### **Project Report Template: Login Anomaly Detection System**

### **1. Title Page**

**Project Title**: Login Anomaly Detection System Using Machine Learning and Flask

**Team members**: Sri Charan - 22139141

Paul Andrew - 22139130

Kaushal - 22139121

Janarthanan - 22139215

**Department**: BCA CTIS III

**College**: Vels Institute of Science, Technology & Advanced Studies

### **2. Abstract**

This project focuses on detecting suspicious attempts using machine learning techniques. It leverages the Isolation Forest algorithm to identify unusual patterns based on user IP, device, and location. The system uses Python with Flask for real-time detection of suspicious attempts.

### **3. Introduction**

With the increase in cyber-attacks and fraud, securing user logins has become a priority. This project builds an Anomaly Detection System that identifies suspicious attempts based on anomalies in the user’s IP address, device, and geographical location. Using machine learning algorithms, such as Isolation Forest, the system detects unusual attempts and provides real-time alerts. This tool will help enhance security and prevent unauthorized access to the system.

### **4. Objective**

The primary objective of this project is to develop a system that can:

1. Detect suspicious attempts using machine learning models.
2. Provide real-time alerts for any detected anomalies.
3. Build a web application using Flask to demonstrate the system’s functionality.
4. Ensure the system can be easily deployed and tested with user data.

### **5. System Architecture**

Flow of data:  
 **User Input** → **Data Preprocessing** → **Model Prediction** → **Suspicious Detection** → **Alert Generation**

The system consists of several components:

* **Frontend (Web Interface)**: A user-friendly form where IP, device, and location data are input.
* **Backend (Flask Server)**: Processes the user input, passes it to the machine learning model, and returns the results.
* **Model**: The Isolation Forest model is used to detect anomalies based on past login data.
* **Alerts**: The system will send an alert if suspicious patterns are detected.

### **6. Methodology**

**1. Data Collection and Preprocessing:**

* We used a sample CSV file (login\_data.csv) that contains the user login data (IP, device, location, and timestamp) and we used this data to train the model to detect suspicious patterns.
* This data was preprocessed to handle missing values and encoded categorical variables (device and location) using Label Encoding.

**2. Machine Learning Model:**

* The Isolation Forest algorithm was used to detect anomalous attempts. It is particularly useful for identifying rare events in large datasets.

**3. Web Application (Flask):**

* The backend of the system was built using Python with Flask, allowing users to input their login data through a web interface.

**4. Alert System:**

* If an anomaly is detected, the system triggers an alert to notify in the web browser of the suspicious attempt.

### **7. Code Implementation**

#### **7.1 Data Preprocessing**

import pandas as pd

from sklearn.preprocessing import LabelEncoder

# Load the data

df = pd.read\_csv('login\_data.csv')

# Label encode device and location columns

device\_encoder = LabelEncoder()

location\_encoder = LabelEncoder()

df['device'] = device\_encoder.fit\_transform(df['device'])

df['location'] = location\_encoder.fit\_transform(df['location'])

# Save the encoders for future use

import pickle

pickle.dump(device\_encoder, open('device\_encoder.pkl', 'wb'))

pickle.dump(location\_encoder, open('location\_encoder.pkl', 'wb'))

**Explanation**: This code loads the data, encodes the categorical columns (device and location) into numeric values using LabelEncoder, and saves the encoders for future use in model inference.

#### **7.2 Machine Learning Model (Isolation Forest)**

from sklearn.ensemble import IsolationForest

# Train the Isolation Forest model

model = IsolationForest(contamination=0.05)

model.fit(df[['IP', 'device', 'location']])

# Save the trained model

import pickle

pickle.dump(model, open('model.pkl', 'wb'))

**Explanation**: This code trains an Isolation Forest model to detect anomalies in the login data. It uses the encoded IP, device, and location columns for training. The model is saved for future use.

#### 

#### 

#### **7.3 Flask Application**

from flask import Flask, render\_template, request

import pickle

# Load the trained model and encoders

model = pickle.load(open('model.pkl', 'rb'))

device\_encoder = pickle.load(open('device\_encoder.pkl', 'rb'))

location\_encoder = pickle.load(open('location\_encoder.pkl', 'rb'))

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

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

def index():

if request.method == 'POST':

ip = request.form['ip']

device = request.form['device']

location = request.form['location']

# Preprocess input

device\_encoded = device\_encoder.transform([device])[0]

location\_encoded = location\_encoder.transform([location])[0]

# Predict anomaly

prediction = model.predict([[ip, device\_encoded, location\_encoded]])

# Check for suspicious login

if prediction == -1:

return "❌ Suspicious Login Detected"

else:

return "✅ Normal Login"

return render\_template('index.html')

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

app.run(debug=True)

**Explanation**: This code creates a simple Flask web application that receives user input, preprocesses the input (encodes device and location), and uses the trained model to predict whether the login is suspicious.

### **8. Results**

Here are the results from testing the system with different inputs:

* **Normal Login:**
  + IP: 192.168.1.8, Device: Safari, Location: Chennai → ✅ Normal Login
* **Suspicious Login:**
  + IP: 10.0.0.1, Device: Opera, Location: Kochi → ❌ Suspicious Login Detected

### **9. Discussion and Conclusion**

The Anomaly Detection System provides a reliable way to detect unusual attempts based on user behavior. However, its performance could be improved with larger datasets and more features (e.g., time-based behavior). Future work could involve integrating email/SMS alerts using Twilio or SMTP for real-time notification.

### **Conclusion**

This project focused on building an **Anomaly Detection System** using machine learning, specifically the **Isolation Forest** algorithm. The system is designed to monitor user logins in real time and detect any suspicious activity based on patterns such as IP address, device, and location.

The major achievements of this project include:

1. **Data Preprocessing**: Effectively encoding and preparing the login data for training the model.
2. **Model Training**: Using the Isolation Forest algorithm to detect anomalies.
3. **Real-time Detection**: Implementing a Flask web application to allow real-time detection of suspicious attempts, providing an intuitive interface for inputting and testing login data.
4. **Model Evaluation**: Ensuring that the model correctly identifies both normal and suspicious attempts based on the training data.

Challenges such as ensuring correct encoding and handling missing data were encountered and overcome during the development process. The system works by flagging any suspicious attempt that does not match a known pattern, such as an unfamiliar IP address, device, or location, and out of range IP address providing an important tool for identifying potential security threats.

This project serves as a strong foundation for developing more advanced security systems that leverage machine learning to enhance cybersecurity and user data protection.