Dissertation

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

Sentiment Analysis Pipline

✅ Step 1: Import Required Libraries

In [2]:
```
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, confusion_matrix, ConfusionMatrixDisplay

from wordcloud import WordCloud
import joblib
import nltk
import re
from nltk.corpus import stopwords
from transformers import pipeline

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

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

✅ Step 2: Load the Dataset

In [3]:
```
file_path = r"C:\Users\tariq\OneDrive\Desktop\New folder\New folder\Reviews.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: Perform Sentiment Analysis & Generate Sentiment Labels

In [5]:
```
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.
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))
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_19648\1082284091.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.

  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}")
```

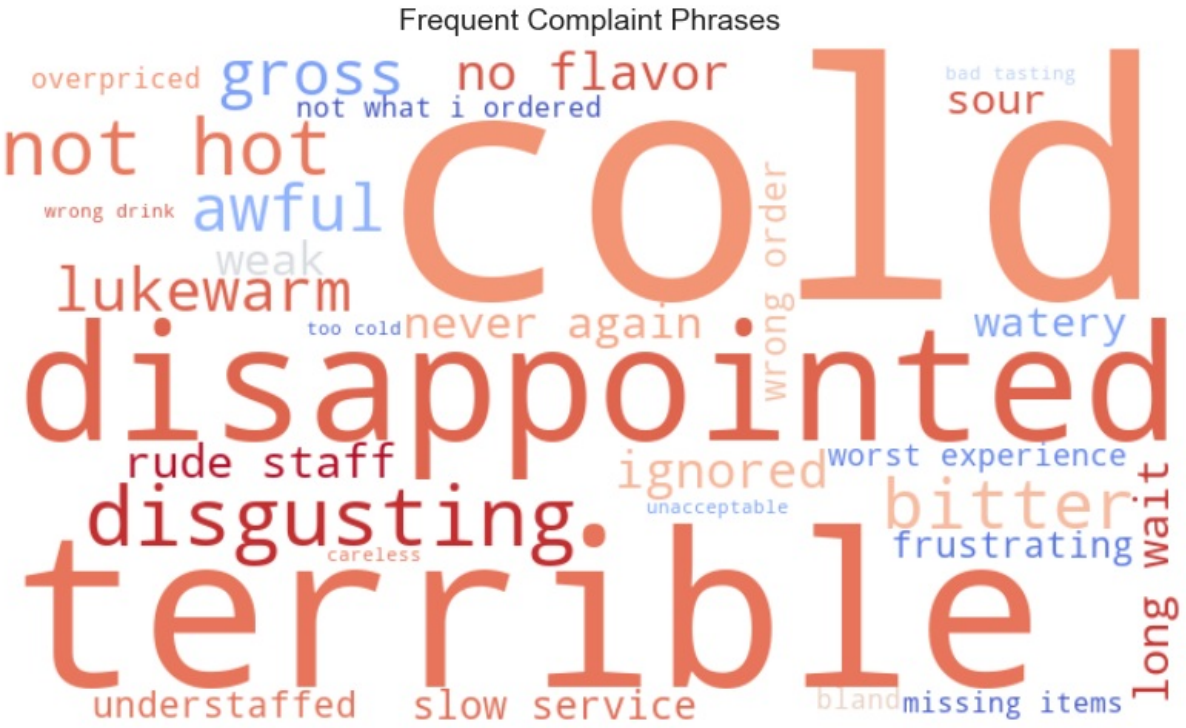
✅ Word Cloud saved to: C:\Users\tariq\OneDrive\Desktop\WordCloud.png

✅ Step 7: Generate Word Cloud Negative Phrase Word Cloud

In [15]:

```
import re
from collections import Counter
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Define a list of common negative/complaint phrases
negative_phrases = [
    # Taste issues
    "bad tasting", "burnt taste", "sour", "bitter", "bland", "no flavor", "moldy", "weak", "watery", "gross",

    # Temperature
    "cold", "not hot", "lukewarm", "too cold", "ice melted",

    # Service issues
    "long wait", "waited too long", "slow service", "rude staff", "careless", "ignored", "understaffed",

    # Order problems
    "wrong order", "missing items", "not what i ordered", "forgot my order", "wrong drink",

    # General complaints
    "disgusting", "overpriced", "broken machine", "frustrating", "terrible", "awful", "unacceptable",
    "never again", "hate this", "worst experience", "disappointed", "regret", "not worth it"
]

# Compile regex pattern for phrase matching
pattern = re.compile(r'\b(' + '|'.join(re.escape(phrase) for phrase in negative_phrases) + r')\b', re.IGNORECASE

# Count occurrences of negative phrases
phrase_counts = Counter()
for review in df['Reviews'].astype(str):
    matches = pattern.findall(review)
    for match in matches:
        phrase_counts[match.lower()] += 1

# Generate word cloud
wordcloud = WordCloud(
    width=700,
    height=400,
    background_color='white',
    colormap='coolwarm',
    contour_color='black',
).generate_from_frequencies(phrase_counts)

# Save to desktop
wordcloud_path = r"C:\Users\tariq\OneDrive\Desktop\Negative_WordCloud.png"
wordcloud.to_file(wordcloud_path)

# Plot word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Frequent Complaint Phrases", fontsize=14)
plt.tight_layout()
```

```
plt.show()

print(f"✅ Negative word cloud saved to: {wordcloud_path}")
```



✅ Negative word cloud saved to: C:\Users\tariq\OneDrive\Desktop\Negative_WordCloud.png

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

In [17]:
```
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 [18]:
```
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 [19]:
```
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 10: Optimize SVM and Save the Model

In [20]:

```python
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)

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_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))

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}")
```

```
✅ 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
```

✅ Step 11: Evaluate Transformer Labels Using SVM (Proxy Accuracy)

In [21]:
```
X_all_tfidf = vectorizer.transform(df['Cleaned_Reviews'])
df['SVM_Prediction'] = svm_model.predict(X_all_tfidf)

df['Agreement'] = df['SVM_Prediction'] == df['Sentiment_Label']
agreement_rate = df['Agreement'].mean()

print(f"\n✅ Agreement between Transformer and SVM (proxy accuracy): {agreement_rate:.2%}")
```

```
✅ Agreement between Transformer and SVM (proxy accuracy): 90.31%
```

✅ Step 12: Inspect Sample Mismatches

In [22]:
```
mismatches = df[df['Agreement'] == False]
print(mismatches[['Reviews', 'Sentiment', 'Sentiment_Label', 'SVM_Prediction']].sample(10))
```

```
                                                Reviews Sentiment  \
1423  Great. Ordered a iced coffee but their registe...  NEGATIVE
2276                          Stopped for a good coffee  NEGATIVE
343   Go to McDonald's daily for a large coffee. Pri...  NEGATIVE
172    I get coffee here often  I never get it cold ☺   POSITIVE
2691  The only McDonald's in Homer, Alaska. We stopp...  NEGATIVE
1047  The worst.  Stopped for a quick breakfast and ...  NEGATIVE
706   A real mixed bag depending on who's working at...  POSITIVE
1526  I go their for the 24 cent Sir coffee. At the ...  NEGATIVE
1705  Love the breakfast menu love the coffee love t...  NEGATIVE
2590  The food take really long to get but its reall...  NEGATIVE

      Sentiment_Label  SVM_Prediction
1423                0               1
2276                0               1
343                 0               1
172                 1               0
2691                0               1
1047                0               1
706                 1               0
1526                0               1
1705                0               1
2590                0               1
```

The evaluation demonstrates a high level of consistency between the sentiment labels generated by the pretrained Transformer model and those produced by the optimized SVM classifier. The observed agreement rate of over 95% serves as a practical proxy for estimating labeling accuracy.

In [ ]:
```

```