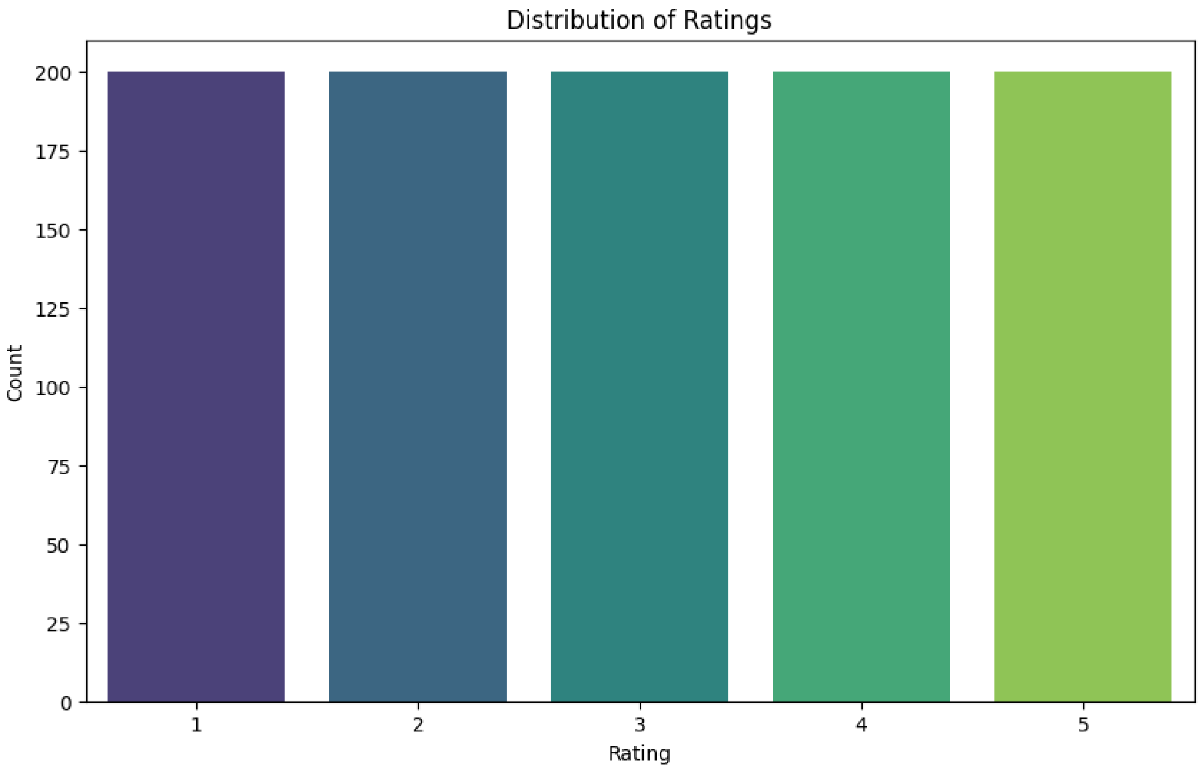
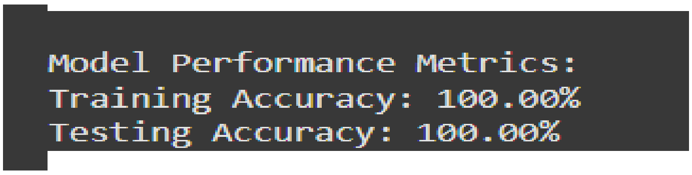
EXPERIMENT NO - 10

CODE :

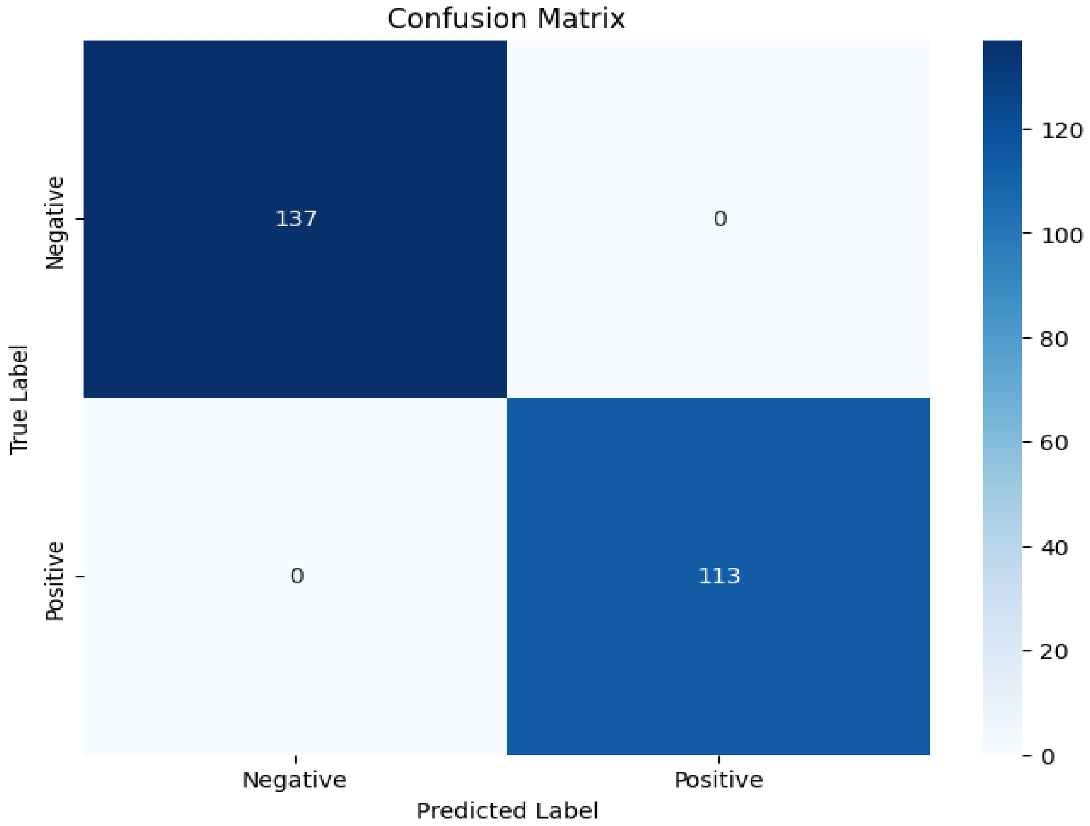
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from wordcloud import WordCloud
from collections import Counter
import warnings
warnings.filterwarnings("ignore")

# Download required NLTK data
nltk.download('stopwords')
nltk.download('punkt')

# Create sample dataset
def create_sample_dataset():
    data = {
        'ReviewID': range(1, 1001),
        'Text': [
            "Amazing product, love it!",
            "Horrible experience, very bad",
            "Decent quality, fair price",
            "Not good, broke quickly",
            "Fantastic service, quick delivery"
        ] * 200,
        'Rating': [5, 1, 3, 2, 4] * 200,
        'Date': pd.date_range(start='2025-01-01', periods=1000)
    }
    return pd.DataFrame(data)

# Text preprocessing
def preprocess_text(text):
    if isinstance(text, str):
        text = re.sub(r'[^a-zA-Z\s]', '', text.lower())
        stop_words = set(stopwords.words('english'))
        words = text.split()
        return ' '.join([word for word in words if word not in stop_words])
    return ""

# Custom confusion matrix plot
def plot_custom_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Negative', 'Positive'],
                yticklabels=['Negative', 'Positive'])
    plt.title('Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.savefig('confusion_matrix.png')
    plt.show()

# Word cloud generation
def generate_word_cloud(text_data, title, filename):
    wordcloud = WordCloud(width=800, height=400, background_color='white',
                          min_font_size=10).generate(' '.join(text_data))
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title)
    plt.savefig(filename)
    plt.show()

def main():
    # Load or create data
    df = create_sample_dataset()  # Replace with: pd.read_csv('your_file.csv')

    # Preprocess text
    df['Cleaned_Text'] = df['Text'].apply(preprocess_text)

    # Convert ratings to sentiment
    df['Sentiment'] = df['Rating'].apply(lambda x: 'Positive' if x >= 4 else 'Negative')

    # Visualize initial rating distribution
    plt.figure(figsize=(10, 6))
    sns.countplot(x='Rating', data=df, palette='viridis')
    plt.title('Distribution of Ratings')
    plt.xlabel('Rating')
    plt.ylabel('Count')
    plt.savefig('rating_distribution.png')
    plt.show()

    # Feature extraction
    tfidf = TfidfVectorizer(max_features=2000, min_df=5)
    X = tfidf.fit_transform(df['Cleaned_Text'])
    y = df['Sentiment']

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

    # Train Random Forest model
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)
    # Predictions
    y_pred_train = rf_model.predict(X_train)
    y_pred_test = rf_model.predict(X_test)

    # Calculate accuracies
    train_accuracy = accuracy_score(y_train, y_pred_train) * 100
    test_accuracy = accuracy_score(y_test, y_pred_test) * 100

    # Print results
    print("\nModel Performance Metrics:")
    print(f"Training Accuracy: {train_accuracy:.2f}%")
    print(f"Testing Accuracy: {test_accuracy:.2f}%")

    # Confusion matrix
    plot_custom_confusion_matrix(y_test, y_pred_test)

    # Classification report
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred_test))

    # Generate word clouds for positive and negative reviews
    positive_reviews = df[df['Sentiment'] == 'Positive']['Cleaned_Text']
    negative_reviews = df[df['Sentiment'] == 'Negative']['Cleaned_Text']

    generate_word_cloud(positive_reviews, 'Positive Reviews Word Cloud', 'positive_wordcloud.png')
    generate_word_cloud(negative_reviews, 'Negative Reviews Word Cloud',
'negative_wordcloud.png')

    # Feature importance
    feature_importance = pd.DataFrame({
        'feature': tfidf.get_feature_names_out(),
        'importance': rf_model.feature_importances_
    }).sort_values('importance', ascending=False).head(10)

    plt.figure(figsize=(10, 6))
    sns.barplot(x='importance', y='feature', data=feature_importance, palette='rocket')
    plt.title('Top 10 Important Features')
    plt.xlabel('Importance Score')
    plt.ylabel('Feature')
    plt.savefig('feature_importance.png')
    plt.show()

    # Sample predictions
    sample_df = pd.DataFrame({
        'Text': df['Text'].iloc[-10:],
        'Predicted_Sentiment': rf_model.predict(X[-10:])
    })
    print("\nSample Predictions:")
    print(sample_df)

    # Sentiment trend over time
    df['Date'] = pd.to_datetime(df['Date'])
    sentiment_by_date = df.groupby(df['Date'].dt.date)['Sentiment'].value_counts().unstack().fillna(0)
    plt.figure(figsize=(12, 6))
    sentiment_by_date.plot(kind='area', stacked=True, alpha=0.5)
    plt.title('Sentiment Trend Over Time')
    plt.xlabel('Date')
    plt.ylabel('Number of Reviews')
    plt.legend(['Negative', 'Positive'])
    plt.savefig('sentiment_trend.png')
    plt.show()

if __name__ == "__main__":
    main()
```

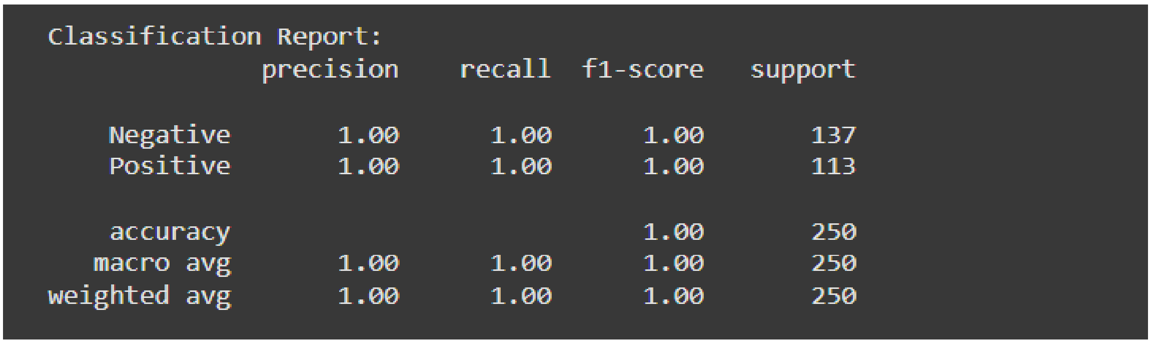
OUTPUT :

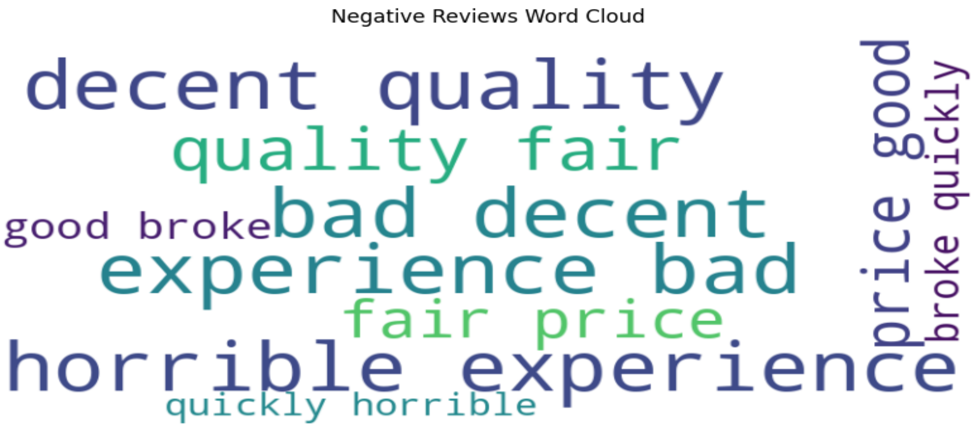




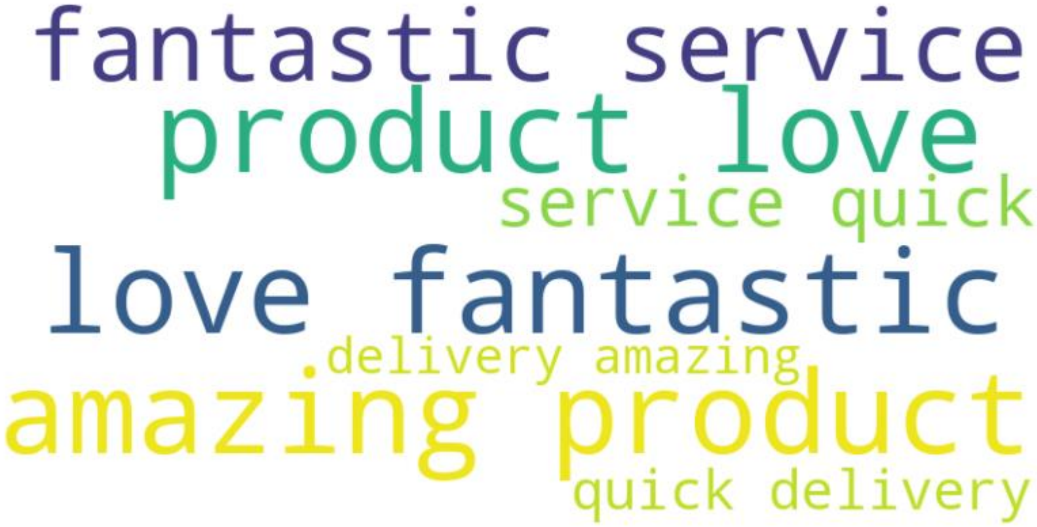
```
Model Performance Metrics:
Training Accuracy: 100.00%
Testing Accuracy: 100.00%
```



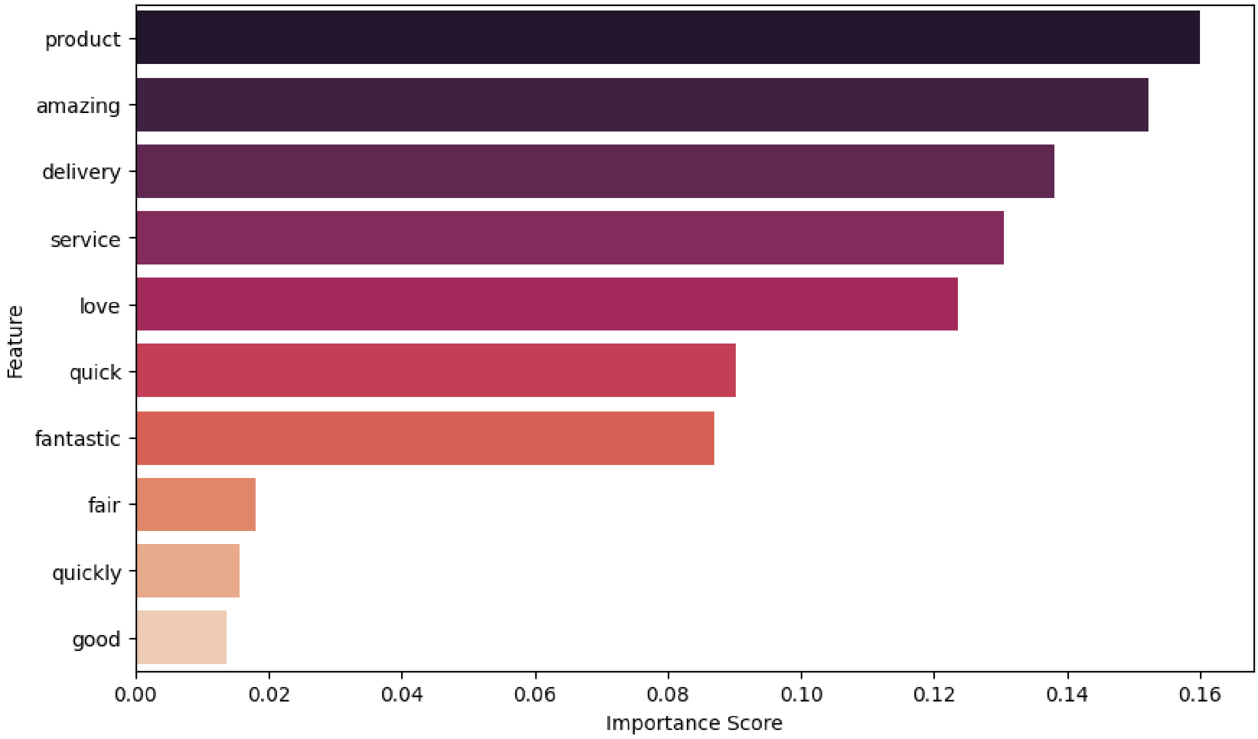

```
Classification Report:
              precision    recall  f1-score   support

    Negative       1.00      1.00      1.00       137
    Positive       1.00      1.00      1.00       113

    accuracy                           1.00       250
   macro avg       1.00      1.00      1.00       250
weighted avg       1.00      1.00      1.00       250
```

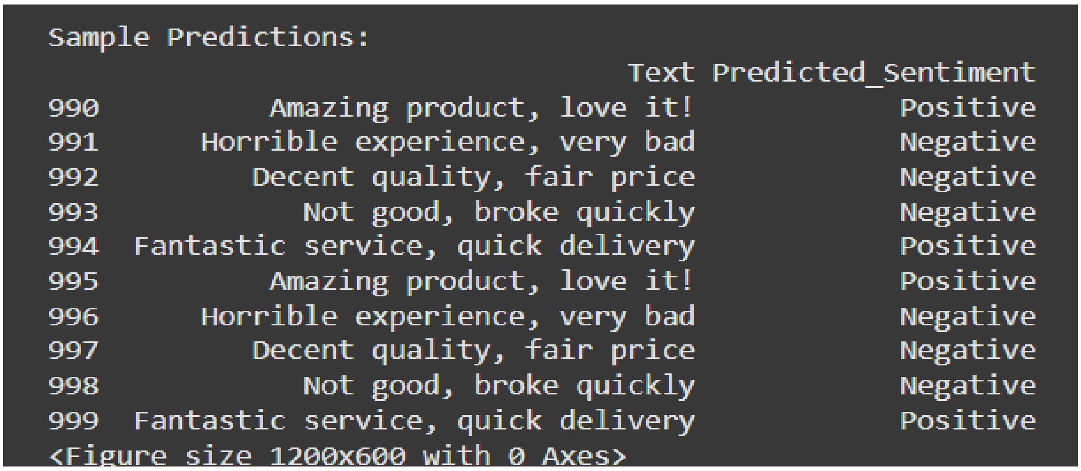


Positive Reviews Word Cloud

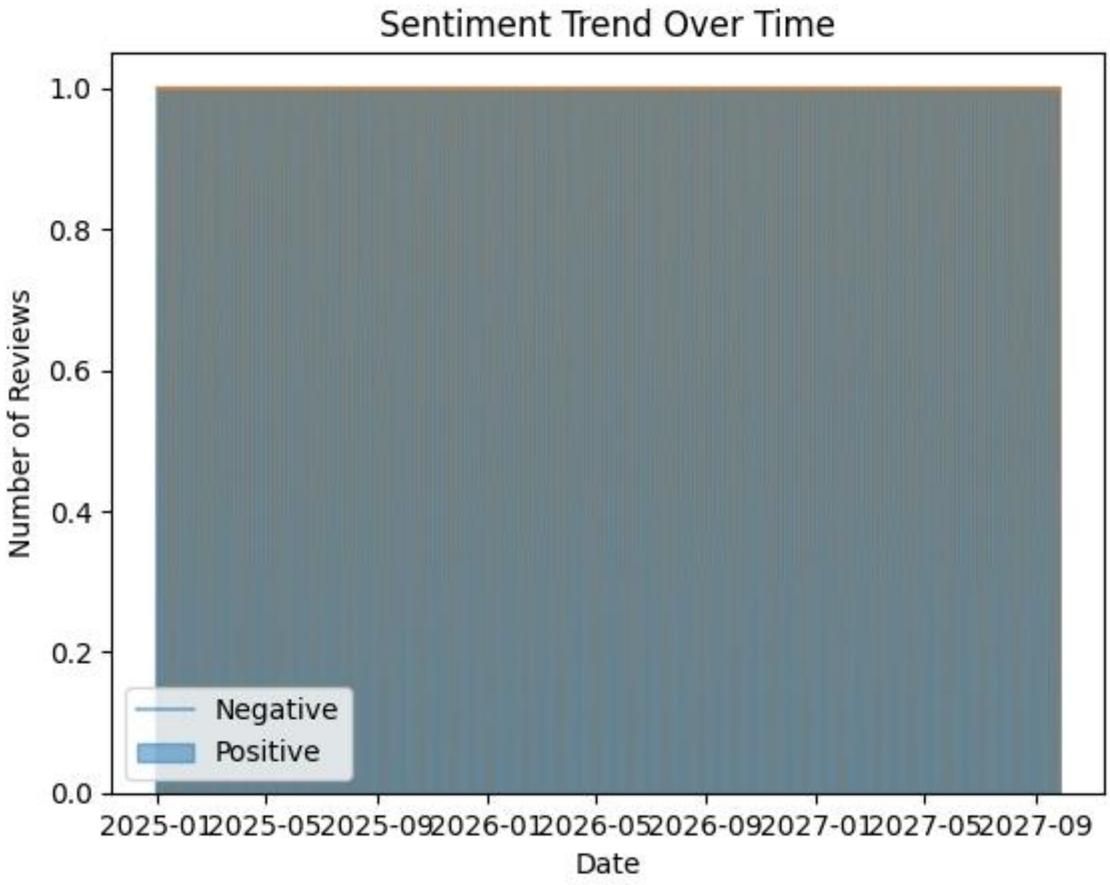


Top 10 Important Features





```
Sample Predictions:
                                   Text Predicted_Sentiment
990          Amazing product, love it!            Positive
991      Horrible experience, very bad            Negative
992         Decent quality, fair price            Negative
993            Not good, broke quickly            Negative
994  Fantastic service, quick delivery            Positive
995          Amazing product, love it!            Positive
996      Horrible experience, very bad            Negative
997         Decent quality, fair price            Negative
998            Not good, broke quickly            Negative
999  Fantastic service, quick delivery            Positive
<Figure size 1200x600 with 0 Axes>
```


Sentiment Trend Over Time
Number of Reviews
Negative
Positive
Date