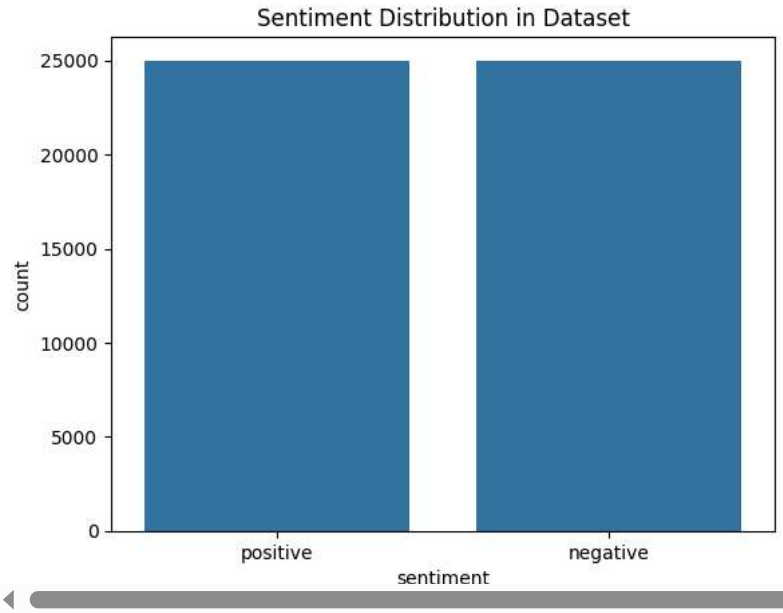
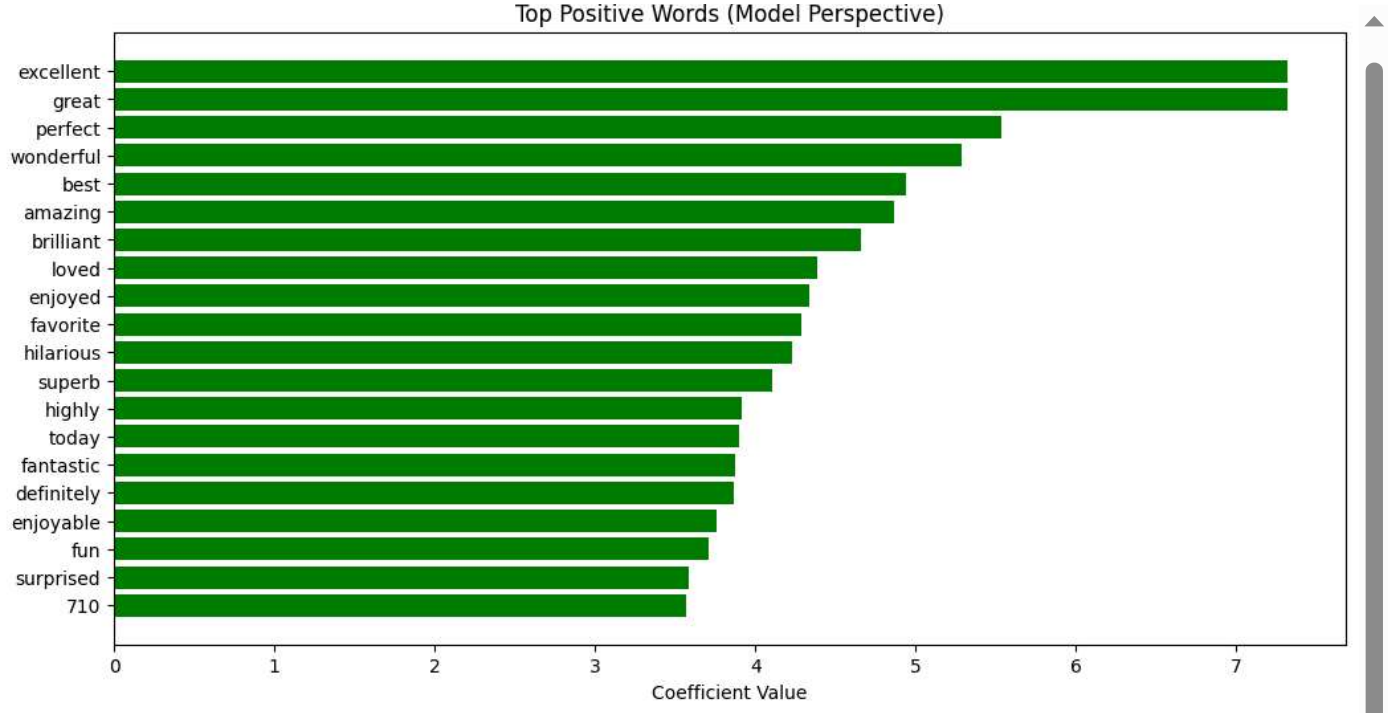
```python
import pandas as pd
import numpy as np
import re
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset, handling potential errors with 'error_bad_lines=False'
# and specifying the quote character if necessary.
df = pd.read_csv("/content/IMDB Dataset.csv")

# 1. Text Preprocessing Function
def clean_text(text):
    text = text.lower()
    text = re.sub('<.*?>', '', text)  # Remove HTML tags
    text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)  # Remove
    punctuation
    text = re.sub('\s+', ' ', text).strip()  # Remove extra spaces
    return text

df['clean_review'] = df['review'].apply(clean_text)

# 2. Split data
X = df['clean_review']
y = df['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 3. TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# 4. Model Training
model = LogisticRegression()
model.fit(X_train_tfidf, y_train)

# 5. Predictions and Evaluation
y_pred = model.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 6. Visualization
sns.countplot(data=df, x='sentiment')
plt.title("Sentiment Distribution in Dataset")
plt.show()
```

```
Accuracy: 0.8945

Classification Report:
              precision    recall  f1-score   support

    negative       0.90      0.88      0.89      4961
    positive       0.89      0.91      0.90      5039

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```




```
from wordcloud import WordCloud

# Separate positive and negative reviews
positive_text = " ".join(df[df['sentiment'] == 'positive']['clean_review'])
negative_text = " ".join(df[df['sentiment'] == 'negative']['clean_review'])

# Generate word clouds
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
wc_pos = WordCloud(width=600, height=400, background_color='white').generate(positive_text)
plt.imshow(wc_pos, interpolation='bilinear')
plt.axis('off')
plt.title('Top Words in Positive Reviews')

plt.subplot(1, 2, 2)
wc_neg = WordCloud(width=600, height=400, background_color='black', colormap='Reds').generate(negative_text)
plt.imshow(wc_neg, interpolation='bilinear')
plt.axis('off')
plt.title('Top Words in Negative Reviews')

plt.tight_layout()
plt.show()

```


```
# Get top words for each sentiment
feature_names = tfidf.get_feature_names_out()
coefs = model.coef_[0]
top_pos_indices = np.argsort(coefs)[-20:]
top_neg_indices = np.argsort(coefs)[:20]

# Plot
plt.figure(figsize=(12, 6))
plt.barh([feature_names[i] for i in top_pos_indices], coefs[top_pos_indices], color='green')
plt.title("Top Positive Words (Model Perspective)")
plt.xlabel("Coefficient Value")
plt.show()

plt.figure(figsize=(12, 6))
plt.barh([feature_names[i] for i in top_neg_indices], coefs[top_neg_indices], color='red')
plt.title("Top Negative Words (Model Perspective)")
plt.xlabel("Coefficient Value")
plt.show()

```




```
1 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
2
3 cm = confusion_matrix(y_test, y_pred, labels=["positive", "negative"])
4 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Positive", "Negative"])
5 disp.plot(cmap='Blues')
6 plt.title("Confusion Matrix")
7 plt.show()
8
```