Bianco Blanco - Sentiment Analysis

Chosen Algorithm: Multinomial Naive Bayes (MNB) and Text Mining
The MNB classifier is well-suited for text data due to its efficiency with high-dimensional datasets and its assumption of feature independence. This makes it ideal for handling the complexities of word counts and TF-IDF features derived from text.
Model Evaluation Metrics
Confusion Matrix: [[0, 145], [0, 1330]]

This matrix indicates a significant bias toward predicting positives, which could impact user trust if negative reviews are misclassified.

Accuracy: 90.17%
Precision: 90.17%
Recall: 100%
F1 Score: 94.96%

Evaluation Visuals and Their Importance:

- ROC Curve: The ROC curve illustrates the trade-off between the true positive rate and false positive rate at various threshold settings. It is critical for evaluating the model's discriminative ability, particularly useful in understanding how well the model distinguishes between classes under different thresholds.
- Precision-Recall Curve: This curve highlights the trade-off between precision and recall, with a high area under the curve (0.97) demonstrating that the model maintains high precision even as recall increases. This is crucial in contexts where minimizing false positives is essential to maintain user trust and product integrity.

Python Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import (confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score, roc_curve, auc,
                             precision_recall_curve, average_precision_score)
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
# Load the data
data = pd.read_csv('/Users/biancoblanco/Downloads/amazon_reviews.csv')

# Set display options
pd.set_option('display.max_columns', None)  # None means show all columns
pd.set_option('display.max_colwidth', 75)

# Basic Data Overview
print("Data Shape:", data.shape)
print("Row Count:", len(data))
print("First 5 Rows of Data:")
print(data.head(5))

# Data preprocessing
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove punctuation and numbers
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    # Tokenize
    tokens = word_tokenize(text)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    filtered_words = [word for word in tokens if not word in stop_words]
    return ' '.join(filtered_words)

data['reviewText'] = data['reviewText'].astype(str).apply(preprocess_text)

# Exploratory Data Analysis (EDA)
# Distribution of Ratings
plt.figure(figsize=(8, 6))
data['overall'].value_counts().sort_index().plot(kind='bar')
plt.title('Distribution of Ratings')
plt.xlabel('Ratings')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# Text vectorization using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['reviewText'])
y = data['overall'].apply(lambda x: 1 if x > 3 else 0)  # Binary classification
of sentiment

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

```
# Train Multinomial Naive Bayes classifier
model = MultinomialNB()
model.fit(X_train, y_train)

# Predictions and evaluation
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Confusion Matrix:\n", cm)
print("Accuracy: {:.2f}%".format(acc * 100))
print("Precision: {:.2f}%".format(precision * 100))
print("Recall: {:.2f}%".format(recall * 100))
print("F1 Score: {:.2f}".format(f1))

# Identify and display incorrectly classified reviews
incorrect_indices = np.where(y_test != y_pred)[0]
incorrect_reviews = data.iloc[incorrect_indices, :]
print("First 5 Incorrectly Classified Reviews:")
print(incorrect_reviews[['reviewText', 'overall']].head(5))

# Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 8))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %
roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

precision, recall, _ = precision_recall_curve(y_test,
model.predict_proba(X_test)[:, 1])
average_precision = average_precision_score(y_test,
model.predict_proba(X_test)[:, 1])

plt.figure(figsize=(8, 8))
plt.step(recall, precision, where='post', color='b', alpha=0.7,
label='Precision-Recall curve (area = %0.2f)' % average_precision)
plt.xlabel('Recall')
```

```
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall curve')
plt.legend(loc="upper right")
plt.show()
```

Bianco Sentiment Analysis:

Optimal Metric: Precision
Given the skewed distribution of ratings towards positive sentiments and the high cost of false positives in this context, optimizing for precision is appropriate. High precision ensures that positive classifications are reliable, which is crucial since consumers might disregard negative reviews that are incorrectly classified as positive.

Algorithm Performance:
While the MNB model demonstrates strong initial performance with all metrics and visual evaluations showing promising results, there is room for improvement. The model's tendency to over-predict positive sentiments could be mitigated by:

- Enhancing Text Preprocessing: Incorporating a more comprehensive list of stop-words could refine the model's focus on sentiment-indicative words.
- Advanced Sentiment Analysis Techniques: Implementing libraries like VADER, which provides contextual awareness by tagging parts of speech, could enhance accuracy, especially since adverbs often carry significant sentiment weight.

Actionable Steps

- Implement a Cleaning Pre-process Step: Further clean and preprocess the data to better capture nuances in sentiment.
- Explore Advanced Models: Consider testing other algorithms or integrating ensemble methods that might offer a more balanced approach to handling both positive and negative reviews effectively.

By adopting these strategies, we aim to enhance the model’s precision and overall reliability, ensuring that Amazon can more accurately reflect product sentiments and assist consumers in making informed purchasing decisions.