In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="dark")
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classificatio
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB, ComplementNB, BernoulliNB
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
import joblib
import warnings
from wordcloud import WordCloud

warnings.filterwarnings(action='ignore', category=Warning)
```

In [ ]:
```
df = pd.read_csv('./data/sentimentTwitter.csv', encoding='ISO-8859-1')
df.head()
```

Out[ ]:

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

In [ ]:
```
df.columns = ['Target', 'ID', 'Date', 'Flag', 'User', 'Text']
```

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

Out[ ]:

	Target	ID	Date	Flag	User	Text
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

In [ ]:
```
df.drop(['ID', 'Flag', 'User'],axis=1, inplace=True)
```

In [ ]:
```
df['Target'].unique()
```

Out[ ]:
```
array([0, 4], dtype=int64)
```

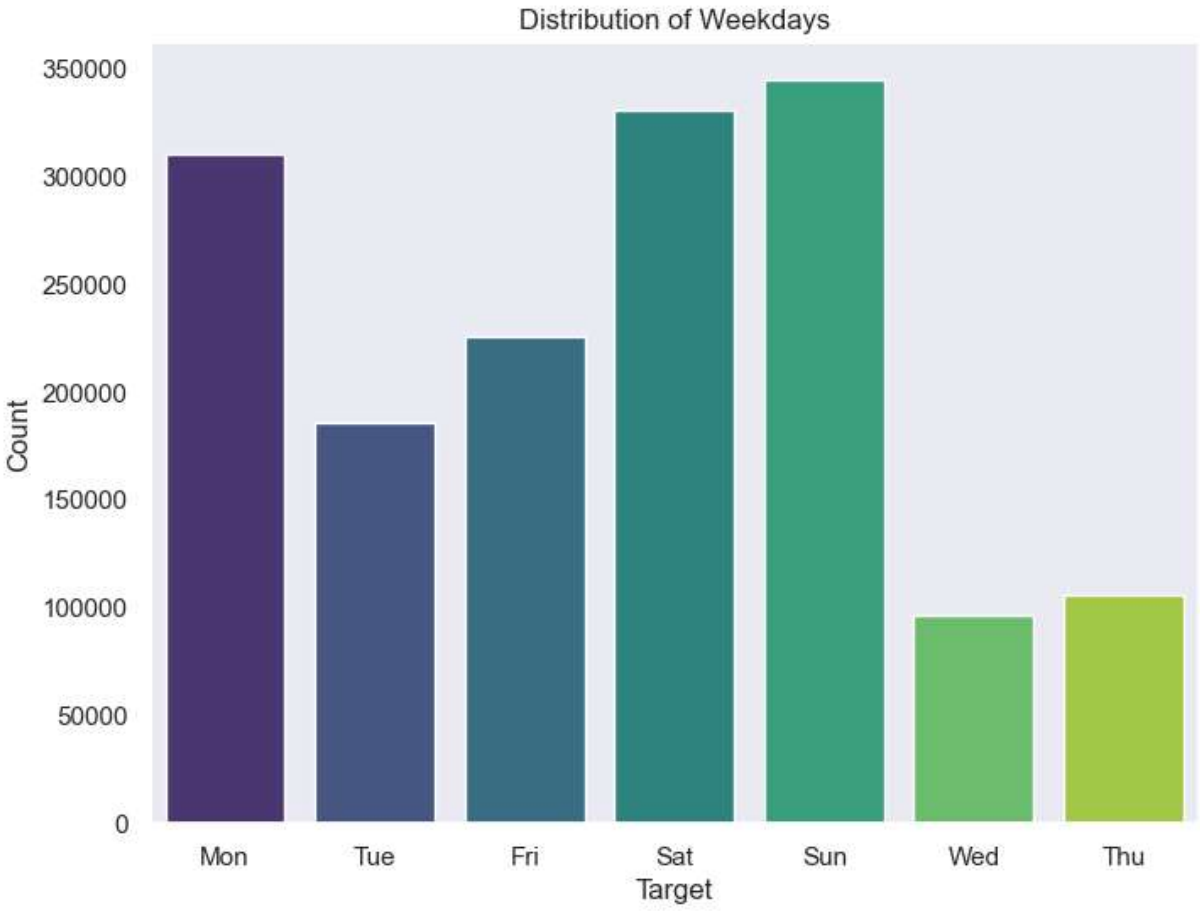
In [ ]:
```
df['Target'].replace({0:'Negative', 4: 'Positive'}, inplace=True)
```

In [ ]:
```
df['WeekDay'] = df['Date'].apply(lambda x: x[:3])
```

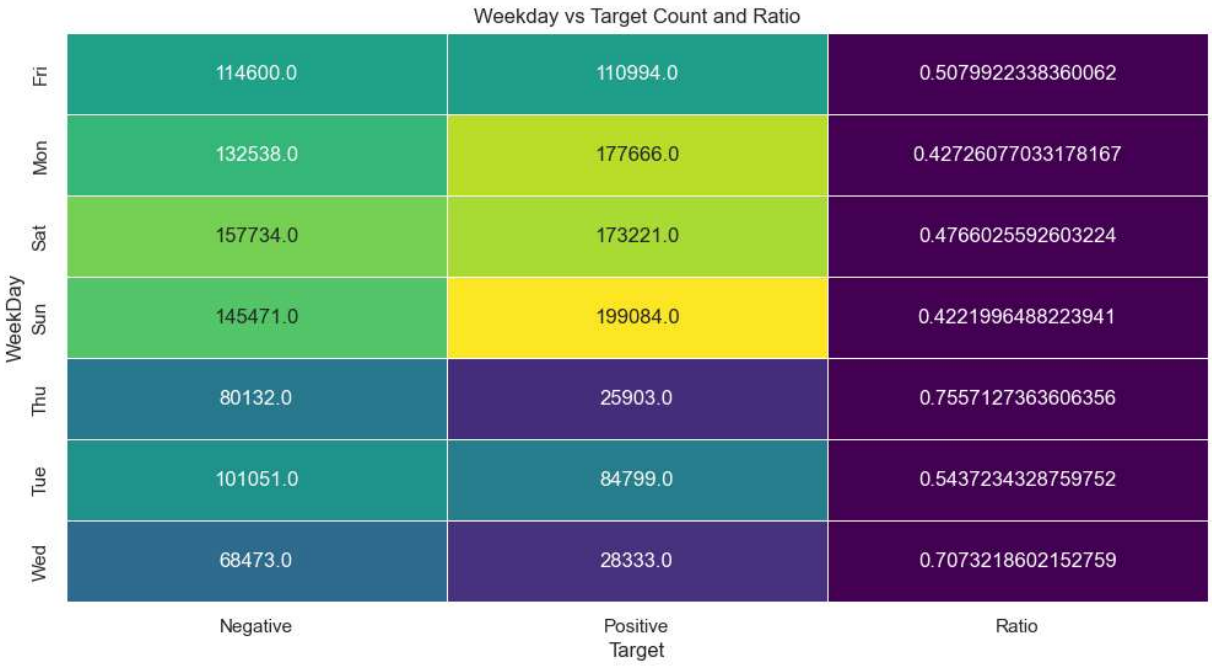
EDA

In [ ]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='Target', data=df, hue='Target', palette='viridis', dodge=False)
plt.title('Distribution of Target Variable')
plt.xlabel('Target')
plt.ylabel('Count')
plt.show();
```

In [ ]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='WeekDay', data=df, hue='WeekDay', palette='viridis', dodge=False)
plt.title('Distribution of Weekdays')
plt.xlabel('Target')
plt.ylabel('Count')
plt.show();
```



In [ ]:
```
pivot_table_weekday = df.groupby(['WeekDay', 'Target']).size().unstack(fill_value=0

pivot_table_weekday['Ratio'] = pivot_table_weekday['Negative'] / (pivot_table_weekd

plt.figure(figsize=(12, 6))
sns.heatmap(pivot_table_weekday, annot=True, fmt='1',
            cmap='viridis', linewidths=.5, cbar_kws={'label': 'Count'}, cbar=False)
plt.title('Weekday vs Target Count and Ratio')
plt.show()
```



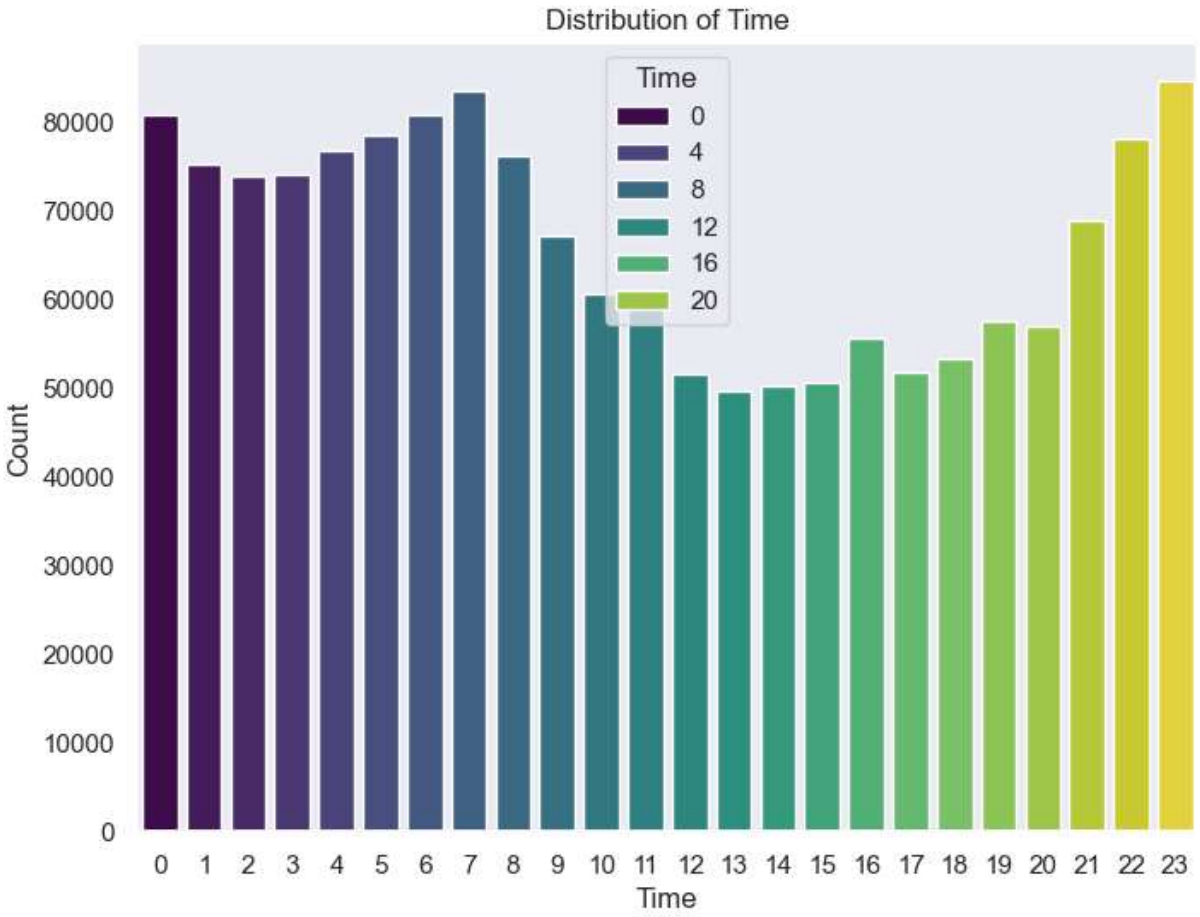
Analyzing the nature of human tweets based on the above chart involves making inferences about the sentiment and behavior of users on different days of the week. Here are some potential insights:

1. **Weekend Positivity:**
    - Weekends (Saturday and Sunday) generally exhibit a higher prevalence of positive tweets. Users may be more relaxed, enjoying leisure activities, or sharing positive experiences.
2. **Monday Optimism:**
    - Mondays show a notably higher prevalence of positive tweets compared to negative ones. This could be attributed to a more optimistic and positive outlook at the beginning of the workweek.
3. **Midweek Negativity:**
    - Wednesdays and Thursdays have a higher prevalence of negative tweets. This might be associated with midweek stress, work-related challenges, or a general dip in mood during these days.
4. **Friday Ambiguity:**
    - Fridays show a relatively balanced distribution between positive and negative tweets, with a slightly higher prevalence of negative tweets. This might indicate a mix of emotions as people approach the weekend.
5. **Tuesday Variability:**
    - Tuesdays show a relatively balanced distribution, with a slightly higher prevalence of negative tweets. This could suggest variability in users' sentiments on this day.

It is crucial to acknowledge that the interpretations provided are contingent upon the assumption that the sentiment conveyed in tweets serves as an accurate reflection of users' emotional states. However, it's essential to recognize the inherent limitations in drawing definitive conclusions about human nature solely from social media data. Various external factors, including current events, holidays, and cultural nuances, can significantly impact individuals' tweeting behavior. Therefore, while these observations offer insights into potential patterns, a comprehensive understanding of human behavior on social media would require a more nuanced analysis, considering a broader range of contextual elements and potential influences.

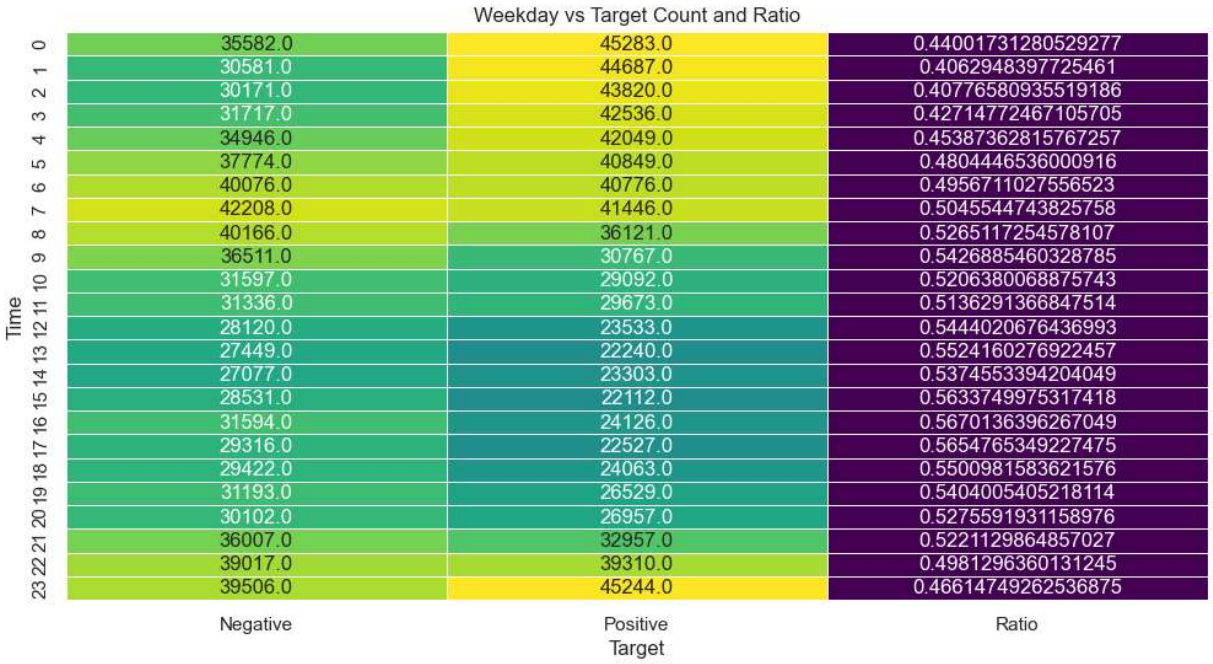
In [ ]:
```
df['Time'] = df['Date'].apply(lambda x: x[11:13]).astype('int64')
```

In [ ]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='Time', data=df, hue='Time', palette='viridis')
plt.title('Distribution of Time')
plt.xlabel('Time')
plt.ylabel('Count')
plt.show()
```



In [ ]:
```
pivot_table_time = df.groupby(['Time', 'Target']).size().unstack(fill_value=0)

pivot_table_time['Ratio'] = pivot_table_time['Negative'] / (pivot_table_time['Posit

plt.figure(figsize=(12, 6))
```

```
sns.heatmap(pivot_table_time, annot=True, fmt='1',
            cmap='viridis', linewidths=.5, cbar_kws={'label': 'Count'}, cbar=False)
plt.title('Weekday vs Target Count and Ratio')
plt.show()
```



The above chart presents the counts of negative and positive tweets for each hour of the day (in a 24-hour clock format), along with the calculated ratio of negative to positive tweets. Analyzing this data allows for insights into human behavior on social media at different times of the day:

1. **Late Night Positivity:**
    - In the early hours of the morning (around 0 to 5 hours), there is a higher prevalence of positive tweets compared to negative ones. This might suggest that users are more likely to share positive sentiments during the late night and early morning hours.
2. **Morning Consistency:**
    - During the morning hours (6 to 9 hours), the ratio of negative to positive tweets remains relatively balanced, indicating a consistent expression of sentiments. Users might be sharing a mix of positive and negative emotions as they start their day.
3. **Midday Positivity:**
    - From 10 to 15 hours, there is a shift towards a higher prevalence of positive tweets. This could be indicative of a more optimistic or positive outlook during the midday hours.
4. **Afternoon Fluctuations:**
    - In the afternoon (16 to 19 hours), the ratio varies, with some hours showing a higher prevalence of negative tweets, while others have a higher prevalence of

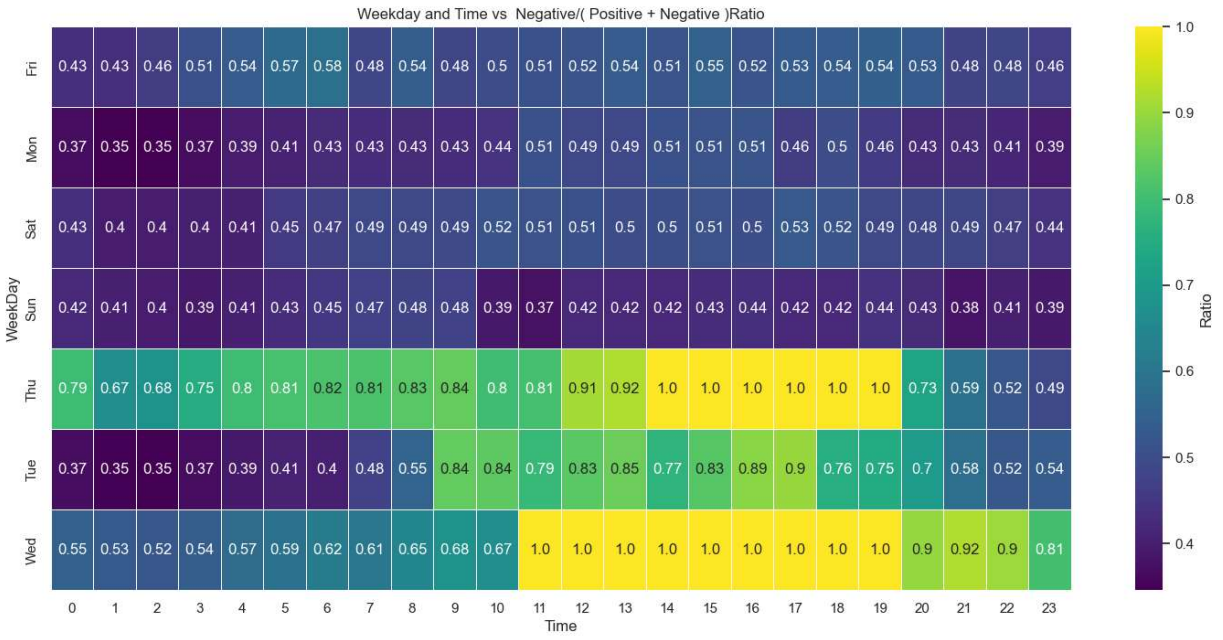
positive tweets. This suggests potential fluctuations in user sentiments during the afternoon.

5. **Evening Mixed Sentiments:**

   - During the evening hours (20 to 23 hours), the ratio fluctuates, and the sentiment expressed in tweets appears to be more varied. Users may share a mix of positive and negative sentiments during the evening.

It's important to note that while the ratio of negative to positive tweets provides an indication of sentiment balance, it doesn't necessarily capture the intensity or specific content of the emotions expressed. Additionally, external factors such as regional, cultural, or event-related influences can significantly impact tweeting behavior.

In [ ]:
```
pivot_table = df.groupby(['WeekDay', 'Time', 'Target']).size().unstack(fill_value=0
pivot_table['Ratio'] = pivot_table['Negative'] / (pivot_table['Negative'] + pivot_t

plt.figure(figsize=(18, 8))
sns.heatmap(pivot_table['Ratio'].unstack(), annot=True, fmt='1.2', cmap='viridis',
plt.title('Weekday and Time vs  Negative/( Positive + Negative )Ratio')
plt.show()
```



Interpreting the data on tweet counts, sentiment, and the ratio of negative to positive tweets based on 'WeekDay' and 'Time' can offer some insights into potential trends in human behavior on social media:

1. **Cyclical Patterns:**

   - The cyclical patterns throughout the day suggest that human behavior on social media, as reflected in tweets, may follow a rhythmic cycle. Different times of the day may be associated with distinct communication patterns and emotional expressions.

2. **Morning Negativity:**

   - The observed increase in the ratio of negative to positive tweets during the morning hours on certain days (Thursday and Friday) could indicate that people may start their day with a higher level of negative sentiment. This might be influenced by factors such as waking up to news or the stresses associated with the beginning of the workweek.

3. **Evening Fluctuations:**

   - The notable increase in the ratio during the evening hours, especially on Thursday and Friday, suggests that these periods might be associated with heightened negative sentiments. This could be influenced by a variety of factors, including fatigue, end-of-workweek stress, or reaction to news events.

4. **Weekend Consistency:**

   - The more consistent distribution of ratios during the weekend might imply that people maintain a relatively stable sentiment pattern during these days. However, the specific reasons for this consistency would require further investigation.

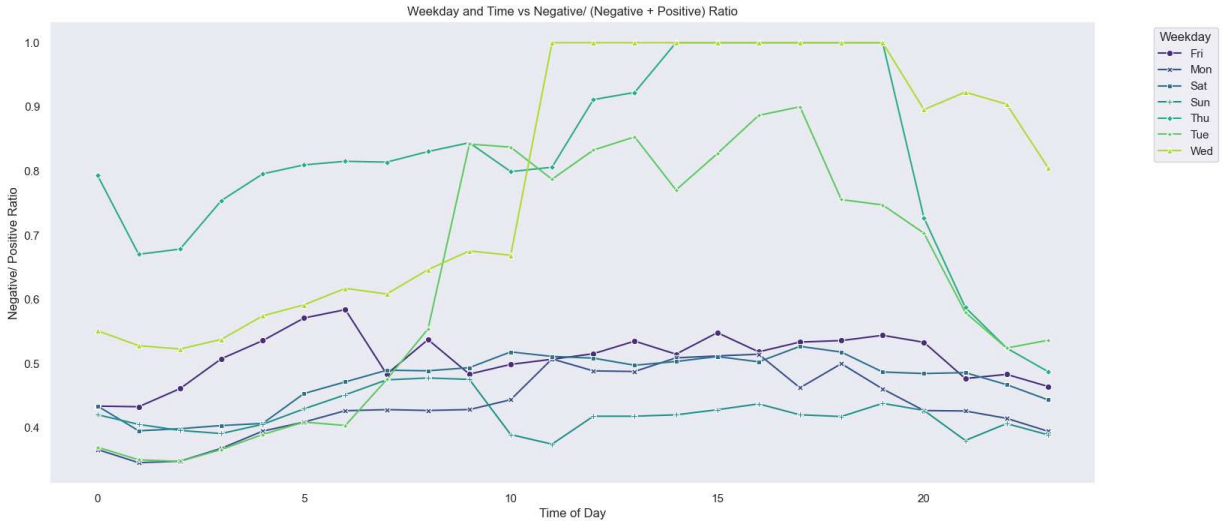
5. **Thursday Evening Spike:**

   - The distinctive spike in negative sentiment on Thursday evenings may be indicative of a specific trend or event that consistently affects Twitter activity during that time. This could be related to cultural or social phenomena, events, or media programming.

6. **Variability Across Days:**

   - The observed variability in sentiment patterns across weekdays suggests that external factors, such as work-related stress or news events, may influence how people express themselves on Twitter.

In [ ]:
```
plt.figure(figsize=(18, 8))
sns.lineplot(data=pivot_table['Ratio'].unstack().T, dashes=False, markers=True, pal
plt.xlabel('Time of Day')
plt.ylabel('Negative/ Positive Ratio')
plt.title('Weekday and Time vs Negative/ (Negative + Positive) Ratio')
plt.legend(title='Weekday', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



The depicted graph reveals intriguing patterns in the temporal dynamics of sentiment among Twitter users on different weekdays. Here are key observations based on the chart:

1. **Thursday (Thu):**
   - The sentiment on Thursday initiates with a negative tone, steadily rising to its peak. However, after the 20th hour, there is a noticeable decline in the negative sentiment ratio. This suggests a shift towards more positive tweets during the later hours of Thursday.
2. **Wednesday (Wed):**
   - On Wednesday, the chart shows a distinctive pattern, starting with a negative sentiment that dramatically rises by 10 hours and maintains a consistently high level until the 20th hour. Subsequently, there is a sharp decline, indicating a significant shift towards a less negative or potentially positive sentiment.
3. **Sunday (Sun):**
   - Sunday commences with a positive sentiment, but there is a rapid transition towards negative sentiment. However, this negative trend does not persist, as the ratio drops back to a neutral level and does not ascend beyond 0.9. This suggests a dynamic fluctuation in sentiment throughout the day on Sundays.

Data Preprocessing

In [ ]:
```
df.drop('Date', axis=1,inplace=True)
```

In [ ]:
```
df_feature = df.drop(['WeekDay','Time','Target'], axis=1)
df_target = df['Target']
X_train, X_test, y_train, y_test = train_test_split(df_feature, df_target)
```

TF-IDF

In [ ]:
```
vectorizer = TfidfVectorizer(max_features=2000)
X_train_final = vectorizer.fit_transform(X_train['Text'])
X_test_final = vectorizer.transform(X_test['Text'])
```

Model Training

In [ ]:
```
lrModel = LogisticRegression()
lrModel.fit(X_train_final, y_train)
print('Train Score of Logistic Regression Model: ')
print(lrModel.score(X_train_final, y_train))
print('Test Score of Logistic Regression Model: ')
print(lrModel.score(X_test_final, y_test))
```

```
Train Score of Logistic Regression Model:
0.7817081514234595
Test Score of Logistic Regression Model:
0.78018
```

In [ ]:
```
mnbModel = MultinomialNB()
mnbModel.fit(X_train_final, y_train)
print('Train Score of MultinomialNB Model: ')
print(mnbModel.score(X_train_final, y_train))
print('Test Score of MultinomialNB Model: ')
print(mnbModel.score(X_test_final, y_test))
```

```
Train Score of MultinomialNB Model:
0.7608906340755284
Test Score of MultinomialNB Model:
0.7595625
```

In [ ]:
```
bnbModel = BernoulliNB()
bnbModel.fit(X_train_final, y_train)
print('Train Score of BernoulliNB Model: ')
print(bnbModel.score(X_train_final, y_train))
print('Test Score of BernoulliNB Model: ')
print(bnbModel.score(X_test_final, y_test))
```

```
Train Score of BernoulliNB Model:
0.7619756349796958
Test Score of BernoulliNB Model:
0.76067
```

In [ ]:
```
CnbModel = ComplementNB()
CnbModel.fit(X_train_final, y_train)
print('Train Score of ComplementNB Model: ')
print(CnbModel.score(X_train_final, y_train))
print('Test Score of ComplementNB Model: ')
print(CnbModel.score(X_test_final, y_test))
```

```
Train Score of ComplementNB Model:
0.7608906340755284
Test Score of ComplementNB Model:
0.7595925
```

In [ ]:
```
sgdModel = SGDClassifier()
sgdModel.fit(X_train_final, y_train)
print('Train Score of SGDClassifier Model: ')
print(sgdModel.score(X_train_final, y_train))
print('Test Score of SGDClassifier Model: ')
print(sgdModel.score(X_test_final, y_test))
```

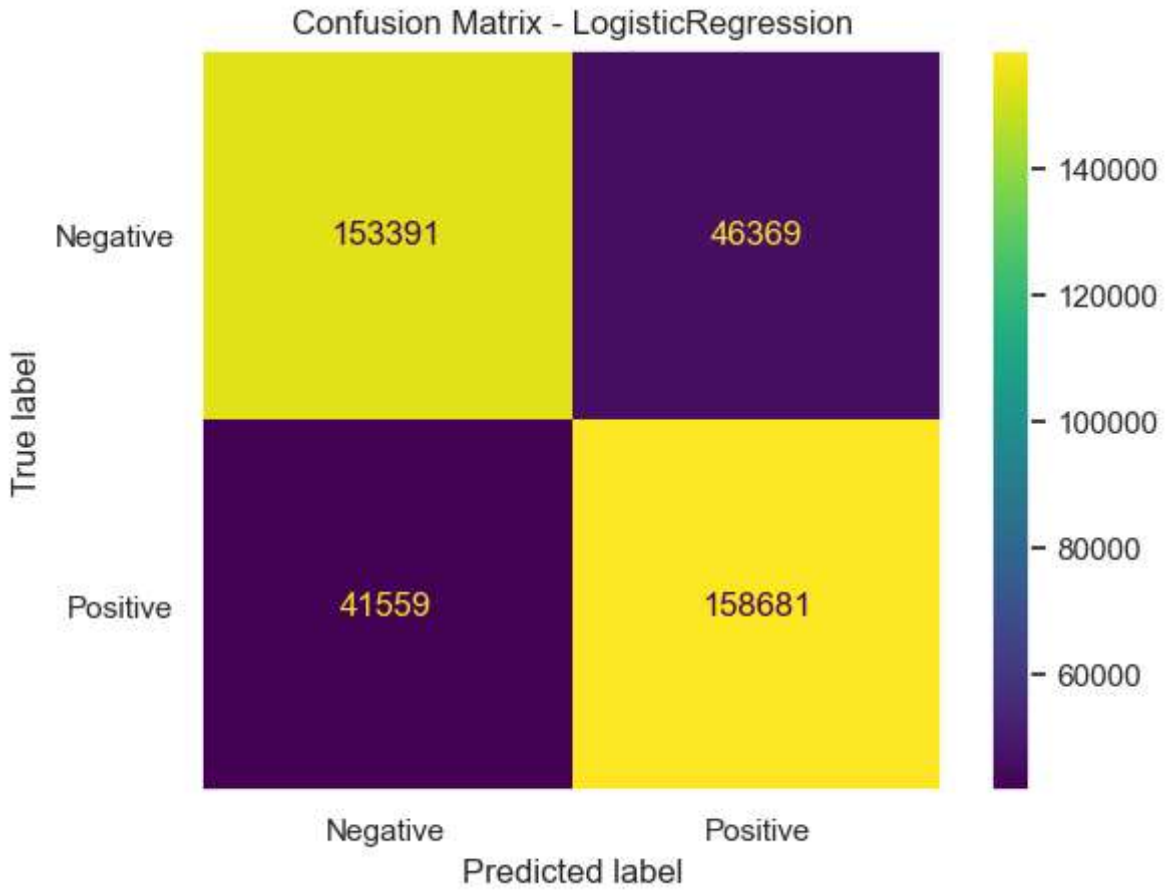
```
Train Score of SGDClassifier Model:
0.7767364806137338
Test Score of SGDClassifier Model:
0.775585
```

Display the Performance for each model

In [ ]:
```
def evaluate_model(model, X_test, y_test):
    model_name = type(model).__name__
    # Predictions on test set
    y_pred = model.predict(X_test)

    print(f"Classification Report for {model_name}:\n")
    print(classification_report(y_test, y_pred))

    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes
    disp.plot(values_format='d')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.show()
```

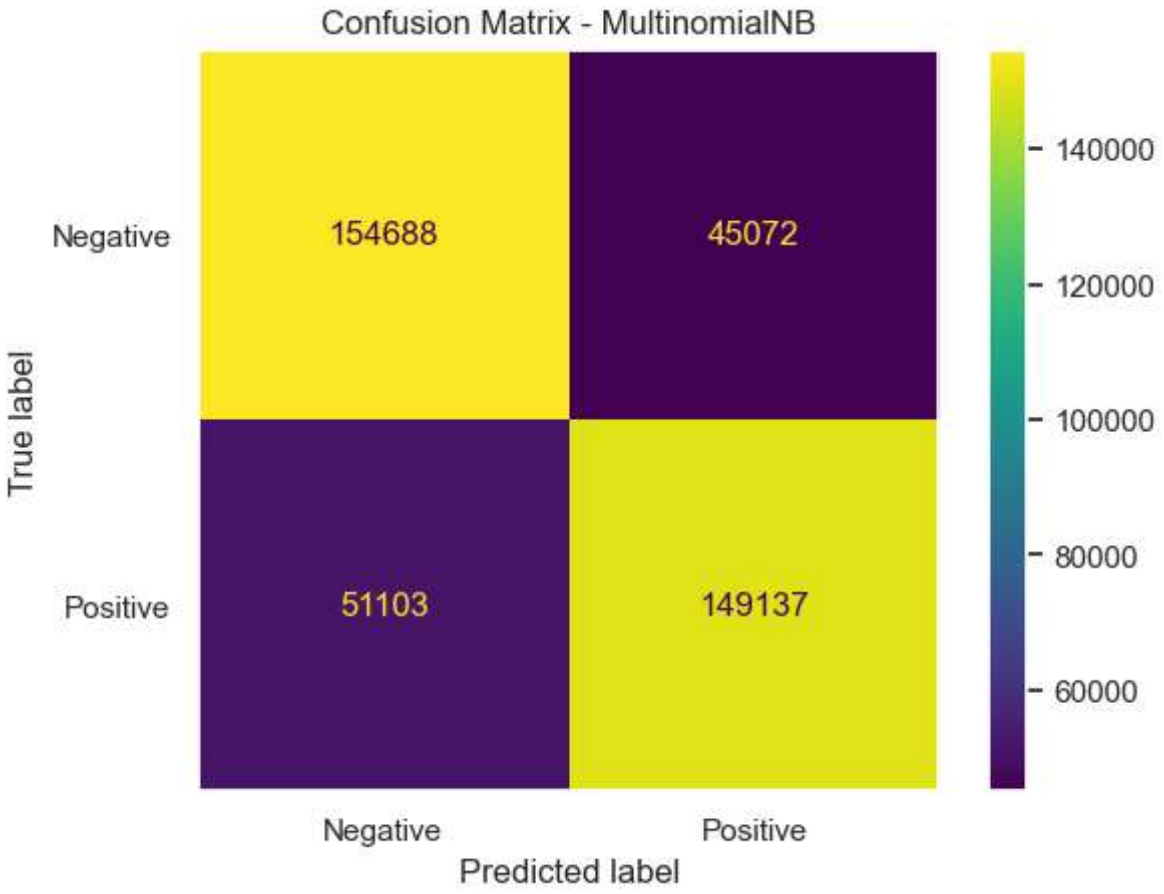
In [ ]:
```
evaluate_model(lrModel,X_test=X_test_final, y_test=y_test)
```

```
Classification Report for LogisticRegression:

              precision    recall  f1-score   support

    Negative       0.79      0.77      0.78    199760
    Positive       0.77      0.79      0.78    200240

    accuracy                           0.78    400000
   macro avg       0.78      0.78      0.78    400000
weighted avg       0.78      0.78      0.78    400000
```



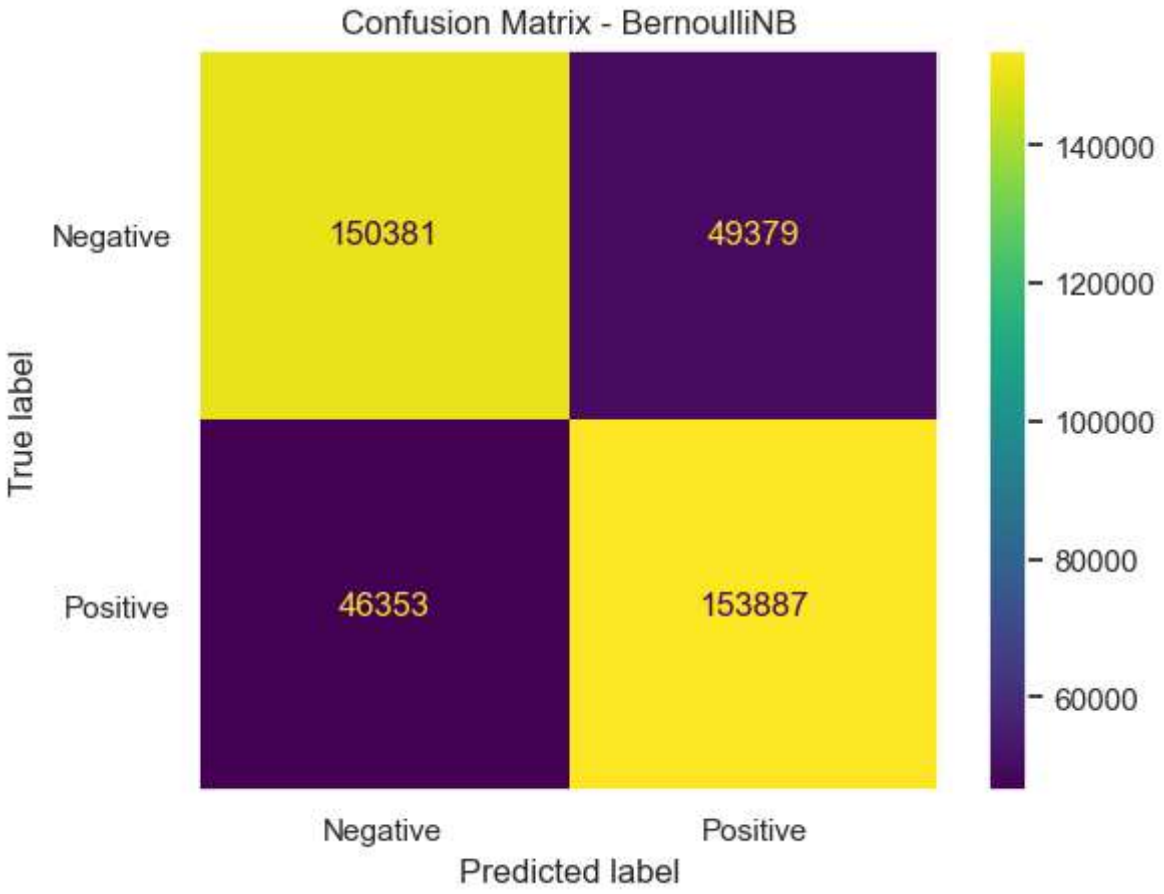
In [ ]:
```
evaluate_model(mnbModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for MultinomialNB:

              precision    recall  f1-score   support

    Negative       0.75      0.77      0.76    199760
    Positive       0.77      0.74      0.76    200240

    accuracy                           0.76    400000
   macro avg       0.76      0.76      0.76    400000
weighted avg       0.76      0.76      0.76    400000
```



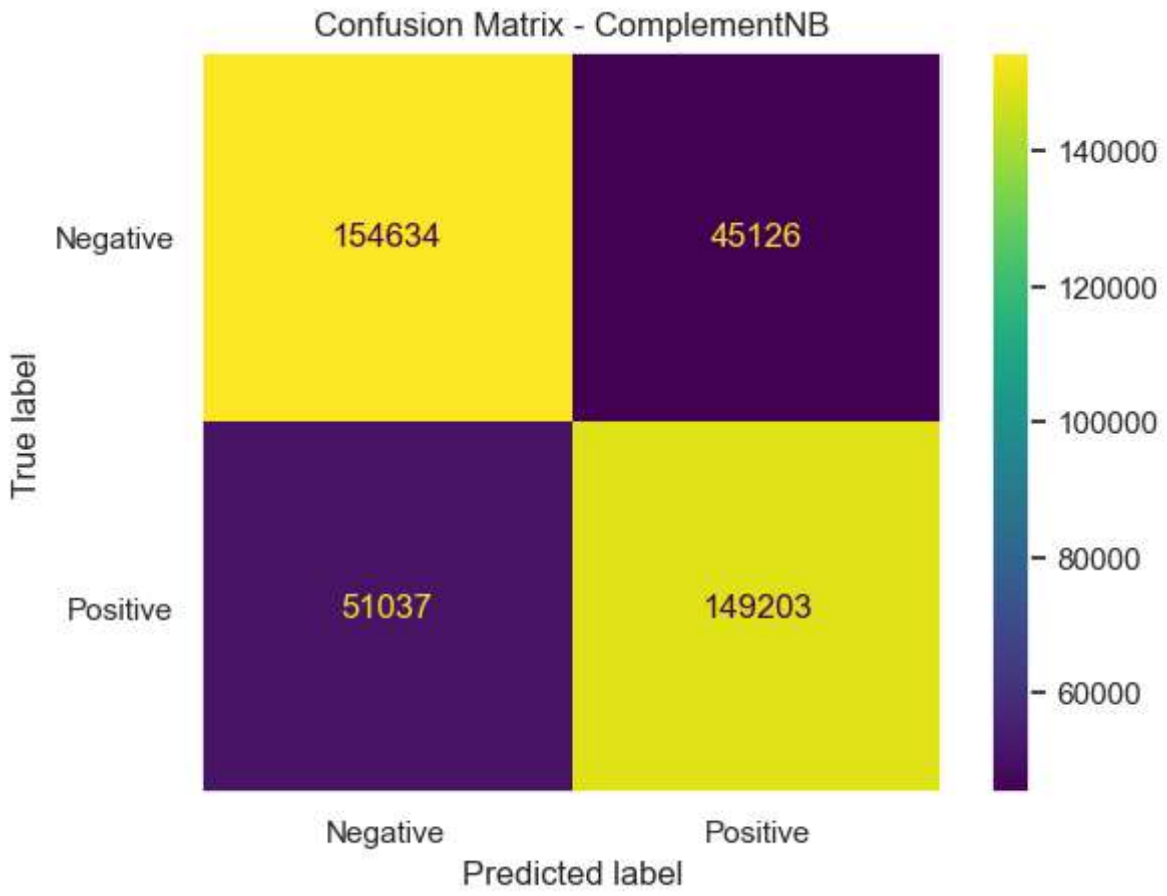
In [ ]:
```
evaluate_model(bnbModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for BernoulliNB:

              precision    recall  f1-score   support

    Negative       0.76      0.75      0.76    199760
    Positive       0.76      0.77      0.76    200240

    accuracy                           0.76    400000
   macro avg       0.76      0.76      0.76    400000
weighted avg       0.76      0.76      0.76    400000
```



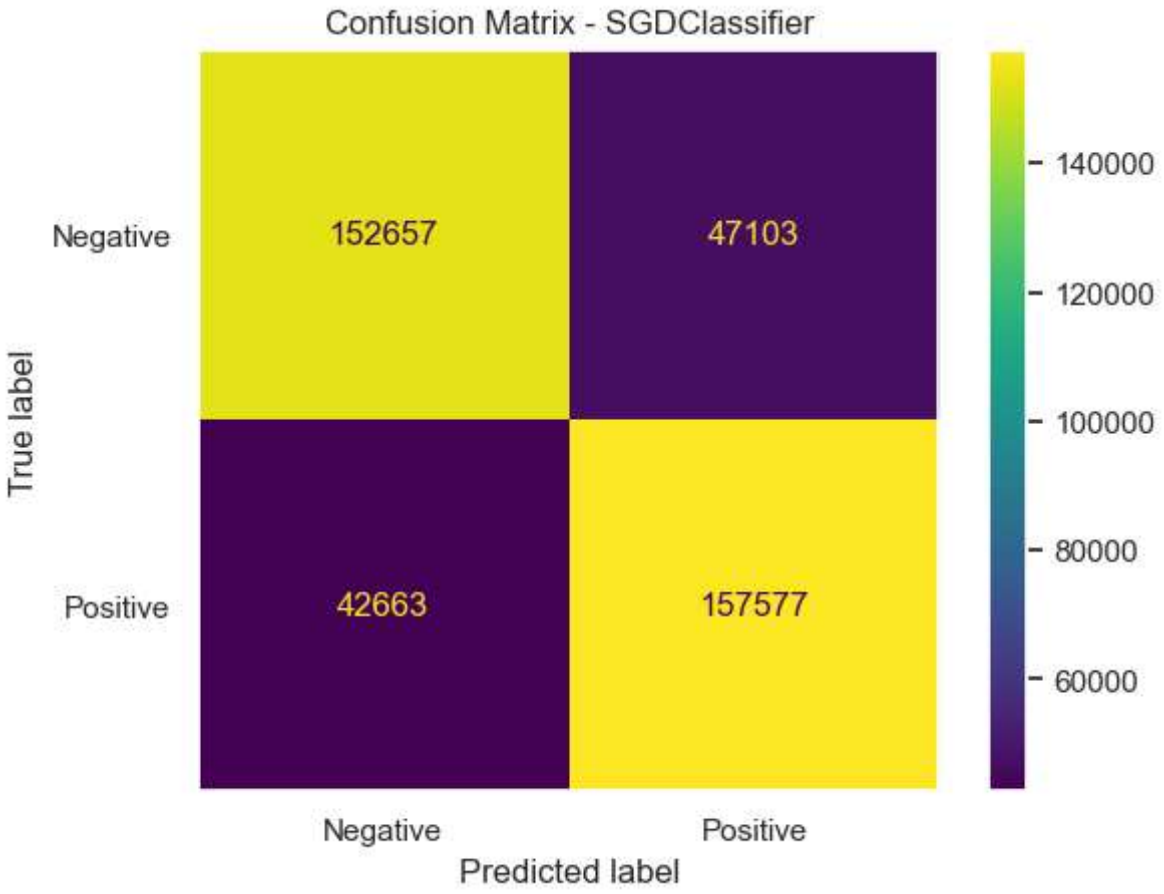
In [ ]:
```
evaluate_model(CnbModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for ComplementNB:

              precision    recall  f1-score   support

    Negative       0.75      0.77      0.76    199760
    Positive       0.77      0.75      0.76    200240

    accuracy                           0.76    400000
   macro avg       0.76      0.76      0.76    400000
weighted avg       0.76      0.76      0.76    400000
```



In [ ]:
```
evaluate_model(sgdModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for SGDClassifier:

              precision    recall  f1-score   support

    Negative       0.78      0.76      0.77    199760
    Positive       0.77      0.79      0.78    200240

    accuracy                           0.78    400000
   macro avg       0.78      0.78      0.78    400000
weighted avg       0.78      0.78      0.78    400000
```

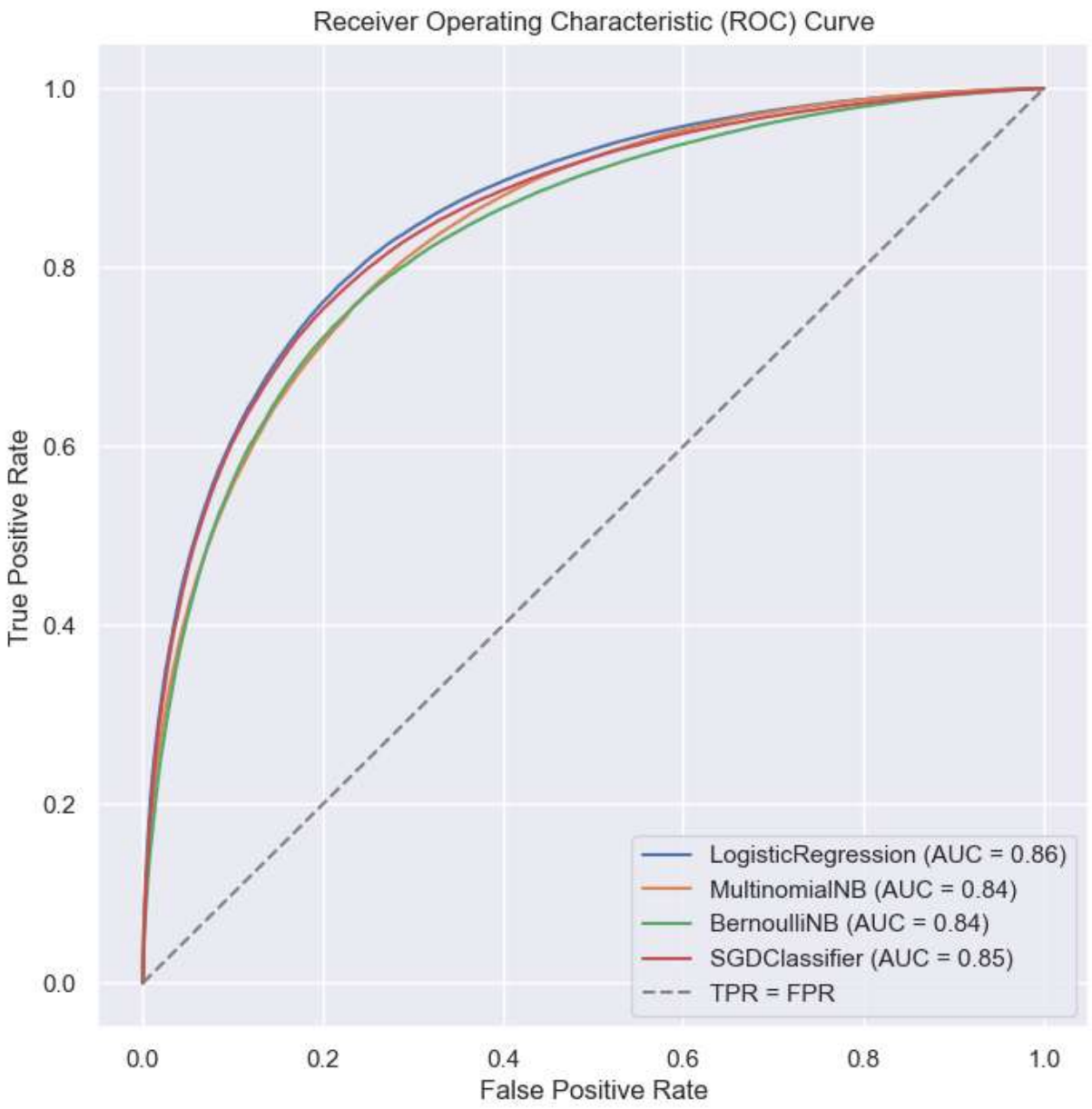


ROC Curve Vizulaization

In [ ]:
```
y_train_mapped = y_train.map({'Negative': 0, 'Positive': 1})
y_test_mapped = y_test.map({'Negative': 0, 'Positive': 1})
```

In [ ]:
```
models = [lrModel, mnbModel, bnbModel, sgdModel]

plt.figure(figsize=(8, 8))

for model in models:
    try:
        y_prob = model.predict_proba(X_test_final)[:, 1]
    except AttributeError:
        y_prob = model.decision_function(X_test_final)

    fpr, tpr, _ = roc_curve(y_test_mapped, y_prob)
    roc_auc = auc(fpr, tpr)

    model_name = type(model).__name__

    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='TPR = FPR')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



In [ ]:
```
score_list = []
for model in models:
    score_list.append(model.score(X_test_final, y_test))


plt.figure(figsize=(10, 6))
ax = sns.barplot(x=models, y=score_list, palette="viridis")
ax.set(xlabel="Models", ylabel="Scores", title="Performance of Models")
plt.show()
```

In [ ]:
```
joblib.dump(lrModel, './models/bestSentiment_v.pkl')
```

Out[ ]:
```
['./models/bestSentiment_v.pkl']
```

CountVectorizer

In [ ]:
```
vectorizer = CountVectorizer(stop_words='english')
X_train_final = vectorizer.fit_transform(X_train['Text'])
X_test_final = vectorizer.transform(X_test['Text'])
```

In [ ]:
```
lrModel = LogisticRegression()
lrModel.fit(X_train_final, y_train)
print('Train Score of Logistic Regression Model: ')
print(lrModel.score(X_train_final, y_train))
print('Test Score of Logistic Regression Model: ')
print(lrModel.score(X_test_final, y_test))
```

```
Train Score of Logistic Regression Model:
0.8073340061116717
Test Score of Logistic Regression Model:
0.7764525
```

In [ ]:
```
mnbModel = MultinomialNB()
mnbModel.fit(X_train_final, y_train)
print('Train Score of MultinomialNB Model: ')
print(mnbModel.score(X_train_final, y_train))
print('Test Score of MultinomialNB Model: ')
print(mnbModel.score(X_test_final, y_test))
```

```
Train Score of MultinomialNB Model:
0.8294598578832149
Test Score of MultinomialNB Model:
0.767995
```

In [ ]:
```
bnbModel = BernoulliNB()
bnbModel.fit(X_train_final, y_train)
print('Train Score of BernoulliNB Model: ')
print(bnbModel.score(X_train_final, y_train))
print('Test Score of BernoulliNB Model: ')
print(bnbModel.score(X_test_final, y_test))
```

```
Train Score of BernoulliNB Model:
0.829940691617243
Test Score of BernoulliNB Model:
0.7678375
```

In [ ]:
```
CnbModel = ComplementNB()
CnbModel.fit(X_train_final, y_train)
print('Train Score of ComplementNB Model: ')
print(CnbModel.score(X_train_final, y_train))
print('Test Score of ComplementNB Model: ')
print(CnbModel.score(X_test_final, y_test))
```

```
Train Score of ComplementNB Model:
0.8294965245804371
Test Score of ComplementNB Model:
0.768005
```

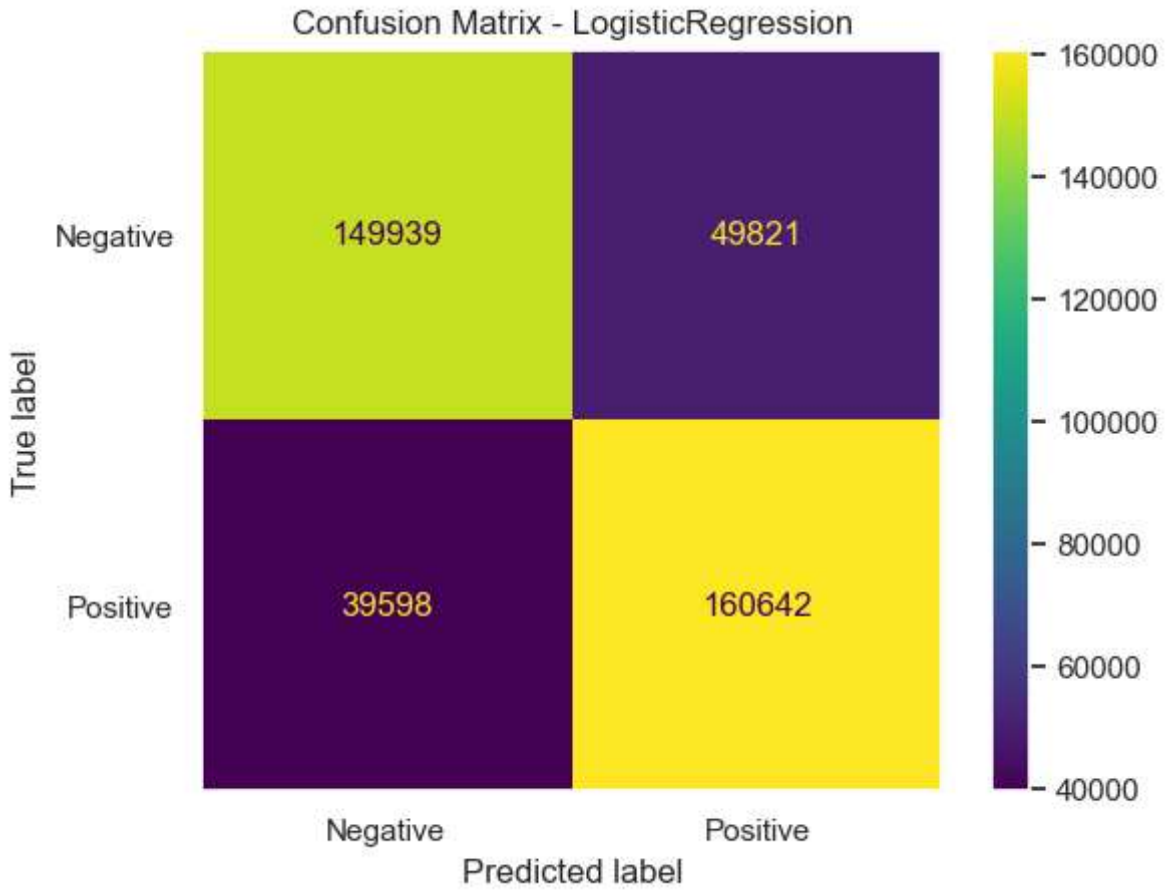
In [ ]:
```
sgdModel = SGDClassifier()
sgdModel.fit(X_train_final, y_train)
print('Train Score of SGDClassifier Model: ')
print(sgdModel.score(X_train_final, y_train))
print('Test Score of SGDClassifier Model: ')
print(sgdModel.score(X_test_final, y_test))
```

```
Train Score of SGDClassifier Model:
0.7718573098810916
Test Score of SGDClassifier Model:
0.7666175
```

Display the Performance for each model

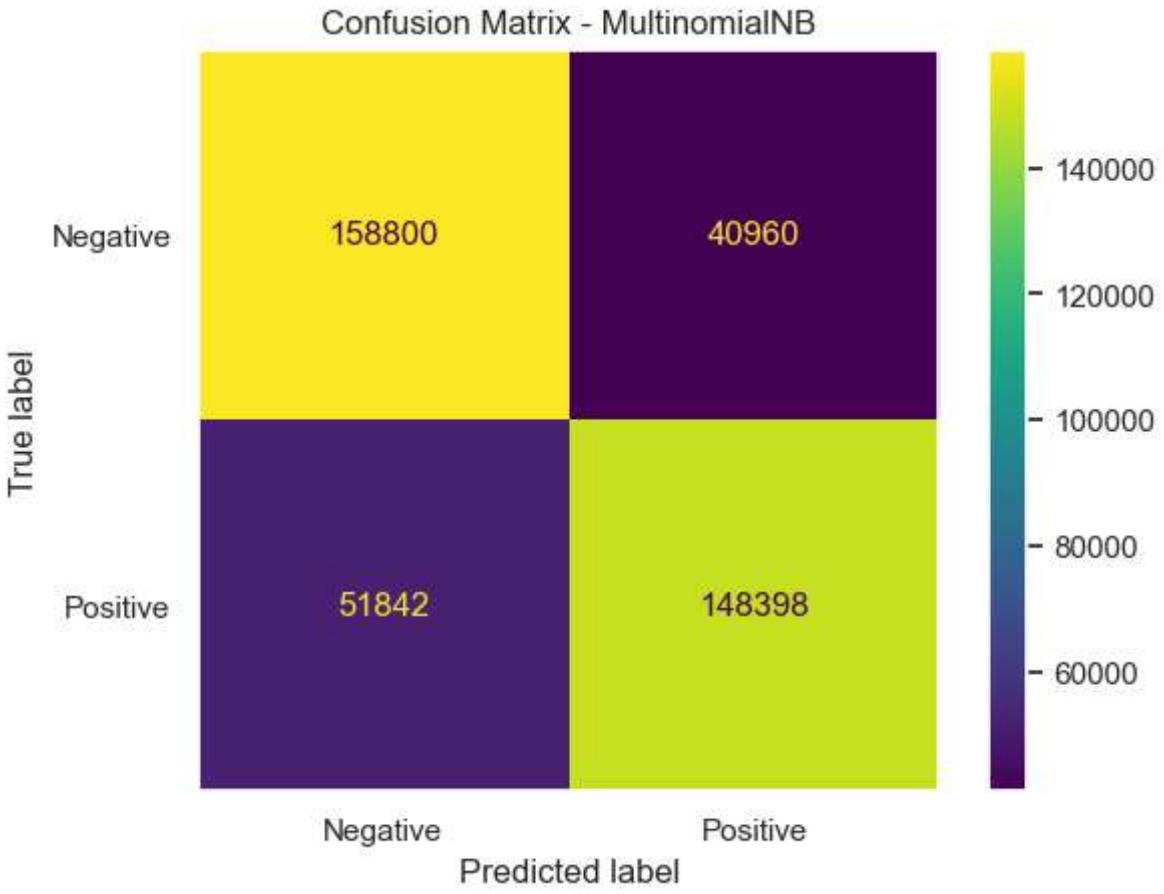
In [ ]:
```
evaluate_model(lrModel,X_test=X_test_final, y_test=y_test)
```

```
Classification Report for LogisticRegression:

              precision    recall  f1-score   support

    Negative       0.79      0.75      0.77    199760
    Positive       0.76      0.80      0.78    200240

    accuracy                           0.78    400000
   macro avg       0.78      0.78      0.78    400000
weighted avg       0.78      0.78      0.78    400000
```



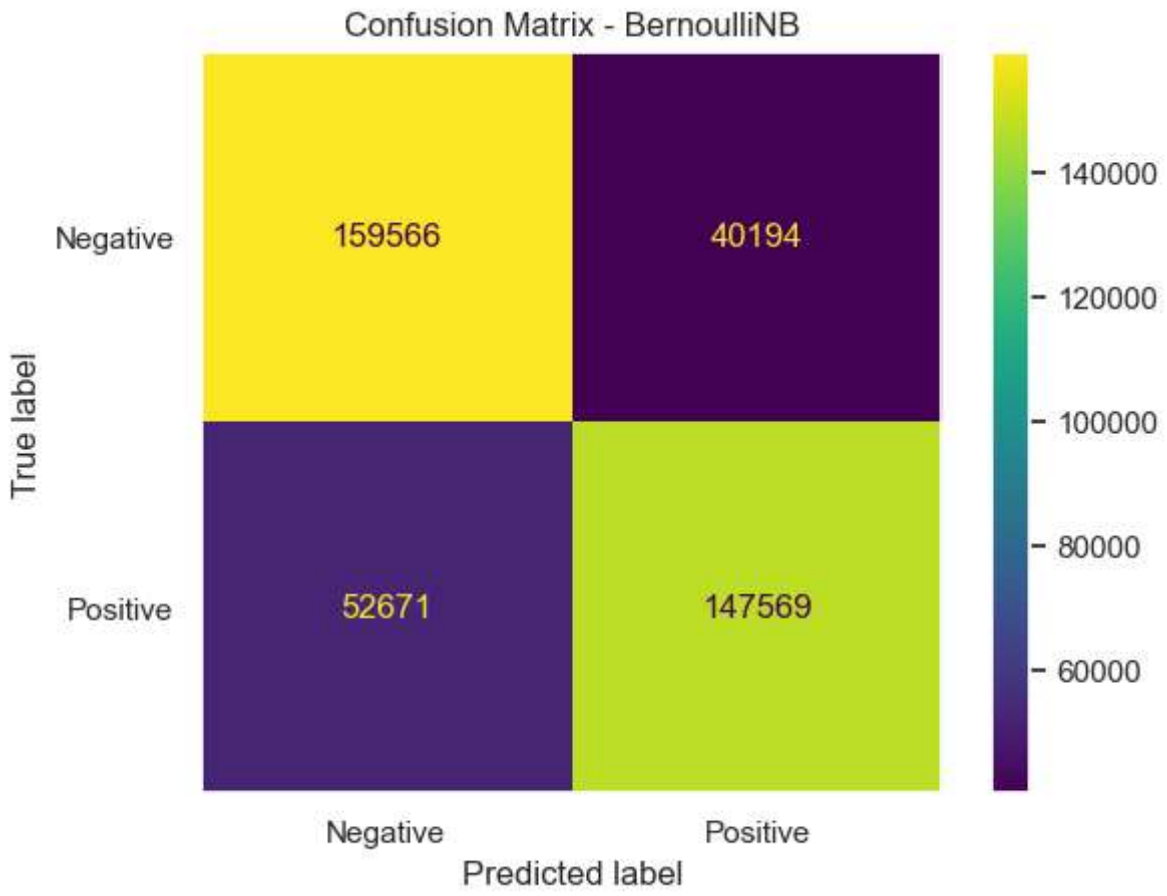
In [ ]:
```
evaluate_model(mnbModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for MultinomialNB:

              precision    recall  f1-score   support

    Negative       0.75      0.79      0.77    199760
    Positive       0.78      0.74      0.76    200240

    accuracy                           0.77    400000
   macro avg       0.77      0.77      0.77    400000
weighted avg       0.77      0.77      0.77    400000
```



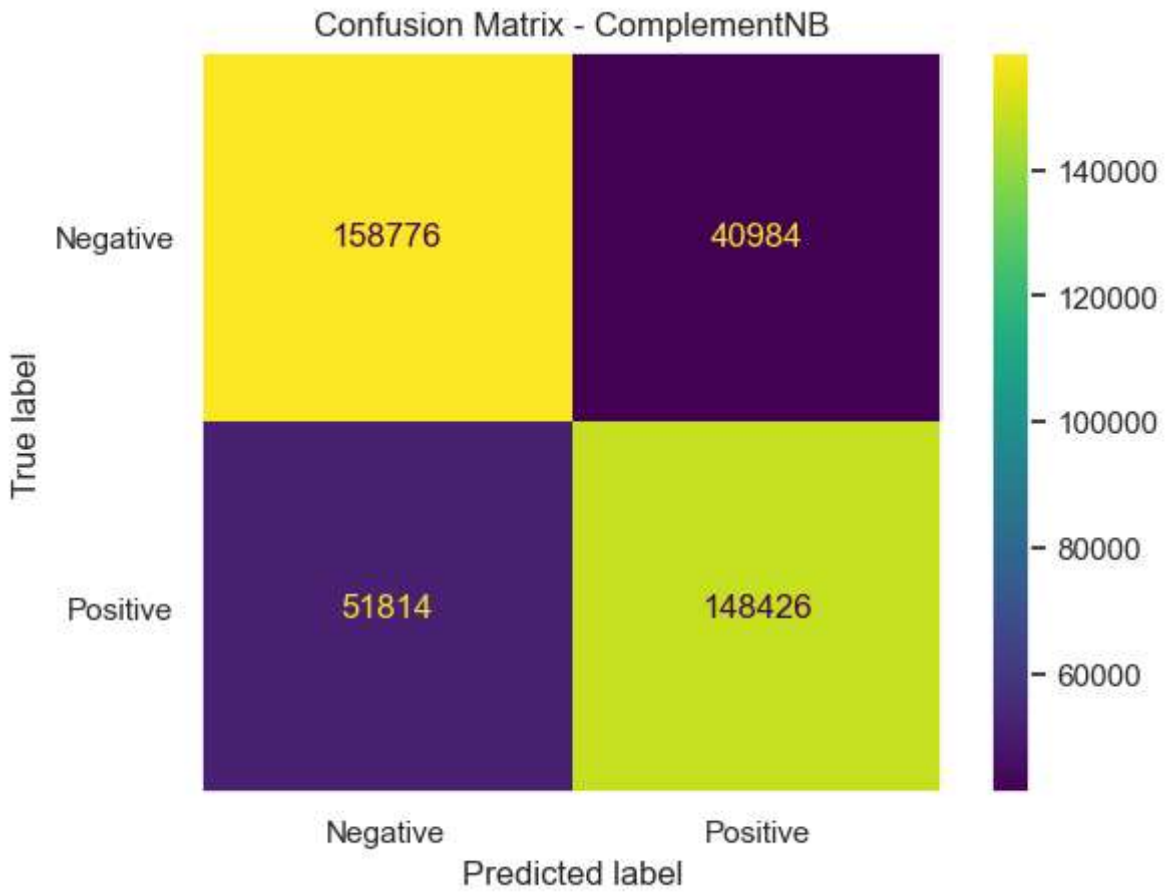
In [ ]:
```
evaluate_model(bnbModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for BernoulliNB:

              precision    recall  f1-score   support

    Negative       0.75      0.80      0.77    199760
    Positive       0.79      0.74      0.76    200240

    accuracy                           0.77    400000
   macro avg       0.77      0.77      0.77    400000
weighted avg       0.77      0.77      0.77    400000
```



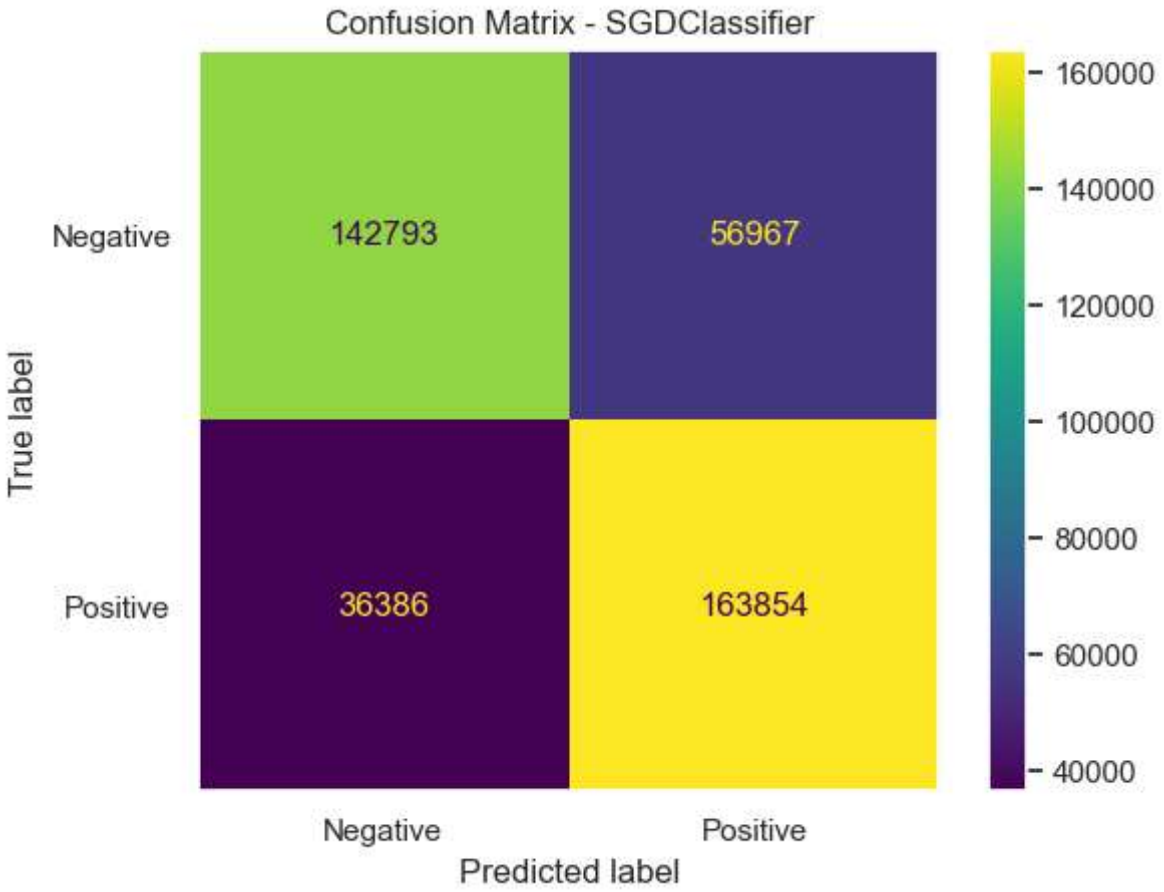
In [ ]:
```
evaluate_model(CnbModel, X_test=X_test_final, y_test=y_test)
```

```
Classification Report for ComplementNB:

              precision    recall  f1-score   support

    Negative       0.75      0.79      0.77    199760
    Positive       0.78      0.74      0.76    200240

    accuracy                           0.77    400000
   macro avg       0.77      0.77      0.77    400000
weighted avg       0.77      0.77      0.77    400000
```



In [ ]:
```
evaluate_model(sgdModel, X_test=X_test_final, y_test=y_test)
```

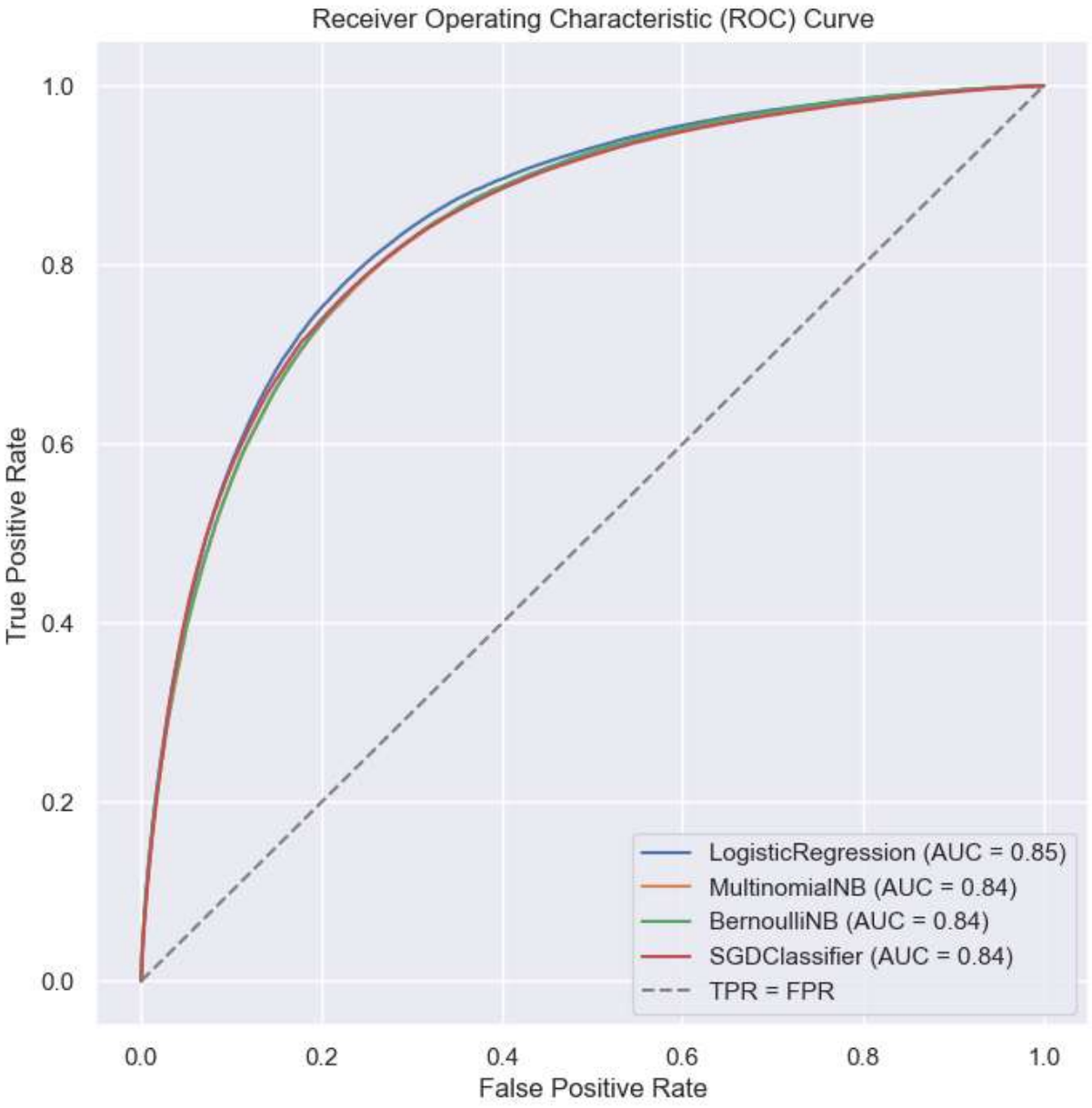
```
Classification Report for SGDClassifier:

              precision    recall  f1-score   support

    Negative       0.80      0.71      0.75    199760
    Positive       0.74      0.82      0.78    200240

    accuracy                           0.77    400000
   macro avg       0.77      0.77      0.77    400000
weighted avg       0.77      0.77      0.77    400000
```



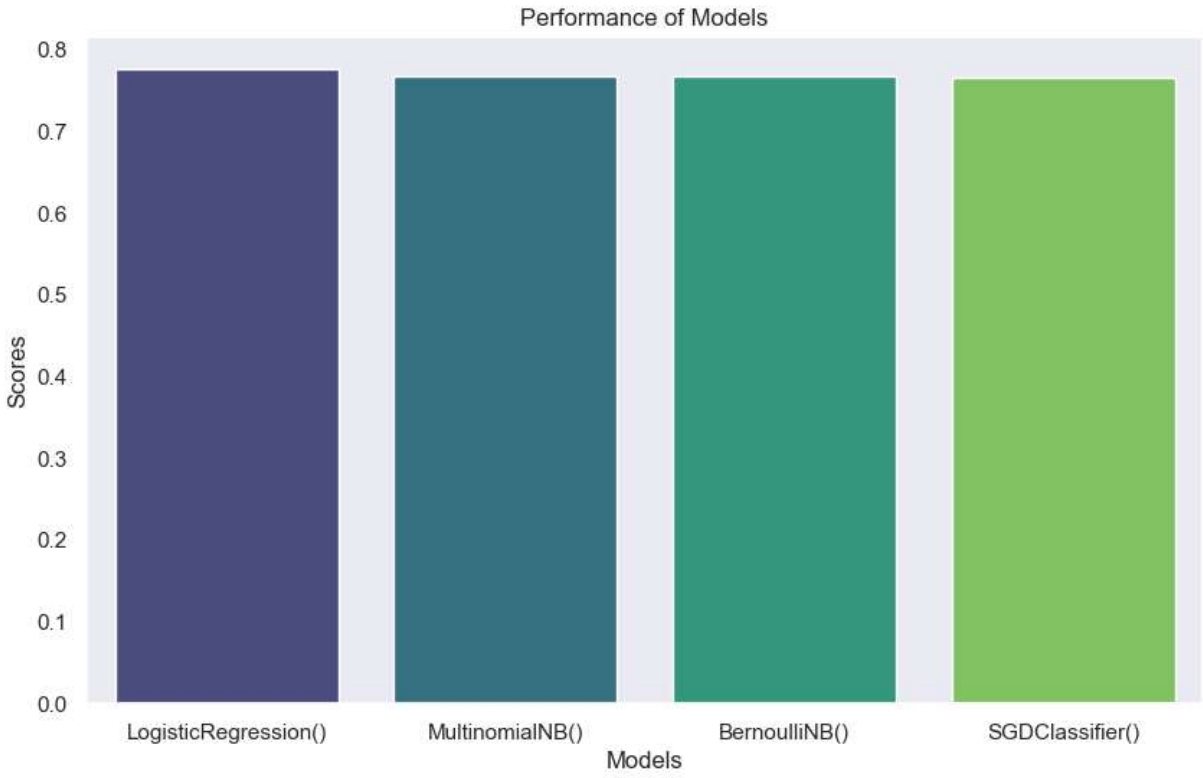
ROC Curve Vizulaization

In [ ]:
```
models = [lrModel, mnbModel, bnbModel, sgdModel]

plt.figure(figsize=(8, 8))

for model in models:
    try:
        y_prob = model.predict_proba(X_test_final)[:, 1]
    except AttributeError:
        y_prob = model.decision_function(X_test_final)

    fpr, tpr, _ = roc_curve(y_test_mapped, y_prob)
    roc_auc = auc(fpr, tpr)

    model_name = type(model).__name__

    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='TPR = FPR')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
```

```
plt.grid(True)
plt.show()
```



In [ ]:
```
score_list = []
for model in models:
    score_list.append(model.score(X_test_final, y_test))


plt.figure(figsize=(10, 6))
ax = sns.barplot(x=models, y=score_list, palette="viridis")
ax.set(xlabel="Models", ylabel="Scores", title="Performance of Models")
plt.show()
```



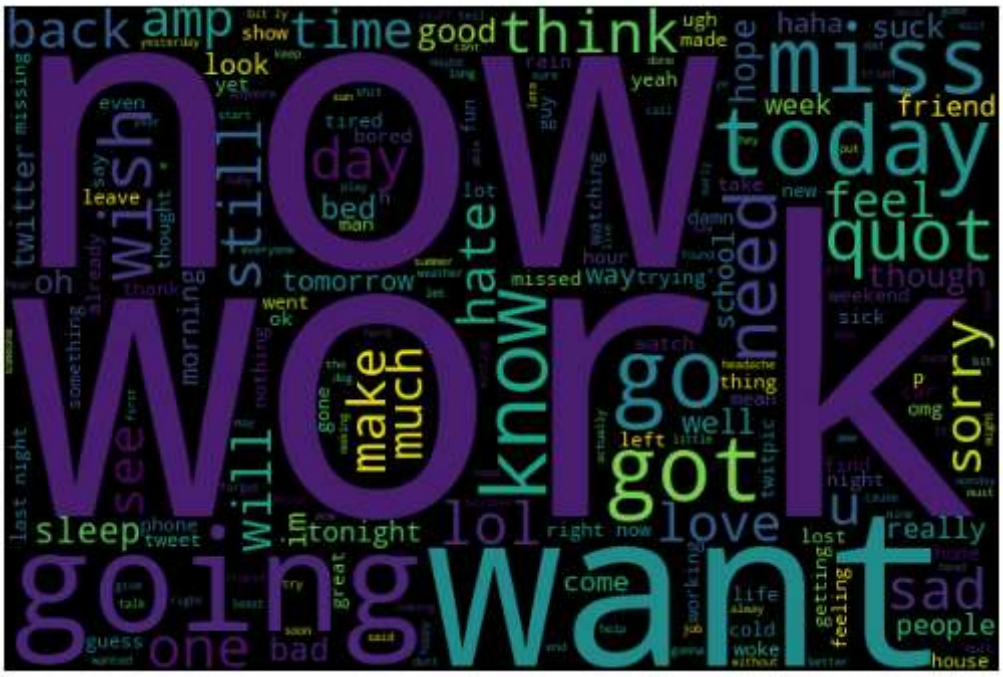
In [ ]:
```
joblib.dump(lrModel, './models/bestSentiment_c.pkl')
```

Out[ ]:
```
['./models/bestSentiment_c.pkl']
```

In [ ]:
```
def visualize(label):
    words=''
    for msg in df[df['Target'] == label]['Text']:
        msg = msg.lower()
        words += msg + ' '
    wordcloud = WordCloud(width=600, height=400).generate(words)
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()
```

Visualize Important words

In [ ]:
```
visualize('Negative')
```



In [ ]:
```
visualize('Positive')
```