesome everythi...        0  

/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/sklearn/
cluster/_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

3. Linear Regression Model for Customer Satisfaction Factors

determine which factors most significantly affect customer satisfaction levels.

In [295]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
```

In [296]:

```
#Full Model
```

In [301]:

```
# Fill missing values with the mean
df_imputed = df.fillna(df.mean())

# Split the dataset into features (X) and target variable (y)
X = df_imputed[['Cleanliness', 'Friendliness', 'TTRO', 'Accuracy']]
y = df_imputed['OSAT']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Train the linear regression model
fullmodel = LinearRegression()
fullmodel.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = fullmodel.predict(X_test)

# Calculatr MSE
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate R-squared
r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)

# Coefficients of the linear regression model
coefficients = pd.DataFrame(fullmodel.coef_, X.columns, columns=['Coef
print("Coefficients:")
print(coefficients)
```

```
Mean Squared Error: 0.3346790023485397
R-squared: 0.37133702796714885
Coefficients:
              Coefficient
Cleanliness      0.189138
Friendliness     0.587789
TTRO             0.104531
Accuracy         0.163784

/var/folders/_0/f33hqpcd4ld8qchps19wfy3w0000gn/T/ipykernel_24740/7928
26721.py:2: FutureWarning: The default value of numeric_only in DataF
rame.mean is deprecated. In a future version, it will default to Fals
e. In addition, specifying 'numeric_only=None' is deprecated. Select
only valid columns or specify the value of numeric_only to silence th
is warning.
  df_imputed = df.fillna(df.mean())
```

In [302]:

```
#Full model with interactions
```

In [303]:

```
# Drop non-numeric columns
numeric_columns = df.columns[df.dtypes != 'object']
df_numeric = df[numeric_columns]

# Fill missing values with the mean
imputer = SimpleImputer(strategy='mean')
df_imputed = pd.DataFrame(imputer.fit_transform(df_numeric), columns=c

# Split the dataset into features (X) and target variable (y)
X = df_imputed[['Cleanliness', 'Friendliness', 'TTRO', 'Accuracy']]
y = df_imputed['OSAT']

# Create interaction terms
X['Cleanliness_Friendliness'] = X['Cleanliness'] * X['Friendliness']
X['Cleanliness_TTRO'] = X['Cleanliness'] * X['TTRO']
X['Cleanliness_Accuracy'] = X['Cleanliness'] * X['Accuracy']
X['Friendliness_TTRO'] = X['Friendliness'] * X['TTRO']
X['Friendliness_Accuracy'] = X['Friendliness'] * X['Accuracy']
X['TTRO_Accuracy'] = X['TTRO'] * X['Accuracy']

# Drop original columns
X = X.drop(['Cleanliness', 'Friendliness', 'TTRO', 'Accuracy'], axis=1

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Train the linear regression model
fullmodel_int = LinearRegression()
fullmodel_int.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = fullmodel_int.predict(X_test)

# Calculate MSE
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate R-squared
r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)

# Coefficients of the linear regression model
coefficients = pd.DataFrame(fullmodel_int.coef_, X.columns, columns=['
print("Coefficients:")
print(coefficients)
```

```
Mean Squared Error: 0.3422864228536423

R-squared: 0.35704720532909484
Coefficients:
                          Coefficient
```

```
Cleanliness_Friendliness     0.010791
Cleanliness_TTRO            -0.142840
Cleanliness_Accuracy         0.170037
Friendliness_TTRO            0.213383
Friendliness_Accuracy       -0.078954
TTRO_Accuracy               -0.053850

/var/folders/_0/f33hqpcd4ld8qchps19wfy3w0000gn/T/ipykernel_24740/1528
96629.py:14: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  X['Cleanliness_Friendliness'] = X['Cleanliness'] * X['Friendliness'
]
```

In [304]:

```
#Reduced Model
```

In [305]:

```
# Fill missing values with the mean
df_imputed = df.fillna(df.mean())

# Split the dataset into features (X) and target variable (y)
X = df_imputed[['Friendliness']]
y = df_imputed['OSAT']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Train the linear regression model
redmodel = LinearRegression()
redmodel.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = redmodel.predict(X_test)

# Calculate MSE
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate R-squared
r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)

# Coefficients of the linear regression model
coefficients = pd.DataFrame(redmodel.coef_, X.columns, columns=['Coeff
print("Coefficients:")
print(coefficients)
```

```
Mean Squared Error: 0.35298574100798513
R-squared: 0.3369495442794501
Coefficients:
              Coefficient
Friendliness     0.814789

/var/folders/_0/f33hqpcd4ld8qchps19wfy3w0000gn/T/ipykernel_24740/1583
333949.py:2: FutureWarning: The default value of numeric_only in Data
Frame.mean is deprecated. In a future version, it will default to Fal
se. In addition, specifying 'numeric_only=None' is deprecated. Select
only valid columns or specify the value of numeric_only to silence th
is warning.
  df_imputed = df.fillna(df.mean())
```

In [306]:

```
#Reduced model with interactions
```

In [307]:

```
# Drop non-numeric columns
numeric_columns = df.columns[df.dtypes != 'object']
df_numeric = df[numeric_columns]

# Fill missing values with the mean
imputer = SimpleImputer(strategy='mean')
df_imputed = pd.DataFrame(imputer.fit_transform(df_numeric), columns=c

# Split the dataset into features (X) and target variable (y)
X = df_imputed[['Cleanliness', 'Friendliness', 'TTRO', 'Accuracy']]
y = df_imputed['OSAT']

# Create interaction terms
X['Friendliness_TTRO'] = X['Friendliness'] * X['TTRO']


# Drop original columns
X = X.drop(['Cleanliness', 'Friendliness', 'TTRO', 'Accuracy'], axis=1

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Train the linear regression model
redmodel_int = LinearRegression()
redmodel_int.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = redmodel_int.predict(X_test)

# Calculate MSE
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate R-squared
r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)

# Coefficients of the linear regression model
coefficients = pd.DataFrame(redmodel_int.coef_, X.columns, columns=['C
print("Coefficients:")
print(coefficients)
```

```
Mean Squared Error: 0.3619239871134012
R-squared: 0.3201598911432786
Coefficients:
                   Coefficient
Friendliness_TTRO     0.100426

/var/folders/_0/f33hqpcd4ld8qchps19wfy3w0000gn/T/ipykernel_24740/1675
136666.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  X['Friendliness_TTRO'] = X['Friendliness'] * X['TTRO']
```

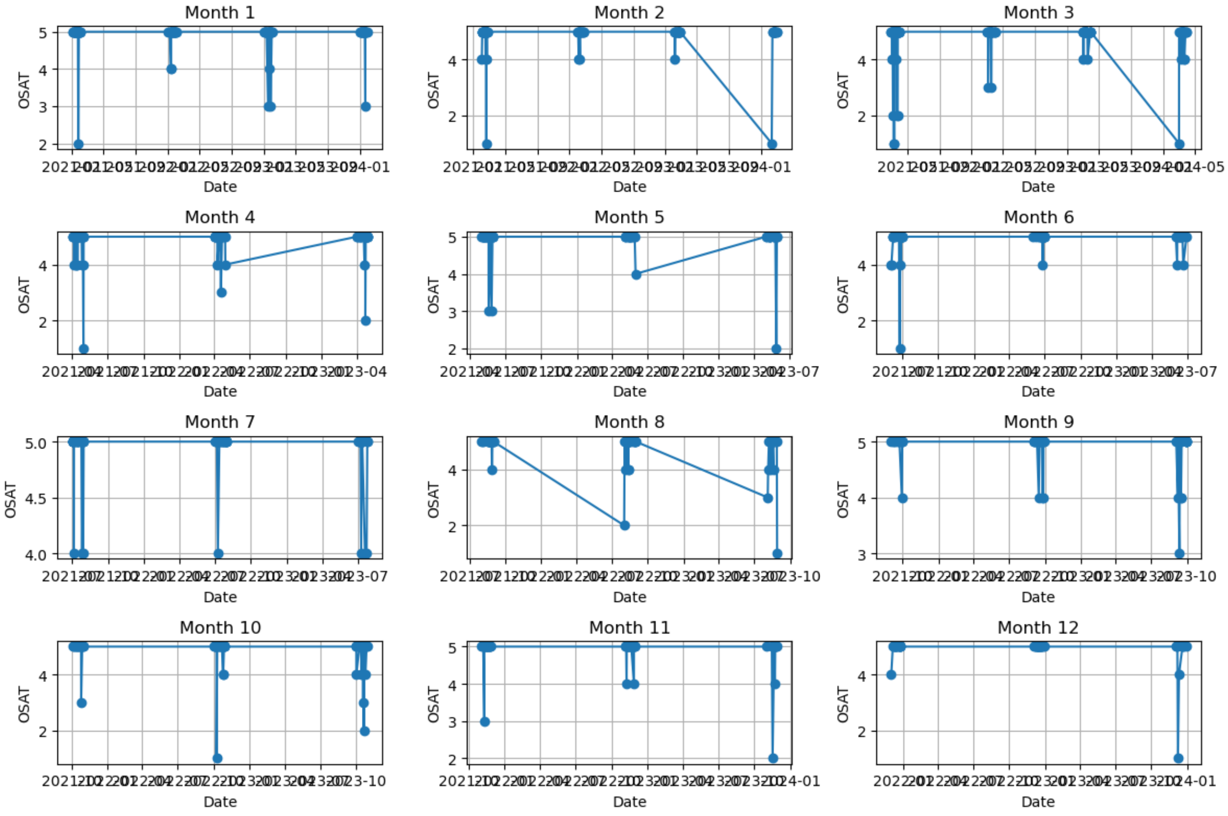
In [309]:
```
#The "Full Model" with interactions is the best-performing model overa
#The "Full Model" without interactions also performs well, with slight
```

4. Time Series Analysis for Trend Detection Over Time

determine if specific time periods where there's a notable concentration of similar positive or negative customer reviews.

In [311]:

```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df2 is your DataFrame with the "Date" and "Overall satisfac
# Convert "Date" column to datetime format
df2['Date'] = pd.to_datetime(df2['Date'])

# Extract year and month from the "Date" column
df2['Year'] = df2['Date'].dt.year
df2['Month'] = df2['Date'].dt.month

# Create a seasonal subseries plot
plt.figure(figsize=(12, 8))
for i in range(1, 13):  # Assuming data spans 12 months
    plt.subplot(4, 3, i)
    plt.plot(df2[df2['Month'] == i]['Date'], df2[df2['Month'] == i]['O
    plt.title(f'Month {i}')
    plt.xlabel('Date')
    plt.ylabel('OSAT')
    plt.grid(True)
plt.tight_layout()
plt.savefig('seasonal_subseries_plot.png')  # Save the plot as an imag
```



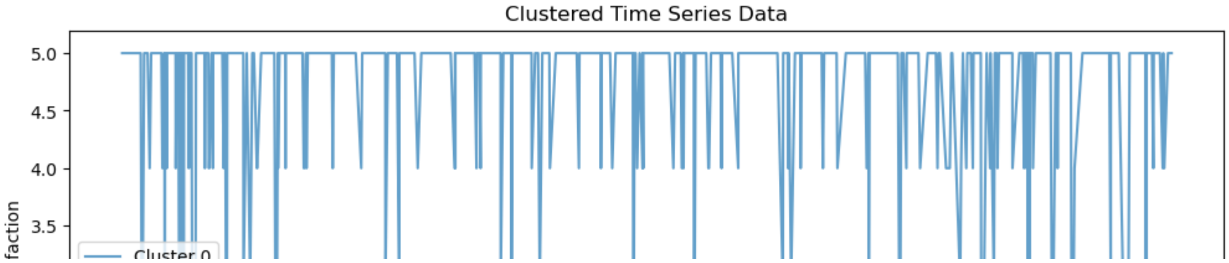
In [312]:

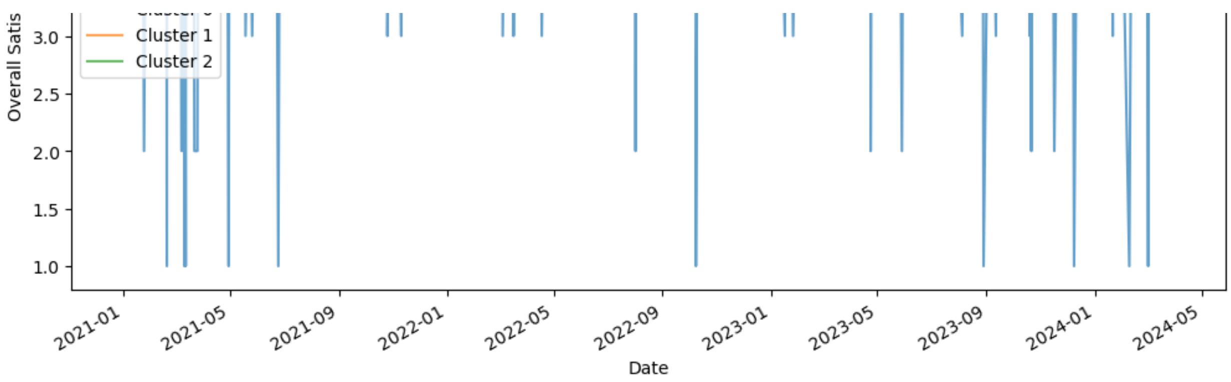
```
from tslearn.clustering import TimeSeriesKMeans
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance

# Load sample time series data
X_train, y_train, _, _ = CachedDatasets().load_dataset("Trace")

# Scale the time series data
X_train = TimeSeriesScalerMeanVariance().fit_transform(X_train)

# Initialize TimeSeriesKMeans clustering model
n_clusters = 3
km = TimeSeriesKMeans(n_clusters=n_clusters, verbose=True, random_stat

# Fit the model to the data
km.fit(X_train)

# Print cluster centers
print("Cluster centers:\n", km.cluster_centers_)

# Predict cluster labels
labels = km.predict(X_train)
print("Cluster labels:\n", labels)
```

```
  [-1.51330864]
  [-1.49873873]
  [-1.47731294]
  [-1.44761851]
  [-1.42858576]
  [-1.39298832]
  [-1.35550956]
  [-1.33258309]
  [-1.28577262]
  [-1.25137132]
  [-1.21684695]
  [-1.17975255]
  [-1.14330807]
  [-1.11204208]
  [-1.06561422]
  [-1.03351861]
  [-0.9927337 ]
  [-0.95287585]
  [-0.91139827]
  [-0.86954151]
```

In [323]:

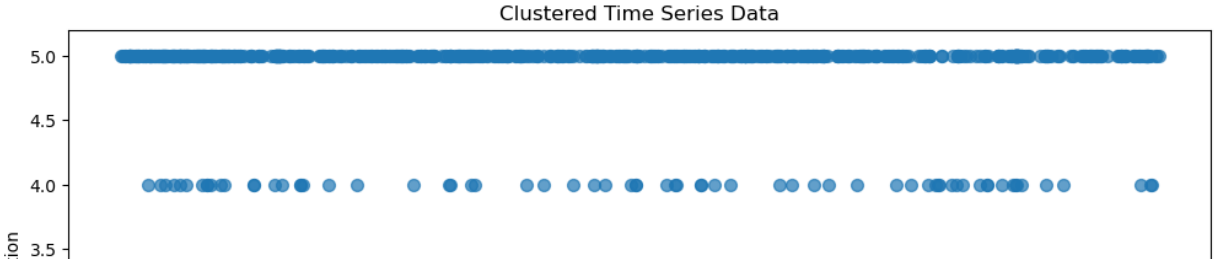
```
import pandas as pd
import matplotlib.pyplot as plt
from tslearn.clustering import TimeSeriesKMeans
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance

# Assuming df2 is your DataFrame with the "Date" and "OSAT" columns
# Convert "Date" column to datetime format
df2['Date'] = pd.to_datetime(df2['Date'])

# Set "Date" column as the index
df2.set_index('Date', inplace=True)

# Sort DataFrame by index (Date)
df2.sort_index(inplace=True)

# Normalize the time series data
scaler = TimeSeriesScalerMeanVariance(mu=0.0, std=1.0)  # Standardize
normalized_data = scaler.fit_transform(df2['OSAT'].values.reshape(-1,

# Define the number of clusters
n_clusters = 3  # You can adjust the number of clusters as needed

# Apply Time Series K-Means clustering
km = TimeSeriesKMeans(n_clusters=n_clusters, verbose=False, random_sta
cluster_labels = km.fit_predict(normalized_data)

# Add cluster labels to DataFrame
df2['Cluster'] = cluster_labels

# Plot each cluster separately
plt.figure(figsize=(12, 6))
for cluster_label in range(n_clusters):
    cluster_data = df2[df2['Cluster'] == cluster_label]['OSAT']
    cluster_data.plot(label=f'Cluster {cluster_label}', alpha=0.7)

plt.title('Clustered Time Series Data')
plt.xlabel('Date')
plt.ylabel('Overall Satisfaction')
plt.legend()
plt.show()
```

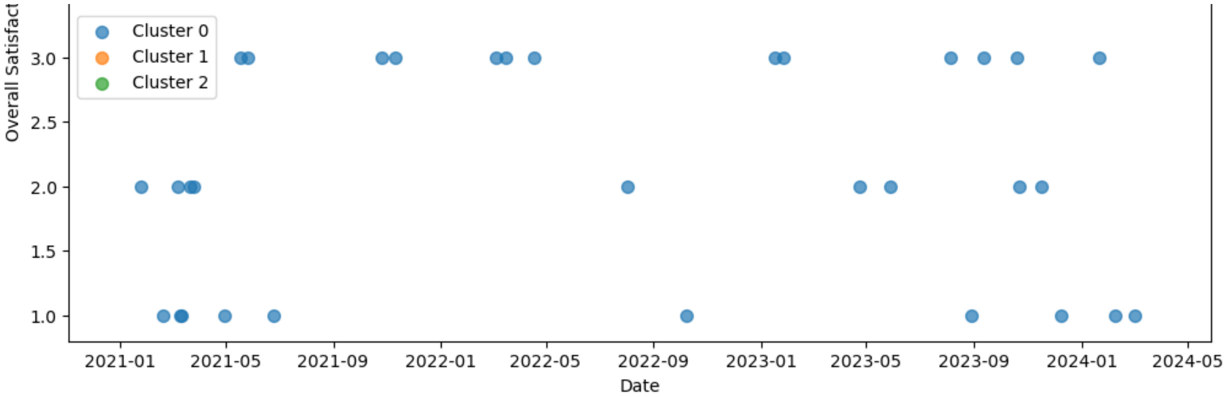



Cluster 0
Cluster 1
Cluster 2
Overall Satis
3.0
2.5
2.0
1.5
1.0
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
Date

In [333]:

```
import pandas as pd
import matplotlib.pyplot as plt
from tslearn.clustering import TimeSeriesKMeans
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance

# Assuming df2 is your DataFrame with the "Date" and "OSAT" columns
# Convert "Date" column to datetime format
df2['Date'] = pd.to_datetime(df2['Date'])

# Set "Date" column as the index
df2.set_index('Date', inplace=True)

# Sort DataFrame by index (Date)
df2.sort_index(inplace=True)

# Normalize the time series data
scaler = TimeSeriesScalerMeanVariance(mu=0.0, std=1.0)  # Standardize
normalized_data = scaler.fit_transform(df2['OSAT'].values.reshape(-1,

# Define the number of clusters
n_clusters = 3  # You can adjust the number of clusters as needed

# Apply Time Series K-Means clustering
km = TimeSeriesKMeans(n_clusters=n_clusters, verbose=False, random_sta
cluster_labels = km.fit_predict(normalized_data)

# Add cluster labels to DataFrame
df2['Cluster'] = cluster_labels

# Plot the clusters
plt.figure(figsize=(12, 6))
for cluster_label in range(n_clusters):
    cluster_data = df2[df2['Cluster'] == cluster_label]
    plt.scatter(cluster_data.index, cluster_data['OSAT'], label=f'Clus

plt.title('Clustered Time Series Data')
plt.xlabel('Date')
plt.ylabel('Overall Satisfaction')
plt.legend()
plt.show()
```





In [ ]:

```

```

In [343]:

```
%matplotlib inline
import pandas as pd

# Assuming 'date_column' is the name of the column containing the date
df2['Date'] = pd.to_datetime(df2['Date'])
df2.set_index('Date', inplace=True)
```

In [268]:

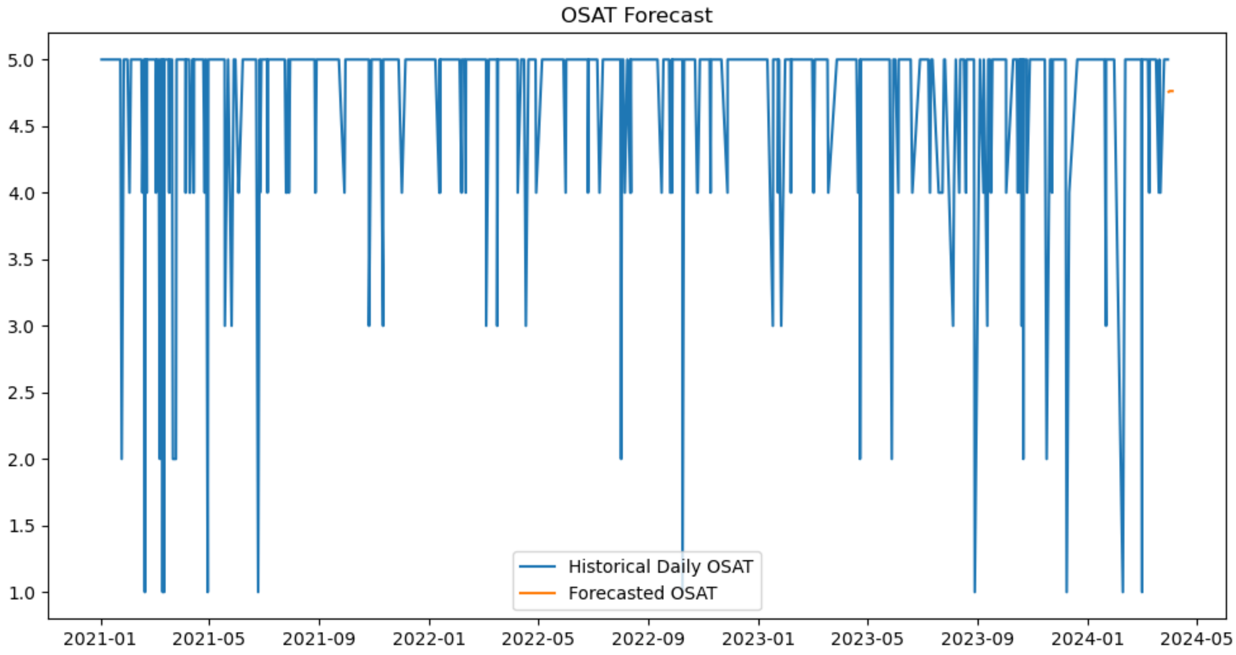
```

```

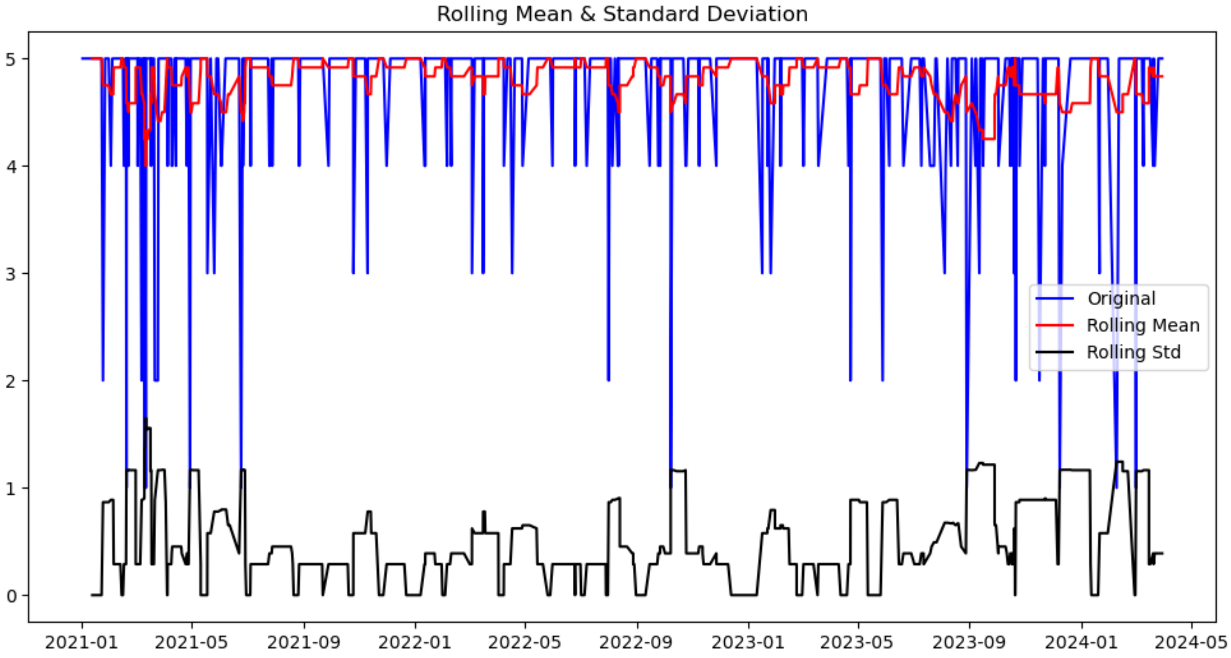
```
statsmodels.tsa.arima.model import ARIMA

lace p, d, q with your determined values




 = ARIMA(df2['OSAT'], order=(p, d, q))
ts = model.fit()
(results.summary())

ecasting
ast = results.get_forecast(steps=5)
ast_index = pd.date_range(start=df2.index[-1], periods=6, freq='D')[1:

tting results
igure(figsize=(12,6))
lot(df2.index, df2['OSAT'], label='Historical OSAT')
lot(forecast_index, forecast.predicted_mean, label='Forecasted OSAT')
ill_between(forecast_index, forecast.conf_int().iloc[:, 0], forecast.c
egend(loc='upper left')
itle('OSAT Forecast')
how()
```

```
                               SARIMAX Results
==========================================================================
=========
Dep. Variable:                   OSAT   No. Observations:
821
Model:                 ARIMA(1, 1, 1)   Log Likelihood
-792.556
Date:                Mon, 15 Apr 2024   AIC
1591.113
Time:                        15:43:54   BIC
1605.241
Sample:                             0   HQIC
1596.534
                                - 821
Covariance Type:                  opg
==========================================================================
=========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
---------
ar.L1         -0.0209      0.038     -0.546      0.585      -0.096
0.054
ma.L1         -0.9926      0.005   -204.955      0.000      -1.002
-0.983
```

```
sigma2          0.4025      0.006     63.853      0.000       0.390
0.415
===================================================================================
==============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
13368.48
Prob(Q):                              0.96   Prob(JB):
0.00
Heteroskedasticity (H):               1.06   Skew:
-4.05
Prob(H) (two-sided):                  0.61   Kurtosis:
21.04
===================================================================================
==============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).

/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:473: ValueWarning: A date index has been pr
ovided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:473: ValueWarning: A date index has been pr
ovided, but it is not monotonic and so will be ignored when e.g. fore
casting.
  self._init_dates(dates, freq)
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:473: ValueWarning: A date index has been pr
ovided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:473: ValueWarning: A date index has been pr
ovided, but it is not monotonic and so will be ignored when e.g. fore
casting.
  self._init_dates(dates, freq)
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:473: ValueWarning: A date index has been pr
ovided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:473: ValueWarning: A date index has been pr
ovided, but it is not monotonic and so will be ignored when e.g. fore
casting.
  self._init_dates(dates, freq)
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:836: ValueWarning: No supported index is av
```

```
els/tsa/base/tsa_model.py:836: ValueWarning: No supported index is av
ailable. Prediction results will be given with an integer index begin
ning at `start`.
  return get_prediction_index(
/Users/jannyvelazquez/anaconda3/lib/python3.11/site-packages/statsmod
els/tsa/base/tsa_model.py:836: FutureWarning: No supported index is a
vailable. In the next version, calling this method in a model without
a supported index will result in an exception.
  return get_prediction_index(
```


OSAT Forecast
Historical Daily OSAT
Forecasted OSAT


Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std