```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Step 2: Sample Dataset
data = {
    "query": [
        "I can't access my account",
        "How do I reset my password?",
        "Please explain my last bill",
        "I want to change my plan",
        "The app keeps crashing",
        "Need help with network issue",
        "Billing error on my statement",
        "Unable to make a call",
        "Want to update my payment method",
        "App is very slow to load"
    ],
    "category": [
        "Technical", "Technical", "Billing", "General",
        "Technical", "Technical", "Billing", "Technical",
        "Billing", "Technical"
    ]
}
df = pd.DataFrame(data)

# Step 3: Preprocessing
df['query_length'] = df['query'].apply(len)

def clean_text(text):
    text = text.lower()
    text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)
    text = re.sub(r"\d+", "", text)
    return text

df['clean_query'] = df['query'].apply(clean_text)

le = LabelEncoder()
df['label'] = le.fit_transform(df['category'])

# Step 4: TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=100)
X_tfidf = tfidf.fit_transform(df['clean_query']).toarray()

# Combine Features
X = np.hstack((X_tfidf, df[['query_length']].values))
y = df['label']

# Step 5: Dimensionality Reduction
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X)

# Step 6: Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)

# Step 7: Model Training
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

xgb = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)

# Step 8: Evaluation
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))

print("XGBoost Classification Report:")
print(classification_report(y_test, y_pred_xgb))

print("Confusion Matrix (XGBoost):")
print(confusion_matrix(y_test, y_pred_xgb))
```