Sentiment Analysis of Flipkart Product Reviews (MLOps)

This notebook demonstrates an end-to-end MLOps workflow for sentiment analysis using Flipkart product reviews. It includes model training, evaluation, hyperparameter tuning, experiment tracking, model registration, and workflow orchestration.

Step 1: Import Required Libraries

We import all necessary libraries for data handling, machine learning, hyperparameter tuning, experiment tracking, and orchestration.

```
!pip install -q optuna mlflow prefect
```

```
import pandas as pd
import numpy as np
import re

from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

import optuna
import mlflow
import mlflow.sklearn
from prefect import flow, task
```

Step 2: Load Dataset

The Flipkart review dataset is loaded. It contains customer reviews and ratings.

```
df = pd.read_csv('badminton.csv')
print(df.shape)
df.head()
```

```
(8518, 8)
```

	Reviewer Name	Review Title	Place of Review	Up Votes	Down Votes	Month	Review text	Ratings
0	Kamal Suresh	Nice product	Certified Buyer, Chirakkal	889.0	64.0	Feb 2021	Nice product, good quality, but price is now r...	4
1	Flipkart Customer	Don't waste your money	Certified Buyer, Hyderabad	109.0	6.0	Feb 2021	They didn't supplied Yonex Mavis 350. Outside ...	1
2	A. S. Raja Srinivasan	Did not meet expectations	Certified Buyer, Dharmapuri	42.0	3.0	Apr 2021	Worst product. Damaged shuttlecocks packed in ...	1
3	Suresh Narayanasamy	Fair	Certified Buyer, Chennai	25.0	1.0	NaN	Quite O. K. , but nowadays the quality of the...	3
4	ASHIK P A	Over priced	NaN	147.0	24.0	Apr 2019	Over pricedJust â?¹620 ..from retailer	1

Next steps: Generate code with df New interactive sheet

Step 3: Data Preprocessing and Sentiment Creation

Ratings are converted into sentiment labels. Positive sentiment corresponds to ratings 4 and 5, negative sentiment corresponds to ratings 1 and 2. Neutral reviews are removed.

```
df.columns = df.columns.str.lower().str.replace(' ', '_')
df = df[df['ratings'] != 3]
df['sentiment'] = df['ratings'].apply(lambda x: 1 if x >= 4 else 0)
```

```
X = df['review_text']
y = df['sentiment']
```

Step 4: Text Cleaning Transformer

A custom text cleaner is defined to normalize review text.

```
class CleanText:
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        if isinstance(X, list):
            X = pd.Series(X)
        return X.astype(str).str.lower().str.replace(r'[^a-z\s]', '', regex=True)
```

Step 5: Train-Test Split

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

Step 6: Hyperparameter Tuning with Optuna

Optuna is used to tune the TF-IDF and Logistic Regression hyperparameters.

```
def objective(trial):
    max_features = trial.suggest_int('max_features', 1000, 5000)
    C = trial.suggest_float('C', 0.1, 5.0)

    pipeline = Pipeline([
        ('clean_text', CleanText()),
        ('tfidf', TfidfVectorizer(max_features=max_features)),
        ('model', LogisticRegression(max_iter=1000, C=C))
    ])

    pipeline.fit(X_train, y_train)
    preds = pipeline.predict(X_test)
    return f1_score(y_test, preds)

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=10)

study.best_params, study.best_value
```

```
[I 2026-02-03 19:45:12,728] A new study created in memory with name: no-name-270c6784-2ea6-45f1-9eae-c5248d3b0698
[I 2026-02-03 19:45:13,236] Trial 0 finished with value: 0.9635247381726255 and parameters: {'max_features': 3679, 'C': 2.7
[I 2026-02-03 19:45:13,471] Trial 1 finished with value: 0.9628828828828829 and parameters: {'max_features': 2610, 'C': 2.1
[I 2026-02-03 19:45:13,773] Trial 2 finished with value: 0.9641174338528452 and parameters: {'max_features': 4896, 'C': 3.6
[I 2026-02-03 19:45:14,331] Trial 3 finished with value: 0.9655172413793104 and parameters: {'max_features': 3702, 'C': 4.5
[I 2026-02-03 19:45:15,083] Trial 4 finished with value: 0.9655172413793104 and parameters: {'max_features': 4638, 'C': 4.5
[I 2026-02-03 19:45:15,747] Trial 5 finished with value: 0.9628828828828829 and parameters: {'max_features': 4963, 'C': 1.7
[I 2026-02-03 19:45:16,410] Trial 6 finished with value: 0.9643895348837209 and parameters: {'max_features': 2377, 'C': 4.8
[I 2026-02-03 19:45:17,033] Trial 7 finished with value: 0.9590017825311943 and parameters: {'max_features': 4570, 'C': 0.4
[I 2026-02-03 19:45:17,593] Trial 8 finished with value: 0.9631768953068592 and parameters: {'max_features': 4079, 'C': 1.9
[I 2026-02-03 19:45:17,977] Trial 9 finished with value: 0.9648168298875589 and parameters: {'max_features': 4940, 'C': 4.6
({'max_features': 3702, 'C': 4.574250808223088}, 0.9655172413793104)
```

Step 7: MLflow Experiment Tracking and Model Registry

The best model is trained again using optimal hyperparameters and logged to MLflow.

```
mlflow.set_experiment('Flipkart_Sentiment_MLOps')
```

```
with mlflow.start_run():
    best_pipeline = Pipeline([
        ('clean_text', CleanText()),
        ('tfidf', TfidfVectorizer(max_features=study.best_params['max_features'])),
        ('model', LogisticRegression(max_iter=1000, C=study.best_params['C']))
    ])

    best_pipeline.fit(X_train, y_train)
    preds = best_pipeline.predict(X_test)
    f1 = f1_score(y_test, preds)

    mlflow.log_params(study.best_params)
    mlflow.log_metric('f1_score', f1)
    mlflow.sklearn.log_model(best_pipeline, 'model', registered_model_name='FlipkartSentimentModel')

f1
```

```
2026/02/03 19:45:18 WARNING mlflow.models.model: `artifact_path` is deprecated. Please use `name` instead.
/usr/local/lib/python3.12/dist-packages/mlflow/models/model.py:1209: FutureWarning: Saving scikit-learn models in the pickl
  flavor.save_model(path=local_path, mlflow_model=mlflow_model, **kwargs)
Registered model 'FlipkartSentimentModel' already exists. Creating a new version of this model...
Created version '2' of model 'FlipkartSentimentModel'.
0.9655172413793104
```

Step 8: Example Sentiment Prediction

The trained model is used to predict sentiment for a new review.

```
def predict_sentiment(review):
    pred = best_pipeline.predict([review])[0]
    return 'Positive (User is satisfied)' if pred == 1 else 'Negative (User is not satisfied)'

predict_sentiment('The shuttle quality is very poor and broke within two days')
```

```
'Negative (User is not satisfied)'
```

Step 9: Prefect Workflow Orchestration

Prefect is used to orchestrate the entire training workflow.

```
@task
def train_and_evaluate():
    return f1

@flow
def mlops_flow():
    score = train_and_evaluate()
    print('Workflow completed with F1 score:', score)

mlops_flow()
```

Conclusion

This notebook implements a complete MLOps workflow including model training, hyperparameter tuning, experiment tracking, model registry, and orchestration. The system can be extended further for deployment.