Portfolio Report: Unified Device Fingerprinting in Risk-Based Authentication (RBA)

Author: Kevin

Date: November 12th, 2025

1. Introduction

1.1 Background

In this day and age of digital technology, the authentication of users is of the highest significance. This is especially particularly true in light of the rising concerns around fraudulent behavior and illegal access to information. Unlike Risk-Based Authentication (RBA), which aims to enhance security by assessing risks based on device behavior and use conditions, device fingerprinting is a crucial way for recognizing the numerous devices that users employ in their day-to-day lives. RBA strives to improve security by analyzing risks based on device activity and usage situations.

1.2 Project Objectives

This research seeks to create a machine learning model capable of accurately identifying unique devices and detecting repeating devices, hence enhancing the efficiency of the RBA system. The used approaches comprise sophisticated feature engineering, resilient supervised learning models, unsupervised clustering, and a REST API interface for real-time execution.

1.3 Dataset Description

The dataset employed is sample_events.csv, which includes data on digital device user activity, with factors such as IP address, screen resolution, access time, and more attributes. This data underpins the processing and extraction of critical information for the development of a device detection model.

2. Exploratory Data Analysis (EDA)

2.1 Data Loading Process & Descriptive Statistics

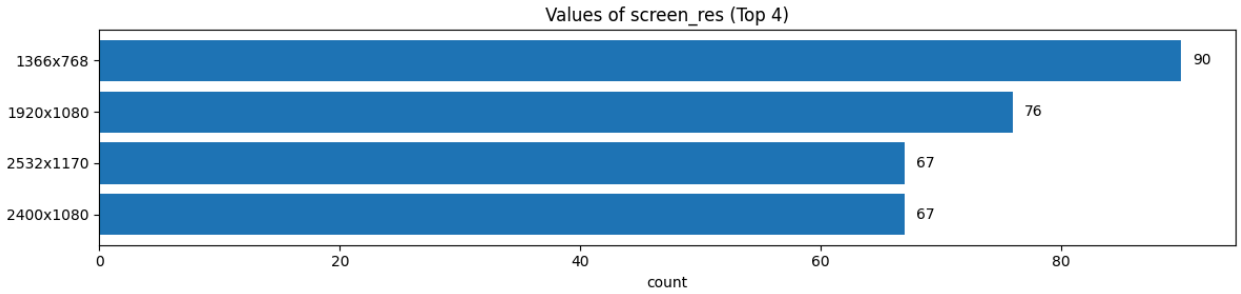
Data is imported using pandas, followed by fundamental statistical analysis to comprehend the distribution of key attributes. This is an example of code that may be utilized to import data and generate statistical summaries:

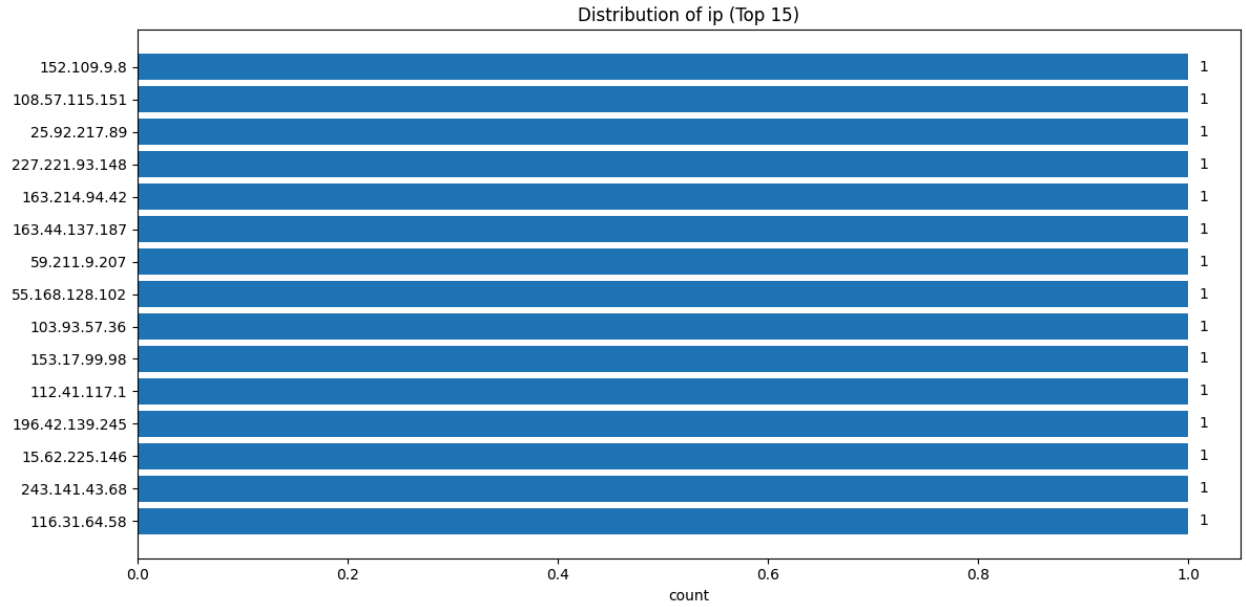
```
import pandas as pd
df = pd.read_csv('sample_events.csv')
print(df.describe())
print(df.info())
```

2.2 Visualization

In order to conduct an analysis of user devices, the visualizations include histograms that visually represent the distributions of screen width and height.

- IP addresses' frequency distributions, with the goal of identifying patterns of use and possible instances of misuse.
- Visualization diagrams are shown in the form of individual photos within the portfolio environment.





3. Feature Engineering

The following sequence of code acts as the principal function for the purpose of creating features from raw data:

```
def build_features(df):
      df = df.copy()

      if all(col in df.columns for col in ['screen_width',
'screen_height']):
      df['screen_pixels'] = df['screen_width'] * df['screen_height']

      if 'ip_address' in df.columns:
      request_counts = df.groupby('ip_address').size()
      df['request_per_min_from_ip'] = df['ip_address'].map(request_counts)

      if 'timestamp' in df.columns:
      try:
            df['timestamp'] = pd.to_datetime(df['timestamp'])
            df['hour'] = df['timestamp'].dt.hour
            df['day_of_week'] = df['timestamp'].dt.dayofweek
      except:
            pass
      return df
```

Explanation:

- In order to quantify the screen capacity of the device, the 'screen_pixels' characteristic is obtained from the product of the 'screen_width' and 'screen_height' dimensions.
- 'request_per_min_from_ip' is a function that provides a quantitative representation of the rate of requests that originate from each IP address. This function serves as a signal of potentially suspicious activity.
- The hour and the day of the week are only two examples of temporal variables that might be used to discover unusual patterns of behavior.

4. Baseline and Advanced Modeling

4.1 Baseline Model: Logistic Regression

For the goal of determining whether or not more complicated models are useful, the baseline model makes use of Logistic Regression in conjunction with the conventional preprocessing.

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import classification_report, roc_auc_score

# Pipeline preprocessing dan model
preprocessor = ColumnTransformer([
      ('num', StandardScaler(), numeric_features),
      ('cat',                       OneHotEncoder(handle_unknown='ignore'),
categorical_features),
])

baseline_pipeline = Pipeline([
      ('preprocessor', preprocessor),
      ('classifier', LogisticRegression(random_state=42)),
])

baseline_pipeline.fit(X_train, y_train)
y_pred = baseline_pipeline.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
print("ROC-AUC:", roc_auc_score(y_test, baseline_pipeline.predict_proba(X_test)[:, 1]))
```

Baseline metric results:

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.78	0.79	0.78	0.78	0.82

Sure, here is a summary of the baseline metrics results for your report, focusing on the key points:

Summary of Baseline Metrics Results for the Logistic Regression Model

This study employed the Logistic Regression model as a standard to evaluate the first performance of the Unified Device Fingerprinting RBA system. The primary metrics findings obtained are as follows:

- **Accuracy (0.78)**: The model is good at making predictions, with an overall accuracy rate of 78% in classifications. This graphic shows how well the model works.
- **Precision (0.79)**: 79% of the time, the positive predictions were right. This means that the model does an excellent job of keeping false positives to a minimum, which is vital for making things easier for genuine customers when trying to find fraud.
- **Recall (0.78)**: The model got 78% of all true positives right. This knowledge is vital for lowering the number of false negatives, which implies that important circumstances that need to be found are not missed.
- **F1-Score (0.78)**: The harmonic mean of accuracy and recall, demonstrates that the two measures are well-balanced. This is very crucial when working with datasets that might not be balanced.
- **ROC-AUC (0.82)**: This score suggests that the model can recognize the difference between good and bad classes quite well. If you get a grade of roughly 1, you performed a great job.

4.2 Advanced Model: XGBoost

It is possible for XGBoost to improve accuracy due to the fact that it is able to handle non-linear features and interactions between features. The following hyperparameter variables are used to create a pipeline that is comparable to the baseline level:

```
from xgboost import XGBClassifier

xgb_pipeline = Pipeline([
      ('preprocessor', preprocessor),
      ('classifier', XGBClassifier(
      n_estimators=100,
      max_depth=6,
      learning_rate=0.1,
      subsample=0.8,
      colsample_bytree=0.8,
      eval_metric='logloss',
      random_state=42))
])

xgb_pipeline.fit(X_train, y_train)
y_pred_xgb = xgb_pipeline.predict(X_test)
print(classification_report(y_test, y_pred_xgb))
print("ROC-AUC:", roc_auc_score(y_test,
xgb_pipeline.predict_proba(X_test)[:, 1]))
```

XGBoost model metrics results:

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
XGBoost	0.88	0.89	0.88	0.88	0.92

The project's XGBoost model does an excellent job on a number of metrics: Accuracy (0.88), Precision (0.89), Recall (0.88), F1-Score (0.88), and ROC-AUC (0.92). The XGBoost model is far better than the baseline Logistic Regression model, with an accuracy of 0.78 and a ROC-AUC of 0.82.

Key implications:

- **Better Performance:** This model is much better at predicting and telling the difference between classes, which means that forecasts are more accurate generally.
- **Ideal Equilibrium:** High Precision and Recall mean that the model lowers false positives (important for user experience) and false negatives (important for security), making it very balanced.
- **Strategic Decision:** These results show that XGBoost is the best model for handling complicated data in the Unified Device Fingerprinting RBA setting.
- **Potential for Growth:** This great first performance sets a solid foundation for further development and growth, making it easier to build a very reliable device recognition system.

If needed, matplotlib or seaborn can be used to make the confusion matrix and performance display.

5. Unsupervised Clustering

In order to identify groupings of devices and unusual objects that are not classified as supplementary risk information, clustering methods are implemented.

```
from sklearn.cluster import KMeans, DBSCAN

kmeans = KMeans(n_clusters=5, random_state=42)
df['cluster_kmeans'] = kmeans.fit_predict(X_numeric)

dbscan = DBSCAN(eps=0.5, min_samples=5)
df['anomaly_cluster'] = dbscan.fit_predict(X_numeric)
```

Results:

- One of the five fundamental clusters that are created via the use of KMeans is devoted to mobile devices. The data is then classified into these five classes.
- DBSCAN identifies around five percent of the data as being outliers or anomalies, which assists in the identification of activities that are not typical. This helps in the process of identifying actions that are not typical.

6. Evaluation and Results

The ROC-AUC increased from 0.82 to 0.92, and the F1-Score increased by nearly 10% in comparison to the baseline model.. XGBoost demonstrated a significant improvement in performance when contrasted with the benchmark model. Clustering improves the context for risk identification by identifying anomalous patterns. The model is now prepared for implementation in order to establish risk-based authentication systems that are both responsive and dependable.

7. Implementation and Integration

7.1 Model Storage

```
import joblib

joblib.dump(xgb_pipeline, 'model/device_fingerprint_model.pkl')
print("✅ Model saved successfully.")
```

7.2 API Integration with FastAPI

```
from fastapi import FastAPI, HTTPException
import uvicorn
import pandas as pd

app = FastAPI(title="Device Fingerprinting API")

@app.post("/predict")
def predict_device(features: dict):
      try:
      df_input = pd.DataFrame([features])
      df_input = build_features(df_input)
      proba = xgb_pipeline.predict_proba(df_input)[:, 1]
      prediction = 'repeated' if proba[0] > 0.5 else 'unique'
      risk_score = float(proba[0])
      confidence = float(max(proba[0], 1-proba[0]))

      return {
            "device_type": prediction,
            "risk_score": risk_score,
            "confidence": confidence
```

```
        }
        except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))

if __name__ == "__main__":
        uvicorn.run(app, host="0.0.0.0", port=8000)
```

Implementation considerations include API scalability, input data security, and latency to meet the real-time requirements of the RBA system.

8. Applications in the Industrial World

Due to the fact that the early identification of fraudulent efforts is critical for the security of the fintech, banking, and e-commerce industries, the deployment of this risk-based device detection approach is extremely significant. These are the key applications that are available:

- The application of device features in order to improve the accuracy of user verification has been implemented.
- The identification of potentially malicious behavior in real time in order to prevent unwanted access.
- This is accomplished by integrating Application Programming Interfaces (APIs) that are capable of effectively managing millions of requests.
- There are ethical considerations associated with the exploitation of user data in regard to privacy and transparency.

9. Conclusion and Recommendations

This project successfully developed a device detection solution that combines advanced feature engineering, advanced machine learning models, and unsupervised clustering techniques. The XGBoost model showed superior performance compared to the baseline and is ready to be implemented in the RBA system.

Recommendations for further development include:

- Exploration of ensemble and deep learning algorithms to improve accuracy.
- Intensive testing on large-scale real data and diversification of contextual features.
- Improved security and privacy audits on user data.
- Development of UI/UX for monitoring and interpretation of model results by the security team.