```
'''
from flask import Flask, render_template, request, redirect, url_for, session, send_file
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from scipy import stats as scipy_stats
import statistics as stats_py
import warnings
warnings.filterwarnings("ignore")

app = Flask(__name__)
app.secret_key = 'secret_key'
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Preprocessing Functions
def handle_missing_values(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].fillna("Missing")
        else:
            df[col] = df[col].fillna(df[col].mean())
    return df

def encode_categorical(df):
    label_encoders = {}
    for col in df.select_dtypes(include='object').columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col].astype(str))
```

```
        label_encoders[col] = le
    return df

def normalize_data(df, method='standard'):
    scaler = StandardScaler() if method == 'standard' else MinMaxScaler()
    numeric_cols = df.select_dtypes(include=np.number).columns
    df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
    return df

def remove_outliers(df, z_thresh=3):
    numeric_df = df.select_dtypes(include=np.number)
    z_scores = np.abs(scipy_stats.zscore(numeric_df))
    return df[(z_scores < z_thresh).all(axis=1)]

def feature_selection(df, target_col):
    X = df.drop(columns=[target_col])
    y = df[target_col]
    selector = SelectKBest(score_func=f_classif, k=5)
    X_new = selector.fit_transform(X, y)
    selected_cols = X.columns[selector.get_support()]
    return df[selected_cols.to_list() + [target_col]]

def text_preprocess(df):
    for col in df.select_dtypes(include='object').columns:
        df[col] = df[col].str.lower().str.replace(r'[^\w\s]', '', regex=True)
    return df

def show_statistics(df):
    numeric = df.select_dtypes(include=np.number)
    stats_summary = {
        "mean": numeric.mean().round(2).to_dict(),
```

```
        "median": numeric.median().round(2).to_dict(),
        "mode": numeric.mode().iloc[0].round(2).to_dict(),
        "std_dev": numeric.std().round(2).to_dict(),
        "outliers": {col: (np.abs(scipy_stats.zscore(numeric[col])) > 3).sum() for col in numeric.columns}
    }
    return stats_summary

def preprocess_pipeline(path, target_column=None):
    df = pd.read_csv(path)
    df = handle_missing_values(df)
    df = text_preprocess(df)
    df = encode_categorical(df)
    df = normalize_data(df)
    df = remove_outliers(df)
    if target_column and target_column in df.columns:
        df = feature_selection(df, target_column)
    stats_summary = show_statistics(df)
    output_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    df.to_csv(output_path, index=False)
    return df, stats_summary, output_path

# Routes
@app.route('/')
def home():
    return render_template('login.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    if username == 'admin' and password == 'admin':
```

```
            session['user'] = username
            return redirect(url_for('index'))
        return render_template('login.html', error="Invalid credentials")

@app.route('/index', methods=['GET', 'POST'])
def index():
    if 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        file = request.files['file']
        target_col = request.form.get('target_column', None)

        if file:
            file_path = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(file_path)
            df, stats_summary, output_file = preprocess_pipeline(file_path, target_col)
            session['preview_data'] = df.head().to_html(classes='table table-striped', index=False,
border=0)
            session['summary'] = stats_summary
            session['download_link'] = output_file
            save_plots(df)
            return redirect(url_for('result'))

    return render_template('index.html')

@app.route('/view_plots')
def view_plots():
    if 'user' not in session:
        return redirect(url_for('login'))
```

```
    plot_paths = [
        url_for('static', filename='plots/histogram.png'),
        url_for('static', filename='plots/scatter_plot.png'),
        url_for('static', filename='plots/correlation_heatmap.png')
    ]
    return render_template('view_plots.html', plot_paths=plot_paths)


@app.route('/download')
def download():
    file_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    return send_file(file_path, as_attachment=True)

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('home'))

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

    '''


from flask import Flask, render_template, request, redirect, url_for, session, send_file
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
```

```
from sklearn.feature_selection import SelectKBest, f_classif
from scipy import stats as scipy_stats
import warnings

warnings.filterwarnings("ignore")

app = Flask(__name__)
app.secret_key = 'secret_key'
UPLOAD_FOLDER = 'uploads'
PLOT_FOLDER = 'static/plots'

os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(PLOT_FOLDER, exist_ok=True)

# ---------------------- Preprocessing Functions ---------------------- #
def handle_missing_values(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].fillna("Missing")
        else:
            df[col] = df[col].fillna(df[col].mean())
    return df

def encode_categorical(df):
    for col in df.select_dtypes(include='object').columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col].astype(str))
    return df

def normalize_data(df, method='standard'):
    scaler = StandardScaler() if method == 'standard' else MinMaxScaler()
```

```
    numeric_cols = df.select_dtypes(include=np.number).columns
    df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
    return df

def remove_outliers(df, z_thresh=3):
    numeric_df = df.select_dtypes(include=np.number)
    z_scores = np.abs(scipy_stats.zscore(numeric_df))
    return df[(z_scores < z_thresh).all(axis=1)]

def feature_selection(df, target_col):
    X = df.drop(columns=[target_col])
    y = df[target_col]
    selector = SelectKBest(score_func=f_classif, k=5)
    X_new = selector.fit_transform(X, y)
    selected_cols = X.columns[selector.get_support()]
    return df[selected_cols.to_list() + [target_col]]

def text_preprocess(df):
    for col in df.select_dtypes(include='object').columns:
        df[col] = df[col].str.lower().str.replace(r'[^\w\s]', '', regex=True)
    return df

def show_statistics(df):
    numeric = df.select_dtypes(include=np.number)
    stats_summary = {
        "mean": {k: float(v) for k, v in numeric.mean().round(2).to_dict().items()},
        "median": {k: float(v) for k, v in numeric.median().round(2).to_dict().items()},
        "mode": {k: float(v) for k, v in numeric.mode().iloc[0].round(2).to_dict().items()},
        "std_dev": {k: float(v) for k, v in numeric.std().round(2).to_dict().items()},
        "outliers": {k: int(v) for k, v in {col: (np.abs(scipy_stats.zscore(numeric[col])) > 3).sum() for col in
numeric.columns}.items()}
```

```
    }
    return stats_summary


def save_plots(df):
    numeric_cols = df.select_dtypes(include=np.number).columns

    # Histogram
    df[numeric_cols].hist(figsize=(10, 8))
    plt.tight_layout()
    plt.savefig(os.path.join(PLOT_FOLDER, 'histogram.png'))
    plt.close()

    # Scatter Plot
    if len(numeric_cols) >= 2:
        sns.scatterplot(data=df, x=numeric_cols[0], y=numeric_cols[1])
        plt.savefig(os.path.join(PLOT_FOLDER, 'scatter_plot.png'))
        plt.close()

    # Correlation Heatmap
    if len(numeric_cols) > 1:
        plt.figure(figsize=(10, 8))
        sns.heatmap(df[numeric_cols].corr(), annot=True, cmap="coolwarm")
        plt.savefig(os.path.join(PLOT_FOLDER, 'correlation_heatmap.png'))
        plt.close()

def preprocess_pipeline(path, target_column=None):
    df = pd.read_csv(path)
    df = handle_missing_values(df)
    df = text_preprocess(df)
    df = encode_categorical(df)
```

```
    df = normalize_data(df)
    df = remove_outliers(df)
    if target_column and target_column in df.columns:
        df = feature_selection(df, target_column)
    stats_summary = show_statistics(df)
    output_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    df.to_csv(output_path, index=False)
    return df, stats_summary, output_path

# ---------- Routes ---------- #
@app.route('/')
def home():
    return render_template('login.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    if username == 'admin' and password == 'admin':
        session['user'] = username
        return redirect(url_for('index'))
    return render_template('login.html', error="Invalid credentials")

@app.route('/index', methods=['GET', 'POST'])
def index():
    if 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        file = request.files['file']
        target_col = request.form.get('target_column', None)
```

```
        if file:
            file_path = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(file_path)
            df, stats_summary, output_file = preprocess_pipeline(file_path, target_col)
            session['preview_data'] = df.head().to_html(classes='table table-striped', index=False,
border=0)
            session['summary'] = stats_summary
            session['download_link'] = output_file
            save_plots(df)
            return redirect(url_for('result'))

    return render_template('index.html')

@app.route('/result')
def result():
    if 'user' not in session:
        return redirect(url_for('login'))

    return render_template('result.html',
                           summary=session.get('summary'),
                           table=session.get('preview_data'),
                           download_link=url_for('download'))

@app.route('/view_plots')
def view_plots():
    if 'user' not in session:
        return redirect(url_for('login'))

    plot_paths = [
        url_for('static', filename='plots/histogram.png'),
```

```
        url_for('static', filename='plots/scatter_plot.png'),
        url_for('static', filename='plots/correlation_heatmap.png')
    ]
    return render_template('view_plots.html', plot_paths=plot_paths)

@app.route('/download')
def download():
    file_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    return send_file(file_path, as_attachment=True)

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('home'))

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

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Upload CSV</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
    <style>
        body {
            margin: 0;
```

```
    padding: 0;
    background-image: url('{{ url_for('static', filename='b.jpg') }}'); /* Replace with your image */
    background-size: cover;
    background-position: center;
    font-family: Arial, sans-serif;
}

.container {
    background: rgba(255, 255, 255, 0.9);
    padding: 40px;
    width: 400px;
    margin: 100px auto;
    border-radius: 12px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
    text-align: center;
}

.container h2 {
    margin-bottom: 25px;
    color: #333;
}

.container input[type="file"],
.container input[type="text"] {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #bbb;
    border-radius: 5px;
}
```

```
        .container button {
            width: 100%;
            padding: 10px;
            background-color: #27ae60;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        .container button:hover {
            background-color: #219150;
        }

        .container a {
            display: inline-block;
            margin-top: 15px;
            text-decoration: none;
            color: #2980b9;
        }

        .container a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Upload CSV File for Data Preprocessing</h2>
        <form action="/index" method="POST" enctype="multipart/form-data">
            <input type="file" name="file" required><br><br>
```

```
        <input type="text" name="target_column" placeholder="Optional: Enter target column"><br><br>
        <button type="submit">Upload</button>
    </form>
    <br>
    <a href="/logout">Logout</a>
  </div>
</body>
</html>
```

LOGIN.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      background-image: url('{{ url_for('static', filename='a.jpg') }}'); /* Replace with your image path */
      background-size: cover;
      background-position: center;
      font-family: Arial, sans-serif;
    }

    .login-container {
      background: rgba(255, 255, 255, 0.9);
      padding: 40px;
      width: 300px;
```

```
    margin: 100px auto;
    border-radius: 10px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
    text-align: center;
}

.login-container h2 {
    margin-bottom: 20px;
    color: #333;
}

.login-container input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #aaa;
    border-radius: 5px;
}

.login-container button {
    width: 100%;
    padding: 10px;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.login-container button:hover {
    background-color: #2980b9;
```

```
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Login to Data Preprocessing Tool</h2>
        <form action="/login" method="POST">
            <input type="text" name="username" placeholder="Username" required><br>
            <input type="password" name="password" placeholder="Password" required><br>
            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

RESULT.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Preprocessing Results</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body {
            margin: 0;
            padding: 0;
            background-image: url('{{ url_for('static', filename='c.jpg') }}'); /* Update with your actual
image */
            background-size: cover;
            background-position: center;
```

```
        background-attachment: fixed;
        font-family: Arial, sans-serif;
    }

    .container {
        background: rgba(255, 255, 255, 0.95);
        padding: 40px;
        border-radius: 12px;
        margin-top: 50px;
        box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
    }

    h2, h3 {
        color: #2c3e50;
    }

    .btn {
        width: 100%;
        margin-top: 10px;
    }

    table {
        background-color: #fff;
    }

    .table th, .table td {
        vertical-align: middle;
    }
  </style>
</head>
<body>
```

```
<div class="container">
    <h2 class="text-center">Preprocessing Results</h2>

    <!-- Statistics Summary Table -->
    <h3>Statistics Summary</h3>
    <table class="table table-bordered">
        <thead>
            <tr>
                <th>Metric</th>
                <th>Value</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td><strong>Mean</strong></td>
                <td>{{ summary['mean'] }}</td>
            </tr>
            <tr>
                <td><strong>Median</strong></td>
                <td>{{ summary['median'] }}</td>
            </tr>
            <tr>
                <td><strong>Mode</strong></td>
                <td>{{ summary['mode'] }}</td>
            </tr>
            <tr>
                <td><strong>Standard Deviation</strong></td>
                <td>{{ summary['std_dev'] }}</td>
            </tr>
            <tr>
```

```
            <td><strong>Outliers</strong></td>
            <td>{{ summary['outliers'] }}</td>
          </tr>
        </tbody>
    </table>

    <!-- Data Preview Table -->
    <h3>Data Preview</h3>
    <div class="table-responsive">
        {{ table|safe }}
    </div>

    <!-- Download Processed CSV -->
    <div>
        <a href="{{ download_link }}" class="btn btn-primary">Download Processed CSV</a>
    </div>

    <!-- Button to View Plots -->
    <div class="mt-3">
        <a href="{{ url_for('view_plots') }}" class="btn btn-secondary">View Plots</a>
    </div>

    <!-- Navigation to Next Page -->
    <div class="mt-3">
        <a href="{{ url_for('index') }}" class="btn btn-secondary">Back to Upload Page</a>
    </div>
</div>

</body>
</html>
```

VIEWPLOTS.HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Generated Plots</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      font-family: Arial, sans-serif;
      text-align: center;
      background-image: url('{{ url_for("static", filename="d.jpg") }}'); /* Replace with your actual image */
      background-size: cover;
      background-position: center;
      background-attachment: fixed;
      color: #333;
    }

    .content-wrapper {
      background-color: rgba(255, 255, 255, 0.95);
      margin: 50px auto;
      padding: 30px;
      width: 90%;
      max-width: 1000px;
      border-radius: 12px;
      box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
    }

    h1 {
      color: #2c3e50;
```

```
        }

        h3 {
            margin-top: 30px;
            color: #444;
        }

        img {
            width: 90%;
            max-width: 600px;
            margin: 20px 0;
            border: 1px solid #ccc;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }

        a.button {
            padding: 10px 20px;
            background: #007BFF;
            color: white;
            text-decoration: none;
            border-radius: 5px;
            display: inline-block;
            margin-top: 30px;
        }

        a.button:hover {
            background: #0056b3;
        }
    </style>
</head>
```

```
<body>
  <div class="content-wrapper">
    <h1>📊 Generated Plots</h1>
    {% for path in plot_paths %}
      <div>
        <h3>{{ path.split('/')[-1].replace('.png','').replace('_',' ').title() }}</h3>
        <img src="{{ url_for('static', filename='plots/' + path.split('/')[-1]) }}" alt="Plot">
      </div>
    {% endfor %}
    <a href="{{ url_for('index') }}" class="button">⬅️ Back to Upload</a>
  </div>
</body>
</html>
```