In [ ]:

```

```

In [1]:

```
import pandas as pd
import re

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 accuracy_score, classification_report, confusion_matrix
```

In [ ]:

```

```

In [4]:

```
uber = pd.read_csv("Downloads/uber_reviews_without_reviewid.csv")
uber
```

Out[4]:

	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersic
0	User_0	NaN	Good	5	0	4.556.100
1	User_1	NaN	Nice	5	0	4.556.100
2	User_2	NaN	Very convenient	5	0	4.532.100
3	User_3	NaN	Good	4	0	4.556.100
4	User_4	NaN	exllence	5	0	4.556.100
...	...	...	...	...	...	
11995	User_11995	NaN	Excellent!!!	5	0	4.553.100
11996	User_11996	NaN	Worst experience after 10pm in Hyde cityno aut...	5	0	4.552.100
11997	User_11997	NaN	Exceptional	5	0	4.552.100
11998	User_11998	NaN	Good Service.	5	0	4.553.100
11999	User_11999	NaN	Very bad experience with this app, booked a sh...	1	0	Na

12000 rows × 10 columns

In [6]:

```
uber = uber[['content', 'score']].dropna()
uber.columns = ['review', 'rating']
```

Create Sentiment Label

In [8]:

```
uber['sentiment'] = uber['rating'].apply(lambda x: 1 if x >= 4 else 0)

print(uber['sentiment'].value_counts())
```

```
sentiment
1    8732
0    3268
Name: count, dtype: int64
```

Clean Text

In [10]:

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

uber['review'] = uber['review'].apply(clean_text)
```

Convert Text to TF-IDF Features

In [12]:

```
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_features=8000,
    ngram_range=(1,2)   # Bigram improves accuracy
)

X = vectorizer.fit_transform(uber['review'])
y = uber['sentiment']
```

In [14]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)
```

In [16]:

```
model = LogisticRegression(
    max_iter=2000,
    class_weight='balanced'
)

model.fit(X_train, y_train)
```

Out[16]:

```
▾ LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=2000)
```

In [18]:

```
predictions = model.predict(X_test)
```

In [20]:

```
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

print("\nClassification Report:\n")
print(classification_report(y_test, predictions))

print("\nConfusion Matrix:\n")
print(confusion_matrix(y_test, predictions))
```

```
Accuracy: 0.9358333333333333

Classification Report:

              precision    recall  f1-score   support

           0       0.84      0.93      0.88       634
           1       0.98      0.94      0.96      1766

    accuracy                           0.94      2400
   macro avg       0.91      0.94      0.92      2400
weighted avg       0.94      0.94      0.94      2400


Confusion Matrix:

[[ 592   42]
 [ 112 1654]]
```

In [ ]:

```python

```

In [ ]:

```python

```

In [24]:

```python
ola = pd.read_csv("Downloads/ola_review_dataset.csv")
ola
```

Out[24]:

	review_id	rating	review_text
0	1e4c163e-144a-4ea4-b0bd-80e5e9f259bd	1	unexpected charges if driver cancel ride don't...
1	447aaf31-3968-45f1-877f-5e04d230606a	2	some problem with the app. Card is saved but c...
2	84e9cae0-5cb2-439f-8cd4-ba97fe72b478	3	Services of RAPIDO are far better.
3	2d50f89e-025a-4a29-88ab-00c5ee052bfe	1	Adding cancellation charges while the driver i...
4	49bcf932-c446-461a-ae58-a2c5f6d91fe6	1	very costly - uber is good
...	...	...	...
9995	224fca30-25c4-4924-bea8-13633d0754cc	5	good app
9996	f813ae5e-68f7-4478-b297-8e60194e52e2	1	location issue... worst map
9997	f96beec1-b3df-4b35-8e06-a5cc15c7e33c	5	very convenient
9998	1494030d-3122-4b84-96c0-315fecc6541c	1	worst app..
9999	aedab05a-6c18-49e7-a8df-54f253e025fb	5	qood service

10000 rows × 3 columns

In [26]:

```python
ola = ola[['review_text', 'rating']].dropna()
ola.columns = ['review', 'rating']
```

In [28]:

```python
ola['sentiment'] = ola['rating'].apply(lambda x: 1 if x >= 4 else 0)

print(ola['sentiment'].value_counts())
```

```
sentiment
0    7755
1    2245
Name: count, dtype: int64
```

In [30]:

```python
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

ola['review'] = ola['review'].apply(clean_text)
```

In [32]:

```python
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_features=8000,
    ngram_range=(1,2)
)

X = vectorizer.fit_transform(ola['review'])
y = ola['sentiment']
```

In [34]:

```python
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)
```

In [36]:

```python
# Train
model = LogisticRegression(max_iter=2000, class_weight='balanced')
model.fit(X_train, y_train)
```

Out[36]:

```
▾ LogisticRegression

LogisticRegression(class_weight='balanced', max_iter=2000)
```

In [38]:

```python
predictions = model.predict(X_test)
```

In [40]:

```python
print("Accuracy:", accuracy_score(y_test, predictions))
```

```
Accuracy: 0.936
```

In [43]:

```python
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

print("\nClassification Report:\n")
print(classification_report(y_test, predictions))

print("\nConfusion Matrix:\n")
print(confusion_matrix(y_test, predictions))
```

```
Accuracy: 0.936

Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1565
           1       0.85      0.86      0.85       435

    accuracy                           0.94      2000
   macro avg       0.91      0.91      0.91      2000
weighted avg       0.94      0.94      0.94      2000


Confusion Matrix:

[[1500   65]
 [  63  372]]
```

In [ ]: