In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import classification_report
# Simulated dataset
data = pd.DataFrame({
    'engagement_score': np.random.randint(30, 100, 100),
    'pages_visited': np.random.randint(1, 15, 100),
    'time_spent': np.random.randint(50, 300, 100),
    'lead_notes': np.random.choice([
        "Requested pricing and product demo",
        "Follow-up needed next week",
        "High budget and ready to sign",
        "Just browsing, low interest",
        "Mentioned competition; hesitant"
    ], 100),
    'priority': np.random.choice(['High', 'Medium', 'Low'], 100)
})
# Encode target variable
le = LabelEncoder()
data['priority_encoded'] = le.fit_transform(data['priority'])
# Separate features and target
X = data[['engagement_score', 'pages_visited', 'time_spent', 'lead_notes']]
y = data['priority_encoded']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# NLP Vectorizer
tfidf = TfidfVectorizer()

# Combine structured and unstructured features
preprocessor = ColumnTransformer(
    transformers=[
        ('text', tfidf, 'lead_notes'),
        ('num', 'passthrough', ['engagement_score', 'pages_visited', 'time_spent'])
    ]
)

# Pipeline with Random Forest
model = make_pipeline(preprocessor, RandomForestClassifier(n_estimators=100, random
# Train
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Report
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
new_lead = pd.DataFrame([{
    'engagement_score': 88,
    'pages_visited': 9,
    'time_spent': 210,
    'lead_notes': "Very interested, wants enterprise solution soon"
}])

pred_encoded = model.predict(new_lead)[0]
pred_label = le.inverse_transform([pred_encoded])[0]
print("🎯 Predicted Priority:", pred_label)
```