**SOFTWARE ENVIRONMENT**

The software environment defines the operating conditions, tools, and configurations used in the design, development, training, and deployment of the **Credit Card Fraud Detection Using Hybrid Classification Models** system. The environment setup ensures compatibility between libraries, reproducibility of results, and stable operation of the Flask-based web application.

**OPERATING SYSTEM:**

The system has been implemented and tested on Windows 10 (64-bit). However, the entire environment is cross-platform compatible and can also be executed on Linux distributions (e.g., Ubuntu 20.04) or macOS. Windows provides a user-friendly interface and strong support for Python-based development environments such as Anaconda, Jupyter Notebook, and Visual Studio Code.

**DEVELOPMENT TOOLS AND LIBRARIES:**

The project uses **Python 3.10.9** as the primary programming language. Python is chosen due to its extensive library support for machine learning, data processing, and visualization. The following tools and libraries form the backbone of the development environment:

|  |  |
| --- | --- |
| **Tool / Library** | **Purpose / Description** |
| **Flask** | Lightweight Python web framework used to build and deploy the fraud detection web application. |
| **NumPy** | Provides support for multi-dimensional arrays and high-performance mathematical operations. |
| **Pandas** | Facilitates data manipulation, cleaning, and feature engineering. |
| **Scikit-learn** | Core library for machine learning algorithms including Logistic Regression, SVM, and Random Forest. |
| **XGBoost** | Gradient boosting framework used for improving classification accuracy in the hybrid ensemble. |
| **Imbalanced-learn (SMOTE)** | Provides resampling techniques to handle class imbalance in the dataset. |
| **Matplotlib / Seaborn** | Used for data visualization and generating plots such as confusion matrices and ROC curves. |
| **Joblib / Pickle** | Libraries for saving and loading trained models for deployment. |

**Python:**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive**  **Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI**  **Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

**Getting Python**

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python [https://www.python.org](https://www.python.org/).

Windows Installation

Here are the steps to install Python on Windows machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
* To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
* Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

**First Python Program**

Let us execute programs in different modes of programming.

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python2.4.3(#1,Nov112010,13:34:43)

[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2

Type"help","copyright","credits"or"license"for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>>print"Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result −

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −

print"Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

**Flask Framework:**

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods −

|  |  |
| --- | --- |
| **Sr.No** | **Methods & Decription** |
| 1 | **GET**  Sends data in unencrypted form to the server. Most common method. |
| 2 | **HEAD**  Same as GET, but without response body. |
| 3 | **POST**  Used to send HTML form data to server. Data received by POST method is not cached by server. |
| 4 | **PUT**  Replaces all current representations of the target resource with the uploaded content. |
| 5 | **DELETE**  Removes all current representations of the target resource given by a URL |

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator. In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>

</form>

</body>

</html>

Now enter the following script in Python shell.

from flask importFlask, redirect,url\_for, request

app=Flask(\_\_name\_\_)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login():

ifrequest.method=='POST':

user=request.form['nm']

return redirect(url\_for('success',name= user))

else:

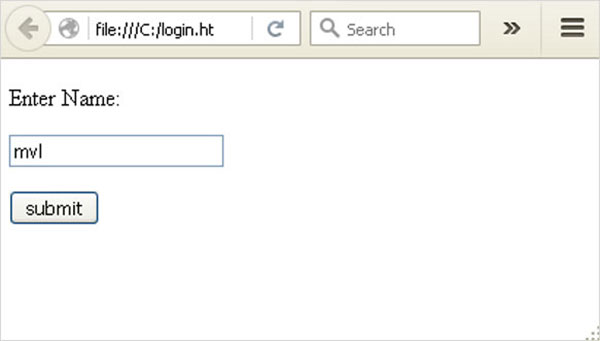
user=request.args.get('nm')

return redirect(url\_for('success',name= user))

if \_\_name\_\_ =='\_\_main\_\_':

app.run(debug =True)

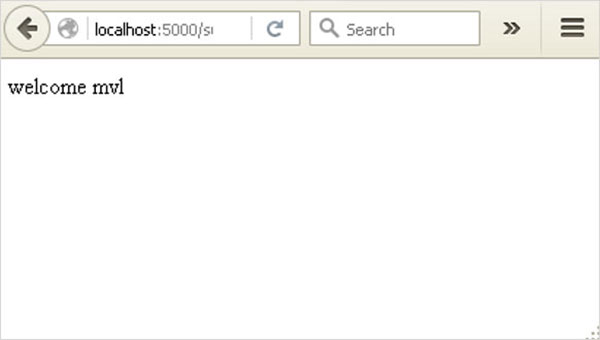
After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

Form data is POSTed to the URL in action clause of form tag.

[**http://localhost/login**](http://localhost/login) is mapped to the **login()** function. Since the server has received data by **POST** method, value of ‘nm’ parameter obtained from the form data is obtained by −

user = request.form['nm']

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to **‘GET’** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of ‘nm’ parameter is now obtained by −

User = request.args.get(‘nm’)

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to ‘nm’ parameter is passed on to ‘/success’ URL as before.

## **Python Install**

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\*Your Name*>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

## **Python Quickstart**

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\*Your Name*>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\*Your Name*>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

## The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\*Your Name*>python

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\*Your Name*>python  
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
 >>> print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\*Your Name*>python  
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
 >>> print("Hello, World!")  
 Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

>>> print("Hello, World!")  
 Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users\*Your Name*>python myfile.py

Python Indentations