In [1]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
import matplotlib.pyplot as plt
import seaborn as sns

# NLTK Downloads
nltk.download("stopwords")
nltk.download("wordnet")
```

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

Out[1]:

```
True
```

In [2]:

```
# 1. Data Collection - Load the dataset
data = pd.read_csv("synthetic_sentiment_dataset.csv")
```

In [3]:

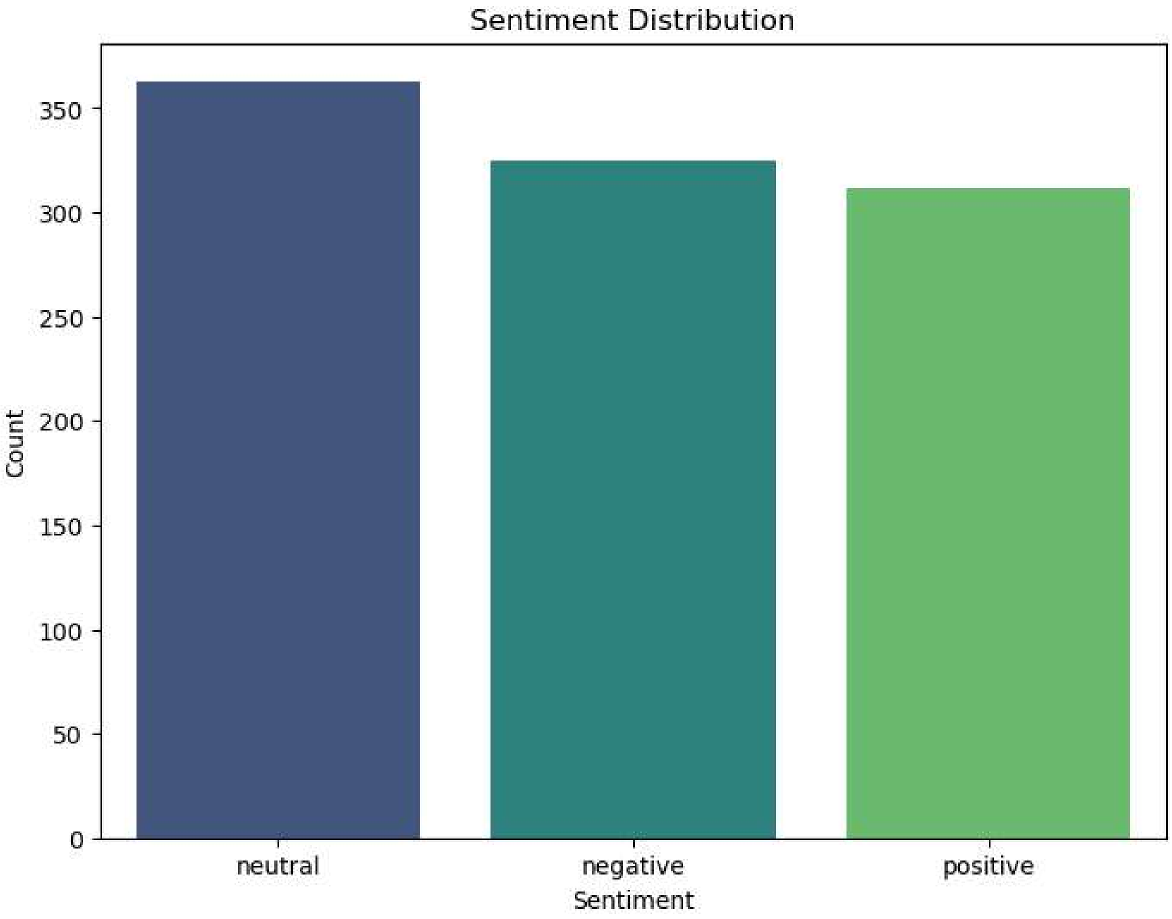
```
# 2. Data Preprocessing Techniques
def clean_text(text):
    """Remove special characters, convert to lowercase."""
    text = text.lower()
    return re.sub(r"[^a-z\s]", "", text)

def tokenize_text(text):
    """Tokenize the text."""
    return text.split()

def remove_stopwords(tokens):
    """Remove common stopwords."""
    return [word for word in tokens if word not in stopwords.words("english")]

def lemmatize_tokens(tokens):
    """Lemmatize tokens to their base form."""
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(word) for word in tokens]

# Apply preprocessing functions
data["Cleaned_Text"] = data["Review"].apply(clean_text)
data["Tokens"] = data["Cleaned_Text"].apply(tokenize_text)
data["Tokens_No_Stopwords"] = data["Tokens"].apply(remove_stopwords)
data["Lemmatized_Tokens"] = data["Tokens_No_Stopwords"].apply(lemmatize_tokens
data["Processed_Text"] = data["Lemmatized_Tokens"].apply(lambda tokens: " ".jo
```

In [4]:

```
# 3. Visualizing Dataset and Key Stats
print("Full Dataset with Preprocessing Steps:")
print(data)
```

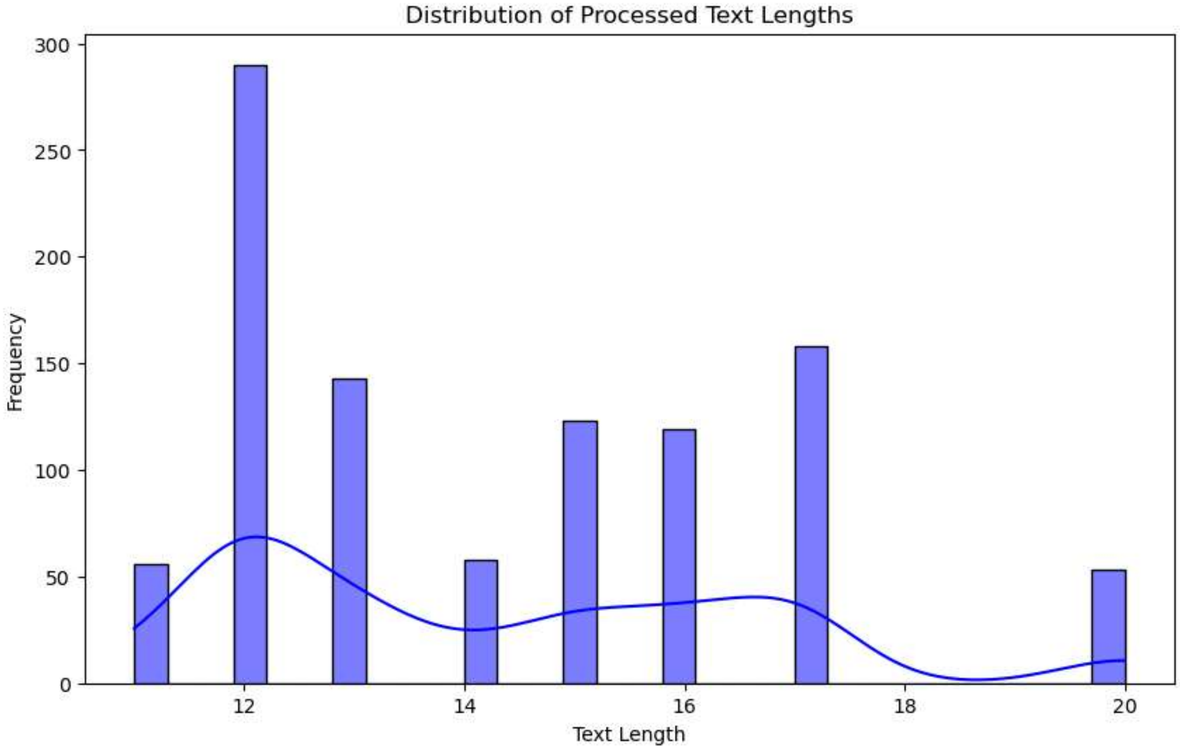
```
Full Dataset with Preprocessing Steps:
                            Review Sentiment                  Cleaned_Text  \
0            This product is fine.   neutral          this product is fine   
1            This product is hate.  negative          this product is hate   
2            This product is love.  positive          this product is love   
3            This product is okay.   neutral          this product is okay   
4            This product is poor.  negative          this product is poor   
..                             ...       ...                           ...   
995     This product is fantastic.  positive     this product is fantastic   
996  This product is satisfactory.   neutral  this product is satisfactory   
997      This product is mediocre.   neutral      this product is mediocre   
998         This product is great.  positive         this product is great   
999         This product is great.  positive         this product is great   

                                Tokens      Tokens_No_Stopwords  \
0            [this, product, is, fine]          [product, fine]   
1            [this, product, is, hate]          [product, hate]   
2            [this, product, is, love]          [product, love]   
3            [this, product, is, okay]          [product, okay]   
4            [this, product, is, poor]          [product, poor]   
..                                 ...                      ...   
995     [this, product, is, fantastic]     [product, fantastic]   
996  [this, product, is, satisfactory]  [product, satisfactory]   
997      [this, product, is, mediocre]      [product, mediocre]   
998         [this, product, is, great]         [product, great]   
999         [this, product, is, great]         [product, great]   

           Lemmatized_Tokens        Processed_Text  
0            [product, fine]          product fine  
1            [product, hate]          product hate  
2            [product, love]          product love  
3            [product, okay]          product okay  
4            [product, poor]          product poor  
..                       ...                   ...  
995     [product, fantastic]     product fantastic  
996  [product, satisfactory]  product satisfactory  
997      [product, mediocre]      product mediocre  
998         [product, great]         product great  
999         [product, great]         product great  

[1000 rows x 7 columns]
```

In [5]:
```
# Sentiment distribution visualization
sentiment_counts = data["Sentiment"].value_counts()
plt.figure(figsize=(8, 6))
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values, palette="viri
plt.title("Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()
```



In [6]:

```
# Text length distribution visualization
data["Text_Length"] = data["Processed_Text"].apply(len)
plt.figure(figsize=(10, 6))
sns.histplot(data["Text_Length"], kde=True, bins=30, color="blue")
plt.title("Distribution of Processed Text Lengths")
plt.xlabel("Text Length")
plt.ylabel("Frequency")
plt.show()
```



In [8]:

```
# 4. Split Data into Training and Testing Sets
X = data["Processed_Text"]  # Features (processed reviews)
y = data["Sentiment"]  # Labels (sentiment)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Convert text into numerical features using TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=500)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [10]:

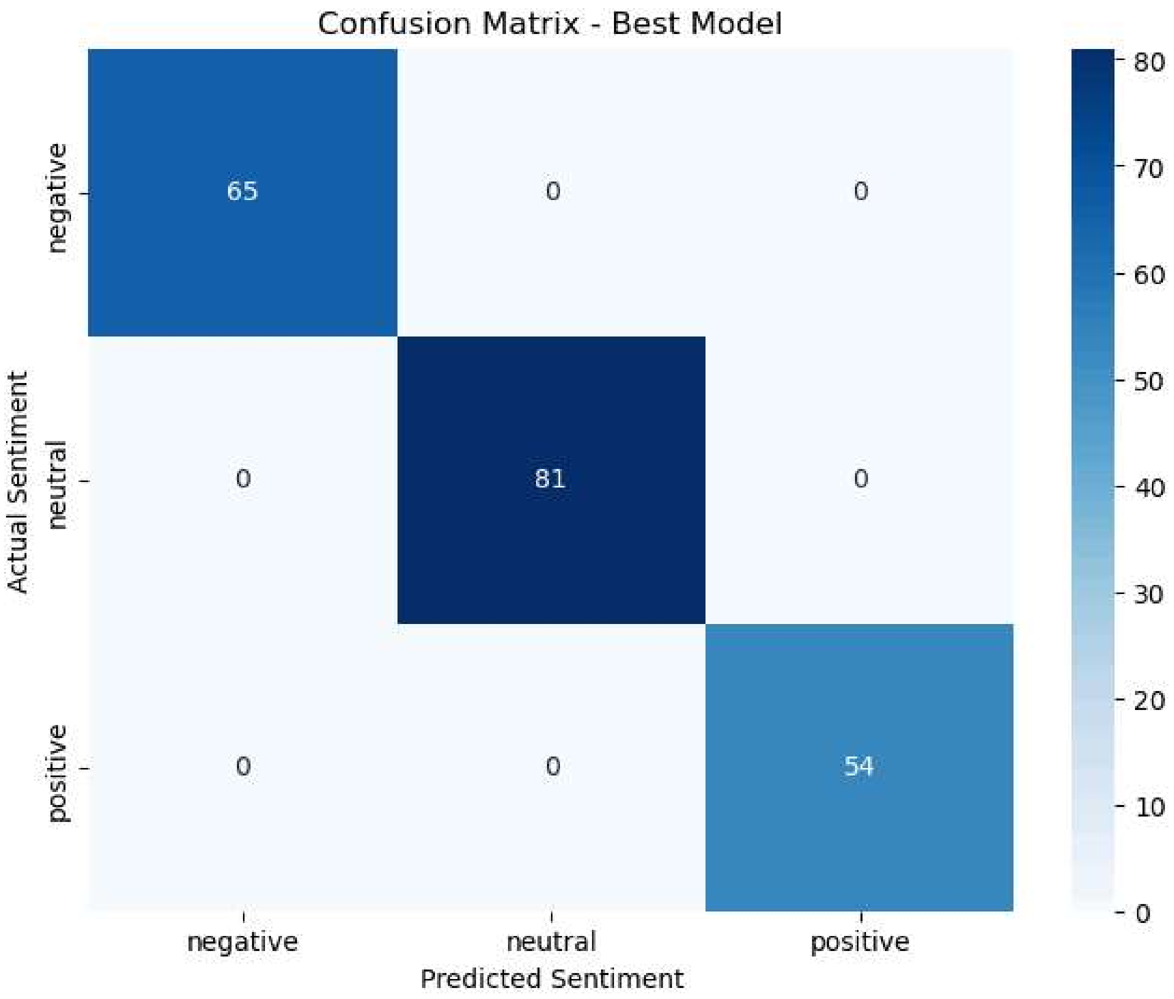
```
# 5. Model Development & Evaluation
models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42)
    "Logistic Regression": LogisticRegression(random_state=42),
    "SVM": SVC(kernel="linear", random_state=42)
}

best_model = None
best_accuracy = 0

print("Model Performance Evaluation:")

# Evaluate models and select the best one
for model_name, model in models.items():
    model.fit(X_train_tfidf, y_train)
    y_pred = model.predict(X_test_tfidf)

    # Metrics calculation
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average="weighted")
    recall = recall_score(y_test, y_pred, average="weighted")
    f1 = f1_score(y_test, y_pred, average="weighted")

    print(f"\n{model_name} Performance:")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-Score: {f1:.2f}")

    # Track the best model
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = model
```

```
Model Performance Evaluation:

Random Forest Performance:
Accuracy: 100.00%
Precision: 1.00
Recall: 1.00
F1-Score: 1.00

Logistic Regression Performance:
Accuracy: 100.00%
Precision: 1.00
Recall: 1.00
F1-Score: 1.00

SVM Performance:
Accuracy: 100.00%
Precision: 1.00
Recall: 1.00
F1-Score: 1.00
```

In [11]:

```
# 6. Confusion Matrix for Best Model
print("\nBest Model: ", best_model)
best_model_predictions = best_model.predict(X_test_tfidf)
conf_matrix = confusion_matrix(y_test, best_model_predictions)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=best_m
plt.title("Confusion Matrix - Best Model")
plt.xlabel("Predicted Sentiment")
plt.ylabel("Actual Sentiment")
plt.show()
```

```
Best Model:  RandomForestClassifier(random_state=42)
```



In [12]:

```
# 7. Display Dataset with Predicted Sentiment
data["Predicted_Sentiment"] = best_model.predict(vectorizer.transform(data["Pr

# Final Dataset with predictions
print("\nFinal Dataset with Predicted Sentiments:")
print(data)

# Save the final dataset with predictions
data.to_csv("final_processed_dataset_with_predictions.csv", index=False)
print("Final processed dataset with predictions has been saved successfully!")
```

```
Final Dataset with Predicted Sentiments:
                             Review Sentiment                    Cleaned_Text  \
0             This product is fine.   neutral          this product is fine
1             This product is hate.  negative          this product is hate
2             This product is love.  positive          this product is love
3             This product is okay.   neutral          this product is okay
4             This product is poor.  negative          this product is poor
..                              ...       ...                           ...
995      This product is fantastic.  positive     this product is fantastic
996   This product is satisfactory.   neutral  this product is satisfactory
997       This product is mediocre.   neutral      this product is mediocre
998          This product is great.  positive         this product is great
999          This product is great.  positive         this product is great

                                 Tokens      Tokens_No_Stopwords  \
0             [this, product, is, fine]          [product, fine]
1             [this, product, is, hate]          [product, hate]
2             [this, product, is, love]          [product, love]
3             [this, product, is, okay]          [product, okay]
4             [this, product, is, poor]          [product, poor]
..                                  ...                      ...
995      [this, product, is, fantastic]     [product, fantastic]
996   [this, product, is, satisfactory]  [product, satisfactory]
997       [this, product, is, mediocre]      [product, mediocre]
998          [this, product, is, great]         [product, great]
999          [this, product, is, great]         [product, great]

           Lemmatized_Tokens        Processed_Text  Text_Length  \
0            [product, fine]          product fine           12
1            [product, hate]          product hate           12
2            [product, love]          product love           12
3            [product, okay]          product okay           12
4            [product, poor]          product poor           12
..                       ...                   ...          ...
995     [product, fantastic]     product fantastic           17
996  [product, satisfactory]  product satisfactory           20
997      [product, mediocre]      product mediocre           16
998         [product, great]         product great           13
999         [product, great]         product great           13

    Predicted_Sentiment
0               neutral
1              negative
2              positive
3               neutral
4              negative
..                  ...
995            positive
996             neutral
997             neutral
998            positive
999            positive

[1000 rows x 9 columns]
Final processed dataset with predictions has been saved successfully!
```