Dissertation

Harnessing Big Data Analytics to Decode Consumer Behavior: Driving Innovations in Coffee Quality and Precision Market Segmentation

Sentiment Analysis Pipeline

Step 1: Import Required Libraries

In [3]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, SpatialDropout1D, Bidirectional, LayerNorma
from tensorflow.keras.optimizers.schedules import ExponentialDecay
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_report

from wordcloud import WordCloud
import joblib
import nltk
import re
from nltk.corpus import stopwords
```

Step 2: Load the Dataset

In [4]:

```
file_path = r"C:\Users\tariq\OneDrive\Desktop\Reviews Dataset.xlsx"
df = pd.read_excel(file_path, engine="openpyxl")
print(df.head())
```

```
                                             Reviews
0  I order in the drive thru. A breakfast meal in...
1  Ordering kiosk is a nuisance. I just wanted a ...
2  Whoever the dude was working the drive thru to...
3  Just had apple pie and coffee. I had to take m...
4  Every was great I was visiting my wife sister ...
```

□Step 3: Step 3️⃣: Perform Sentiment Analysis & Generate Sentiment Labels

In [5]:

```
from transformers import pipeline

sentiment_pipeline = pipeline("sentiment-analysis")

def get_sentiment_label(text):
    if isinstance(text, str) and text.strip():
        result = sentiment_pipeline(text[:512])[0]
        return result['label']
    return "NEUTRAL"

df['Sentiment'] = df['Reviews'].astype(str).apply(get_sentiment_label)
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714e
b0f (https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
```

```
WARNING:tensorflow:From C:\Users\tariq\anaconda3_new\Lib\site-packages\tf_keras\src\losses.py:2976: The name tf.
losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy i
nstead.
```

```
Device set to use cpu
```

Step 4: Convert Sentiment to Numeric Values

In [6]:

```
label_mapping = {'POSITIVE': 1, 'NEGATIVE': 0}
df['Sentiment_Label'] = df['Sentiment'].map(label_mapping)
```

Step 5: Visualize Sentiment Distribution

In [7]:

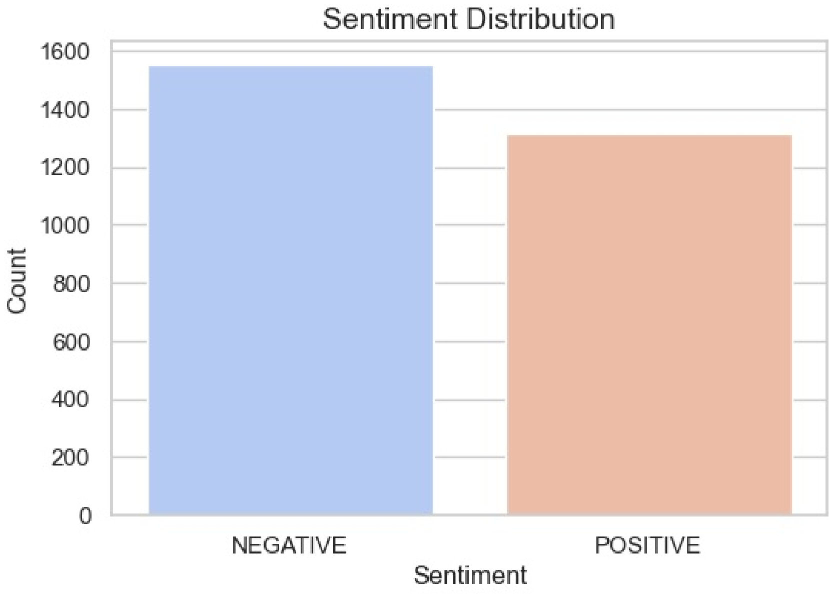
```
sns.set(style="whitegrid")
plt.figure(figsize=(6,4))
ax = sns.countplot(x=df['Sentiment'], palette="coolwarm")
plt.title("Sentiment Distribution", fontsize=14)
plt.xlabel("Sentiment")
plt.ylabel("Count")

save_path = r"C:\Users\tariq\OneDrive\Desktop\Sentiment_Distribution.png"
plt.savefig(save_path, dpi=300, bbox_inches="tight")
plt.show()
print(f"✅ Sentiment distribution chart saved to: {save_path}")
```

```
C:\Users\tariq\AppData\Local\Temp\ipykernel_20696\2819600571.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(x=df['Sentiment'], palette="coolwarm")
```



```
✅ Sentiment distribution chart saved to: C:\Users\tariq\OneDrive\Desktop\Sentiment_Distribution.png
```

Step 6: Generate Word Cloud

In [8]:

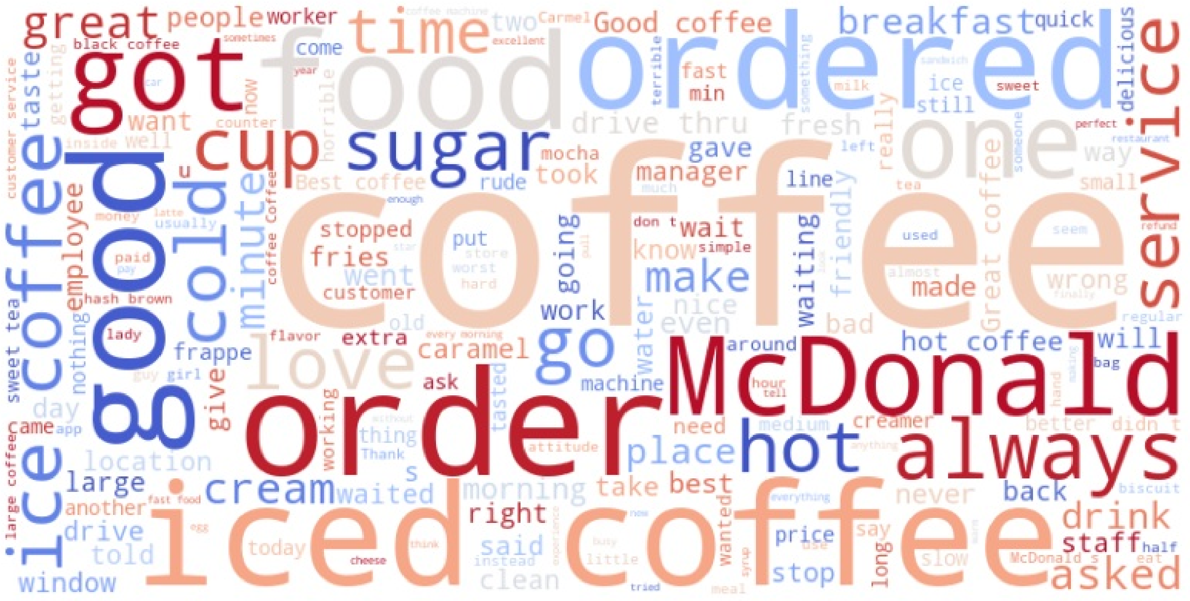
```
text = " ".join(df['Reviews'].astype(str))

wordcloud = WordCloud(width=800, height=400, background_color="white", colormap="coolwarm").generate(text)

wordcloud_path = r"C:\Users\tariq\OneDrive\Desktop\WordCloud.png"
wordcloud.to_file(wordcloud_path)

plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()

print(f"✅ Word Cloud saved to: {wordcloud_path}")
```



✅ Word Cloud saved to: C:\Users\tariq\OneDrive\Desktop\WordCloud.png

Step 7: Split Dataset (80% Train / 20% Test)

In [9]:

```
X_train, X_test, y_train, y_test = train_test_split(
    df['Reviews'], df['Sentiment_Label'], test_size=0.2, random_state=42, stratify=df['Sentiment_Label']
)
```

Step 8: Convert Text to TF-IDF Features

In [10]:

```
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

print(f"✅ TF-IDF Transformation Complete! Shape: {X_train_tfidf.shape}")
```

```
✅ TF-IDF Transformation Complete! Shape: (2295, 3653)
```

Step 9: Train Multiple ML Models

In [11]:

```
models = {
    "Logistic Regression": LogisticRegression(max_iter=200),
    "Random Forest": RandomForestClassifier(n_estimators=200, max_depth=10, random_state=42),
    "Naive Bayes": MultinomialNB(),
    "Support Vector Machine (SVM)": SVC(kernel='linear', C=1.0)
}

results = {}
for name, model in models.items():
    print(f"\n Training {name}...")
    model.fit(X_train_tfidf, y_train)
    y_pred = model.predict(X_test_tfidf)
    acc = accuracy_score(y_test, y_pred)
    results[name] = acc
    print(f"{name} Accuracy: {acc:.4f}")
    print(classification_report(y_test, y_pred))
    print("-" * 50)

best_model_name = max(results, key=results.get)
print(f"\n✅ Best Model: {best_model_name} with Accuracy: {results[best_model_name]:.4f}")
```

```
 Training Logistic Regression...
Logistic Regression Accuracy: 0.8449
              precision    recall  f1-score   support

           0       0.82      0.92      0.87       311
           1       0.89      0.76      0.82       263

    accuracy                           0.84       574
   macro avg       0.85      0.84      0.84       574
weighted avg       0.85      0.84      0.84       574

--------------------------------------------------

 Training Random Forest...
Random Forest Accuracy: 0.8014
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       311
           1       0.81      0.74      0.77       263

    accuracy                           0.80       574
   macro avg       0.80      0.80      0.80       574
weighted avg       0.80      0.80      0.80       574

--------------------------------------------------

 Training Naive Bayes...
Naive Bayes Accuracy: 0.8397
              precision    recall  f1-score   support

           0       0.80      0.94      0.86       311
           1       0.91      0.72      0.81       263

    accuracy                           0.84       574
   macro avg       0.85      0.83      0.83       574
weighted avg       0.85      0.84      0.84       574

--------------------------------------------------

 Training Support Vector Machine (SVM)...
Support Vector Machine (SVM) Accuracy: 0.8589
              precision    recall  f1-score   support

           0       0.83      0.92      0.88       311
           1       0.90      0.78      0.84       263

    accuracy                           0.86       574
   macro avg       0.86      0.85      0.86       574
weighted avg       0.86      0.86      0.86       574

--------------------------------------------------

✅ Best Model: Support Vector Machine (SVM) with Accuracy: 0.8589
```

Step : Optimize SVM and Save the Model

In [12]:

```python
nltk.download("stopwords")
stop_words = set(stopwords.words("english"))

def clean_text(text):
    if isinstance(text, str):
        text = text.lower()
        text = re.sub(r'[^a-z\s]', '', text)
        text = " ".join([word for word in text.split() if word not in stop_words])
    return text

df["Cleaned_Reviews"] = df["Reviews"].apply(clean_text)

# TF-IDF with tuned parameters
vectorizer = TfidfVectorizer(max_features=8000, ngram_range=(1,2), max_df=0.9, min_df=2)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# SVM with class weights
svm_model = SVC(kernel='linear', C=2.0, class_weight="balanced")
svm_model.fit(X_train_tfidf, y_train)
y_pred = svm_model.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred)
print(f"\n✅ Final SVM Accuracy: {accuracy:.4f}")
print(classification_report(y_test, y_pred))
```

```python
# Save model and vectorizer
model_path = r"C:\Users\tariq\OneDrive\Desktop\Best_SVM_Model.pkl"
vectorizer_path = r"C:\Users\tariq\OneDrive\Desktop\TFIDF_Vectorizer.pkl"

joblib.dump(svm_model, model_path)
joblib.dump(vectorizer, vectorizer_path)

print(f"✅ Model saved to: {model_path}")
print(f"✅ Vectorizer saved to: {vectorizer_path}")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\tariq\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
✅ Final SVM Accuracy: 0.8624
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       311
           1       0.88      0.81      0.84       263

    accuracy                           0.86       574
   macro avg       0.86      0.86      0.86       574
weighted avg       0.86      0.86      0.86       574

✅ Model saved to: C:\Users\tariq\OneDrive\Desktop\Best_SVM_Model.pkl
✅ Vectorizer saved to: C:\Users\tariq\OneDrive\Desktop\TFIDF_Vectorizer.pkl
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js