```
from flask import Flask, request, render_template, redirect, url_for
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
import shap
import matplotlib.pyplot as plt

app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Ensure the upload directory exists
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']

    if file.filename == '':
        return "No selected file"

    filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
    file.save(filepath)

    result = process_csv(filepath)
    return render_template('result.html', result=result)

def process_csv(filepath):
    df = pd.read_csv(filepath)

    # Basic preprocessing
    df = df.dropna()

    # Encode categorical features
    label_encoders = {}
    for col in df.select_dtypes(include='object').columns:
        if col != 'Churn':
            le = LabelEncoder()
            df[col] = le.fit_transform(df[col])
            label_encoders[col] = le

    # Encode target
```

```
    # Encode target
    target_encoder = LabelEncoder()
    df['Churn'] = target_encoder.fit_transform(df['Churn'])

    # Features and target
    X = df.drop('Churn', axis=1)
    y = df['Churn']

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42)

    # Scale
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Train model
    model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
    model.fit(X_train_scaled, y_train)

    # Evaluate
    y_pred = model.predict(X_test_scaled)
    report = classification_report(y_test, y_pred, output_dict=True)
    report_text = classification_report(y_test, y_pred)

    # SHAP feature importance
    explainer = shap.Explainer(model, X_train_scaled)
    shap_values = explainer(X_test_scaled)

    shap.summary_plot(shap_values, X_test, plot_type='bar', show=False)
    plt.tight_layout()
    shap_path = os.path.join('static', 'shap_summary.png')
    plt.savefig(shap_path)
    plt.clf()

    return {
        'report': report_text,
        'shap_plot': shap_path
    }

if __name__ == '__main__':
    app.run(debug=True)


 ••• ain__'

is is a development server. Do not use it in a production deployment. Use a production WSGI
0.0.1:5000
C to quit
ng with stat
```