SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS

(SCORE)

WINTER SEMESTER 2023-24

PMCA507P: MACHINE LEARNING LAB

PROJECT

SLOT: L25+L26

FACULTY NAME: Prof. ARUN PANDIAN J

NAME: DHINESH V

REG NO: 23MCA0118

NAME: SANJAY G

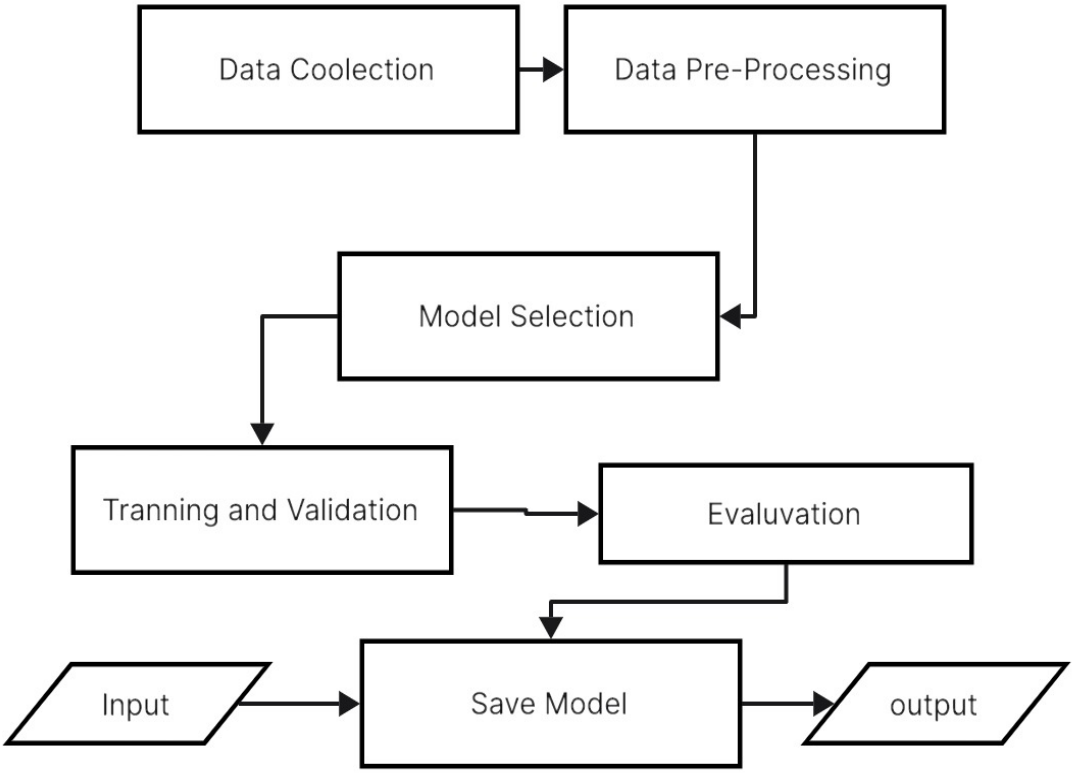
REG NO: 23MCA0245

NAME: GOWTHAM KS

REG NO: 23MCA0183

Q2: Classify the customer's credit rating (good or bad) based on their personal and bank account details.

FLOWCHART



CODE IMPLEMENTATION

Creditcardscore.ipynb

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder,MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import matplotlib.pyplot as plt
```

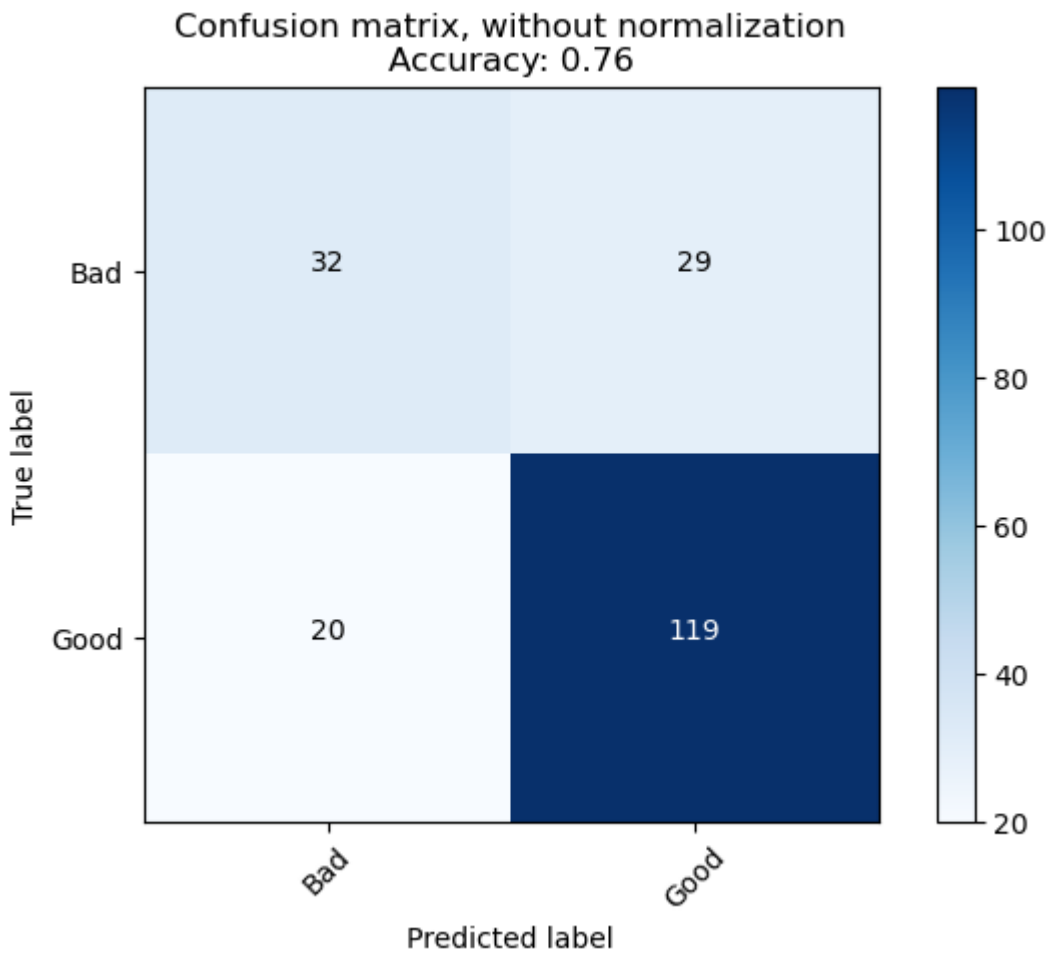
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score, f1_score

dataset = pd.read_csv('credit_rating.csv')
dataset.drop(columns='S.No',inplace=True)
selected=['CHK_ACCT','Duration','Balance in Savings A/C','History','Credit
Amount','Purpose of credit','Duration','Age','Install_rate','Real
Estate','Credit classification']
dataset=dataset[selected]
dataset = dataset.loc[:, ~dataset.columns.duplicated()]
dataset.head()

encoder = LabelEncoder()
categorical_columns =
dataset.select_dtypes(include=['object']).columns.tolist()
for ft in categorical_columns:
    dataset[ft]=encoder.fit_transform(dataset[ft])
# categ.pop(categ.index('Credit classification'))
# dataset.drop(columns=categ,inplace=True)

x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
y[1]
```

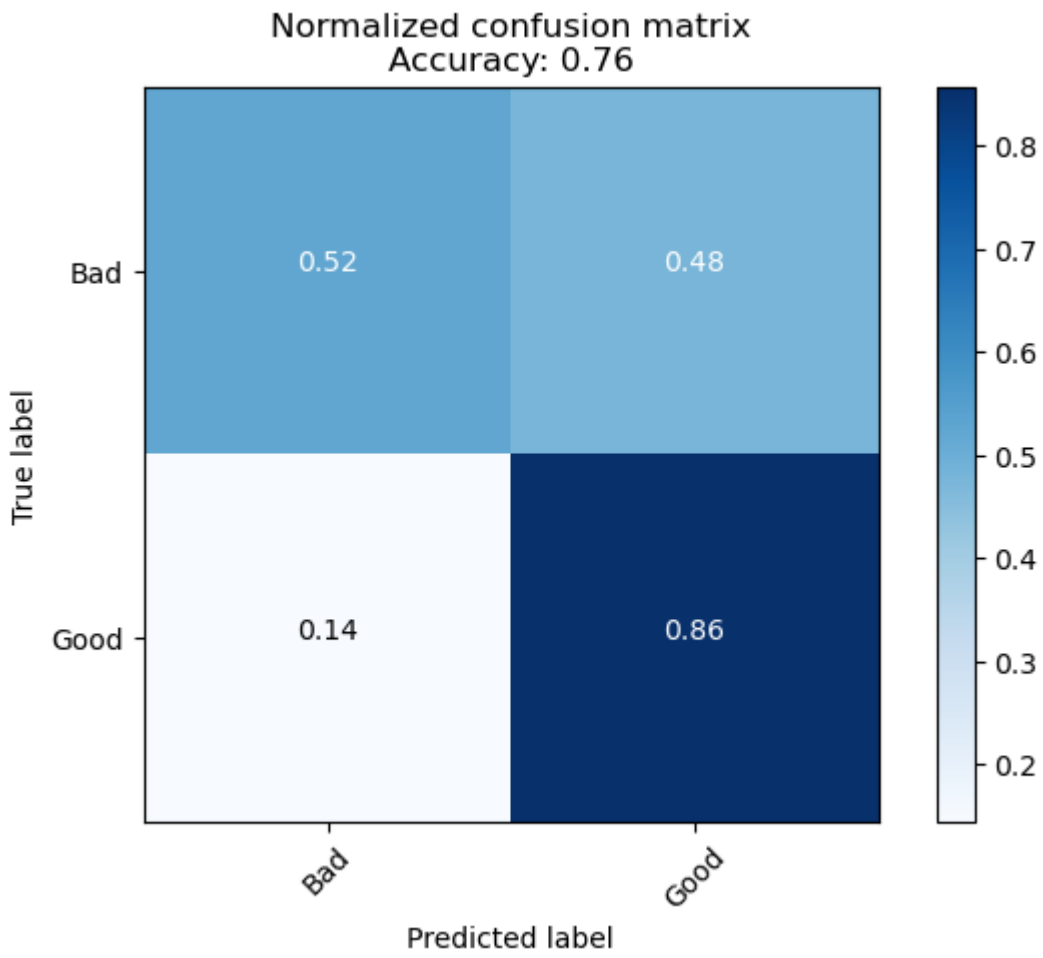
```
0
```

```
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.2,random_state=45)

sc = MinMaxScaler()
X_train_std = sc.fit_transform(x_train)
X_test_std = sc.fit_transform(x_test)
rf = RandomForestClassifier()
rf.fit(X_train_std, y_train)
# Predict
y_pred = rf.predict(X_test_std)
# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Assuming y_true contains the true labels and y_pred contains the predicted
labels

def plot_confusion_matrix(cm, classes, acc, normalize=False, title='Confusion
matrix', cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
```

```
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title + '\nAccuracy: {:.2f}'.format(acc))
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, format(cm[i, j], fmt),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Compute accuracy
accuracy = accuracy_score(y_test, y_pred)

# Define the class labels
class_names = ['Bad', 'Good']

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cm, classes=class_names, acc=accuracy, title='Confusion
matrix, without normalization')

# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cm, classes=class_names, acc=accuracy, normalize=True,
title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
Normalized confusion matrix
```


Confusion matrix, without normalization
Accuracy: 0.76
Bad
Good
True label
32
29
20
119
Bad
Good
Predicted label
100
80
60
40
20




```
dataset['Credit classification'].value_counts()
```

```
Credit classification
1    700
0    300
Name: count, dtype: int64
```

```
import joblib
from flask import Flask, render_template, request
joblib.dump(rf, 'randomForest.joblib')
```

```
['randomForest.joblib']
```

```
def make_prediction(chkacct, duration, balanceInSavings, history,creditAmount,
purposeofcredit, age, install_rate,RealEstate):
    model = joblib.load('randomForest.joblib')
    inputAttri = [[chkacct,duration, balanceInSavings,history,creditAmount,
purposeofcredit, age, install_rate,RealEstate]]
    inputAttri=sc.transform(inputAttri)
    prediction = model.predict(inputAttri)
    if prediction[0] == 1:
        return 'GOOD'
    else:
        return 'BAD'

make_prediction(1,   2,    9,    4, 6,    6,   22,    1,    3)
```

'GOOD'

```
from flask import Flask, render_template, request



app = Flask(__name__,static_url_path='/static')

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # Get form data
        chkacct = int(request.form['chkacct'])
        duration = int(request.form['duration'])
        balanceInSavings = int(request.form['balanceInSavings'])
        history = int(request.form['history'])
        creditAmount = int(request.form['creditAmount'])
        purposeofcredit = int(request.form['purposeofcredit'])
        age = int(request.form['age'])
        install_rate = int(request.form['install_rate'])
        RealEstate = int(request.form['RealEstate'])

        result =
make_prediction(chkacct,duration,balanceInSavings,history,creditAmount,purpose
ofcredit,age,install_rate,RealEstate)

        return render_template('result.html', prediction=result)

    return render_template('form.html')

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

```
* Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [26/Apr/2024 18:35:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Apr/2024 18:35:09] "GET
/static/wallpaperflare.com_wallpaper.jpg HTTP/1.1" 304 -
127.0.0.1 - - [26/Apr/2024 18:35:12] "POST / HTTP/1.1" 200 -
```

Form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Form</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100..900
;1,100..900&display=swap" rel="stylesheet">
    <style>
        input,select{
            border: 0.1px solid white;
            border-radius: 1%;
        }
        body{
            font-family: "Montserrat", sans-serif;
            color: white;
            background-image:
url('../static/wallpaperflare.com_wallpaper.jpg');
            background-position: center;
            background-size: cover;
        }
        body, html {
            height: 100%;
            margin: 0;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .container {
            width: 500px;
            border: 2px solid black;
            border-radius: 2.5%;
            padding: 60px 20px;
            display: flex;
            flex-direction: column;
            align-items: center;
            background-color: rgba(0, 0, 0,0.6);

        }

        .grid-tab {
            width: 100%;
            display: grid;
            grid-template-columns: repeat(2, 1fr);
            grid-gap: 20px;
        }

        form {
```

```
            display: flex;
            flex-direction: column;
            align-items: center;
        }

        .Buttons{
            background-color: rgb(41, 81, 189);
            color: aliceblue;
            font-weight: bold;
            border-style:none;
            padding: 10px;
            border-radius: 6px;
            vertical-align: top;
            transition:  .25s;
        }

    .Buttons:hover{
        color: black;
        padding-left: 30px;
        padding-right: 30px;
    }

    input:hover{
        background-color: blanchedalmond;
    }
    select:hover{
        background-color: blanchedalmond;
    }
    </style>
</head>
<body>
    <div class="container">
        <form action="/" method="post">
            <div class="grid-tab">
                <label for="chkacct">Checking Account:</label>
                <select id="chkacct" name="chkacct">
                    <option value="2">no-account</option>
                    <option value="0">0DM</option>
                    <option value="1">less-200DM</option>
                    <option value="3">over-200DM</option>
                </select>

                <label for="balanceInSavings">Balance in Savings A/C:</label>
                <select id="balanceInSavings" name="balanceInSavings">
                    <option value="0">unknown</option>
                    <option value="1">less100DM</option>
                    <option value="2">less1000DM</option>
                    <option value="3">over1000DM</option>
```

```
                </select>

                <label for="history">History:</label>
                <select id="history" name="history">
                    <option value="0">critical</option>
                    <option value="1">duly-till-now</option>
                    <option value="2">delay</option>
                    <option value="3">all-paid-duly</option>
                    <option value="4">bank-paid-duly</option>
                </select>

                <label for="purposeofcredit">Purpose of Credit:</label>
                <select id="purposeofcredit" name="purposeofcredit">
                    <option value="0">radio-tv</option>
                    <option value="1">education</option>
                    <option value="2">furniture</option>
                    <option value="3">new-car</option>
                    <option value="4">used-car</option>
                    <option value="5">business</option>
                    <option value="6">domestic-app</option>
                    <option value="7">repairs</option>
                    <option value="9">retraining</option>
                    <option value="8">others</option>
                </select>

                <label for="RealEstate">Real Estate:</label>
                <select id="RealEstate" name="RealEstate">
                    <option value="0">real-estate</option>
                    <option value="1">building-society</option>
                    <option value="3">car</option>
                    <option value="2">none</option>
                </select>

                <label for="duration">Duration (months):</label>
                <input type="number" id="duration" name="duration" min="1"
value="1">

                <label for="creditAmount">Credit Amount:</label>
                <input type="number" id="creditAmount" name="creditAmount"
value="0">

                <label for="age">Age:</label>
                <input type="number" id="age" name="age" min="18" value="18">

                <label for="install_rate">Installment Rate:</label>
                <input type="number" id="install_rate" name="install_rate"
value="1">
            </div>
```

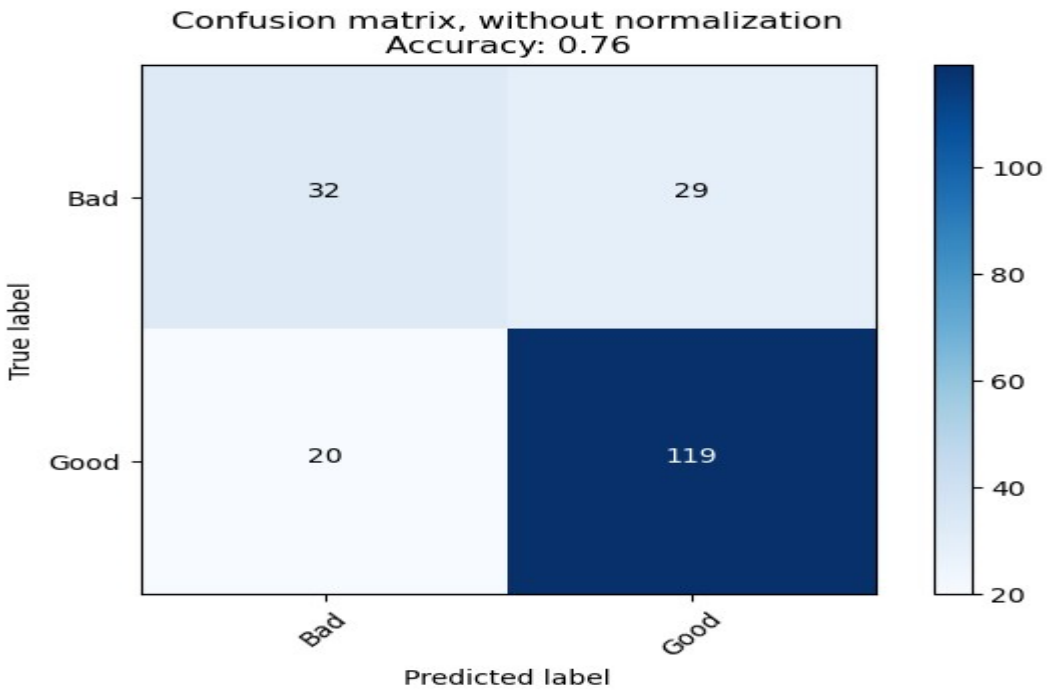
```
            <input class='Buttons' type="submit" value="Submit" style="margin-
top: 20px;">
        </form>
    </div>
</body>
</html>
```

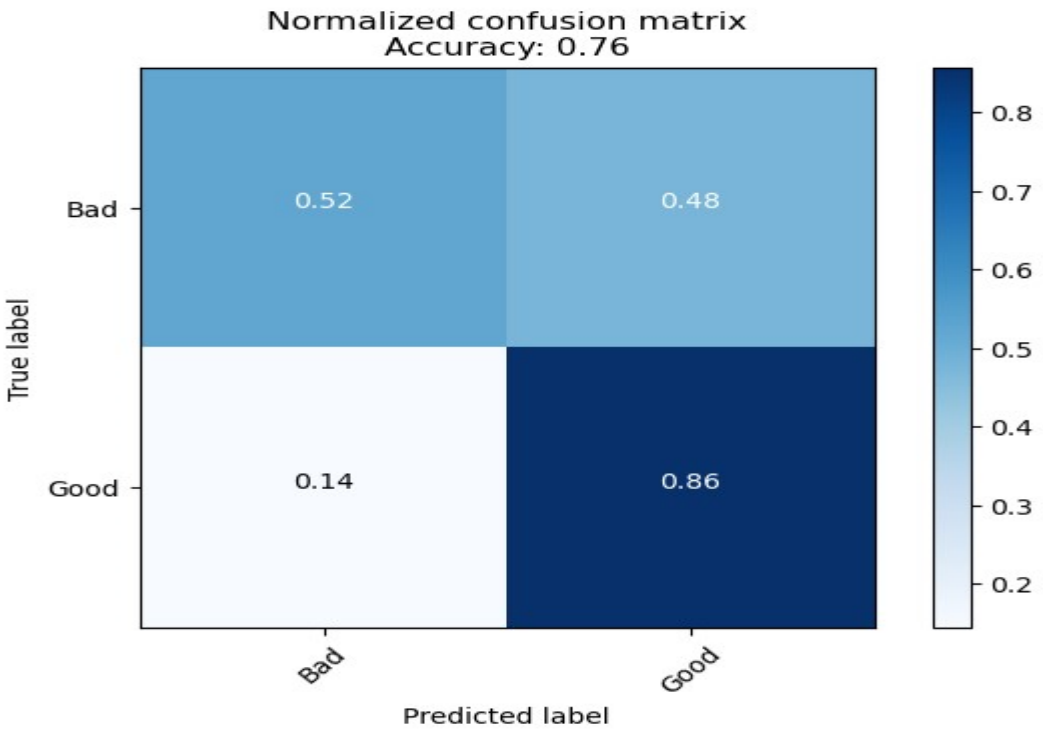
Result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result Page</title>
</head>
<body>
    <center>

        <p>Your Credit Score is {{prediction}}</p>
    </center>
</body>
</html>
```

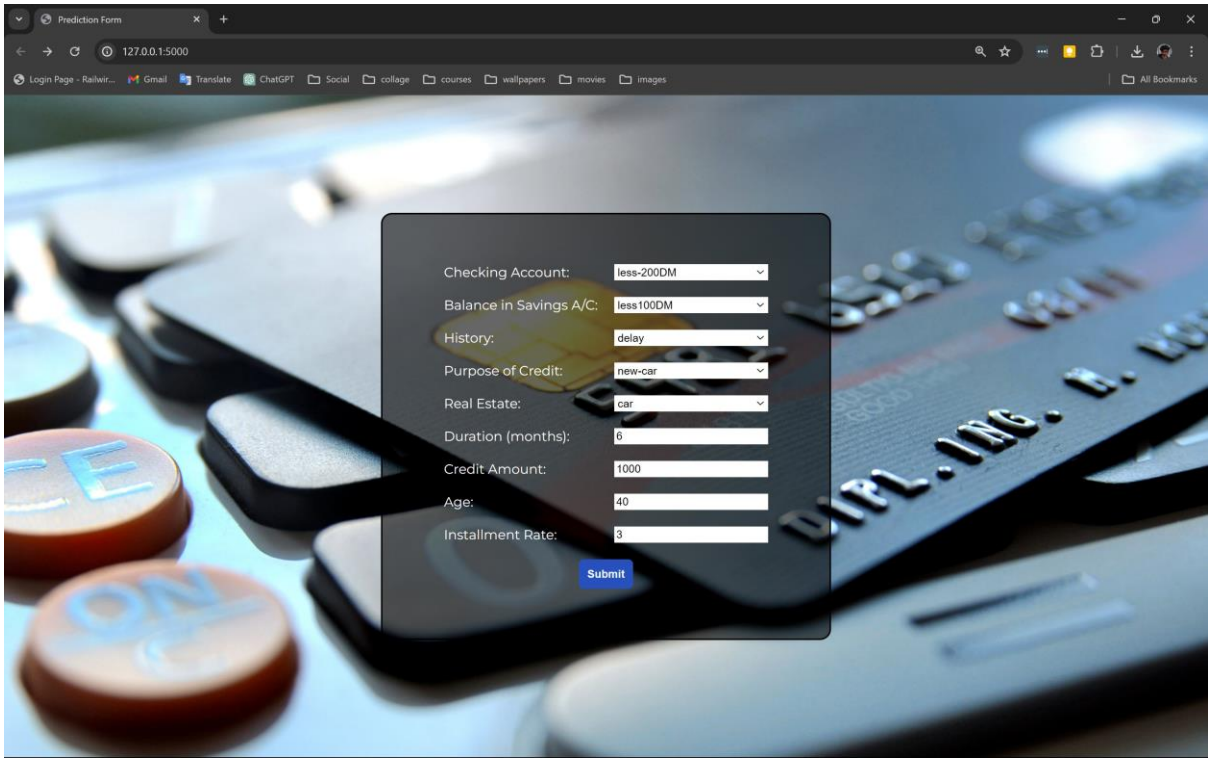
OUTPUT

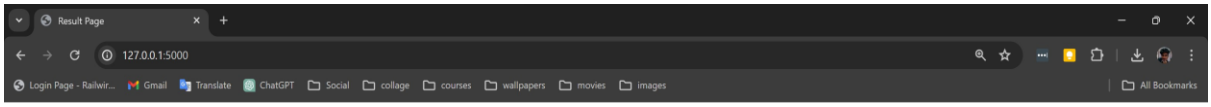
Confusion matrix





WebPage (http://127.0.0.1:5000)





Your Credit Score is GOOD