Python File:

```
#Importing all the necessary libraries for the project
from flask import Flask, render_template, request, redirect,url_for
from PIL import Image
from flask import flash
import numpy as np
import cv2
import urllib.request
import os
from werkzeug.utils import secure_filename
from werkzeug.datastructures import  FileStorage

#Creating the destination folder.
app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads/'
app.secret_key = "secret key"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024 #Max Dimension of a picture
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif']) #Setting the extension types if user uploads
another format by mistake

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/')
def home():
    return render_template('index.html')  #Routes to the "index.html" doc

#Uploading an image and setting exceptions
@app.route('/', methods=['POST'])
def upload_image():
    if 'file' not in request.files:
        flash('No image file selected')
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        flash('No image file selected')
        return redirect(request.url)

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
         #print('upload_image filename: ' + filename)
        flash('Image successfully uploaded')
        flash('The Original image along with the Edge Detection filters are displayed below:')

        #Image processing code
        image = cv2.imread(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        gradients_sobelx = cv2.Sobel(image, -1, 1, 0)
        gradients_sobely = cv2.Sobel(image, -1, 0, 1)
        gradients_sobelxy = cv2.addWeighted(gradients_sobelx, 0.5, gradients_sobely, 0.5, 0)
        gradients_laplacian = cv2.Laplacian(image, -1)
        canny_output = cv2.Canny(image, 80, 150)
        prewitt_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
        prewitt_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
        prewitt = cv2.addWeighted(prewitt_x, 0.5, prewitt_y, 0.5, 0)
        blurred = cv2.GaussianBlur(image, (5, 5), 0)

        # apply binary mask
        ret, binary = cv2.threshold(blurred, 200, 255, cv2.THRESH_BINARY)

        #Roberts Filter
        roberts_x = np.array([[0, 1], [-1, 0]], dtype=int)
        roberts_y = np.array([[1, 0], [0, -1]], dtype=int)
        x = cv2.filter2D(image, cv2.CV_16S, roberts_x)
        y = cv2.filter2D(image, cv2.CV_16S, roberts_y)
        absx = cv2.convertScaleAbs(x)
        absy = cv2.convertScaleAbs(y)
        roberts = cv2.addWeighted(absx, 0.5, absy, 0.5, 0)

        #Applying Mean Thresholding
        mean_threshold = cv2.adaptiveThreshold(blurred, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 11, 2)
    #     Apply SVM filter
    #     svm_filter = cv2.SVM()
    #     svm_filter.load('svm_filter.xml')
    #     svm_edges = svm_filter.predict(img)
```

```
    #     Apply K-Means filter
    #     kmeans_filter = cv2.KMeans()
    #     kmeans_filter.load('kmeans_filter.xml')
    #     kmeans_edges = kmeans_filter.predict(img)

    #    Combine the two filters
    #     edges = np.logical_or(svm_edges, kmeans_edges)

    #    #Convert the edges to an image
    #     edges = np.uint8(edges) * 255
```

```
        #Saving processed images back in the "static/upload" folder
        sobelx_filename = 'sobelx_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], sobelx_filename), gradients_sobelx)
        sobely_filename = 'sobely_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], sobely_filename), gradients_sobely)
        sobelxy_filename = 'sobelxy_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], sobelxy_filename), gradients_sobelxy)
        laplacian_filename = 'laplacian_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], laplacian_filename),
gradients_laplacian)
        canny_filename = 'canny_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], canny_filename), canny_output)

        prewittx_filename = 'prewitty_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], prewittx_filename), prewitt_x)
        prewitty_filename = 'prewittx_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], prewitty_filename), prewitt_y)
        blurred_filename = 'blurred_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], blurred_filename), blurred)
        meanthreshold_filename = 'meanthreshold_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], meanthreshold_filename),
mean_threshold)
        roberts_filename = 'roberts_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], roberts_filename), roberts)
        binary_filename = 'binary_' + filename
        cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], binary_filename), binary)
```

```
        #Rendering template with processed images.
        return render_template('index.html',
                               filename=filename,
                               sobelx_filename=sobelx_filename,
                               sobely_filename=sobely_filename,
                               sobelxy_filename=sobelxy_filename,
                               laplacian_filename=laplacian_filename,
                               canny_filename=canny_filename,
                               prewittx_filename=prewittx_filename,
                               prewitty_filename=prewitty_filename,
                               blurred_filename=blurred_filename,
                               meanthreshold_filename=meanthreshold_filename,
                               roberts_filename=roberts_filename,
                               binary_filename=binary_filename)

    #Roberts filter and Binary mask


    else:
        flash('Allowed image types are - png, jpg, jpeg, gif')
        return redirect(request.url)

#Display
@app.route('/display/<filename>')
def display_image(filename):
    #print('display_image filename: ' + filename)
    image = cv2.imread('static/image.jpg')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 100, 200)
    cv2.imwrite('static/edges.jpg', edges)
    return redirect(url_for('static', filename='uploads/' + filename), code=301)

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

HTML Code- filename- 'index.html'

```
<html>
<head>
<title>Face Edge Detection using Different Edge detection filters </title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
bootstrap.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
</head>
<body>
<p><h1 align="center">Face Edge Detection using Different Edge detection filters </h1></p>
<div class="container">
<div class="row">
    <h2>Select a file to upload</h2>
    <p>
        {% with messages = get_flashed_messages() %}
          {% if messages %}
            <ul>
            {% for message in messages %}
              <li>{{ message }}</li>
            {% endfor %}
            </ul>
          {% endif %}
        {% endwith %}
    </p>

    <form method="post" action="/" enctype="multipart/form-data">
        <dl>
            <p>
                <input type="file" name="file" class="form-control" autocomplete="off" required>
            </p>
        </dl>
        <p>
            <input type="submit" value="Submit" class="btn btn-info">
        </p>
    </form>
    {% if filename %}
    <div class="py-5">
        <div class="container">
          <div class="row">
            <div class="col-md-3">
              <div class="card">
                <div class="card-block">
                  <h4 class="card-title">Input Image</h4>
                  <img src="{{ url_for('display_image', filename=filename) }}" width="125"
height="175">
                </div>
              </div>
            </div>
            <div class="col-md-3">
              <div class="card">
                <div class="card-block">
                    <h4 class="card-title">Sobel X Filter</h4>
                    <img src="{{ url_for('display_image', filename= sobelx_filename) }}" width="125"
height="175">
                </div>
              </div>
            </div>
            <div class="col-md-3">
              <div class="card">
                <div class="card-block">
                    <h4 class="card-title">SobelY Transformation</h4>
                    <img src="{{ url_for('display_image', filename= sobely_filename) }}" width="125"
height="175">
                </div>
              </div>
            </div>
            <div class="col-md-3">
                <div class="card">
                  <div class="card-block">
                      <h4 class="card-title">SobelXY Edge Detection</h4>
                      <img src="{{ url_for('display_image', filename= sobelxy_filename) }}"
width="125" height="175">
                  </div>
                </div>
              </div>
          </div><br>
          <div class="row">

            <div class="col-md-3">
              <div class="card">
```

```
                <div class="card-block">
                    <h4 class="card-title">Laplacian Edge detection</h4>
                    <img src="{{ url_for('display_image', filename= laplacian_filename) }}"
width="125" height="175">
                </div>
              </div>
            </div>
            <div class="col-md-3">
              <div class="card">
                <div class="card-block">
                    <h4 class="card-title">Canny Edge Detection</h4>
                    <img src="{{ url_for('display_image', filename= canny_filename) }}" width="125"
height="175">
                </div>
              </div>
            </div>
            <div class="col-md-3">
                <div class="card">
                  <div class="card-block">
                      <h4 class="card-title">PrewittX Filter</h4>
                      <img src="{{ url_for('display_image', filename= prewittx_filename) }}"
width="125" height="175">
                  </div>
                </div>
              </div>
              <div class="col-md-3">
                <div class="card">
                  <div class="card-block">
                      <h4 class="card-title">Blurred Edge Detection</h4>
                      <img src="{{ url_for('display_image', filename= blurred_filename) }}"
width="125" height="175">
                  </div>
                </div>
              </div>
              <div class="col-md-3">
                  <div class="card">
                    <div class="card-block">
                        <h4 class="card-title">Mean Thresholding</h4>
                        <img src="{{ url_for('display_image', filename= meanthreshold_filename) }}"
width="125" height="175">
                    </div>
                </div>
              </div>
              <div class="col-md-3">
                  <div class="card">
                    <div class="card-block">
                        <h4 class="card-title">PrewittY Filter</h4>
                        <img src="{{ url_for('display_image', filename= prewitty_filename) }}"
width="125" height="175">
                    </div>
                </div>
              </div>
              <div class="col-md-3">
                  <div class="card">
                    <div class="card-block">
                        <h4 class="card-title">Roberts Filter Detection</h4>
                        <img src="{{ url_for('display_image', filename= roberts_filename) }}"
width="125" height="175">
                    </div>
                </div>
              </div>
              <div class="col-md-3">
                  <div class="card">
                    <div class="card-block">
                        <h4 class="card-title"> Binary Mask </h4>
                        <img src="{{ url_for('display_image', filename= meanthreshold_filename) }}"
width="125" height="175">
                    </div>
                </div>
              </div>
          </div>
        </div>
      </div>

    {% endif %}
</div>
</div>
</body>
</html>
```