**Project Tree for Nairobi Upper Hill Cyber Heist Investigation**

# Tools and Resources Needed

1. Machine Learning Frameworks (e.g., TensorFlow, PyTorch)
2. Programming Languages (primarily Python)
3. APIs - (VirusTotal API for malware detection)
4. Datasets as provided i.e., file signatures, headers, and footers dataset, IP address ranges

# Setup Your Environment

1. **Install Required Software:** Ensure Python is installed along with key libraries like Pandas, Scikit-learn, spaCy, NLTK, TensorFlow, and others relevant to your challenges.
2. **Prepare Your Data:** Organize the datasets and files you'll need for each challenge into accessible directories.

# Folder 1: Document Analysis

## Challenge 1: Natural Language Processing (NLP) for Cyber-Attack Investigation

**Objective:** Automate the extraction of relevant files using NLP.

**Tasks:**

1. ***Script Development***

* Develop a Python script to parse file contents.
* Implement NLP techniques to identify keywords ('MPESA', 'HAWALA', 'KAFIR', etc.).

1. ***File Extraction***

* Filter and extract files containing the specified keywords.
* Save extracted files to a directory named 'evidence'.

1. ***Reporting***

* Generate a CSV report listing the extracted files and their details (e.g., filename, path, keywords found).

1. ***Testing and Validation***

* Test the script on a sample dataset.
* Adjust and optimize based on test results.

**Algorithms:** Utilize NLP techniques like keyword search, regular expressions, or more advanced text analytics with Natural Language Toolkit (NLTK) or spaCy for more complex semantic processing.

**Libraries:** Python's NLTK and spaCy are excellent for text processing. Use Pandas for data manipulation and CSV file operations.

**Implementation Tips:**

* Pre-process text data to standardize formats (e.g., lowercasing, removing punctuation).
* Consider expanding keyword search capabilities using synonyms or related terms to increase the capture rate of relevant documents.
* Utilize Python's os and shutil modules for file operations like moving identified files to a specified directory.

## Challenge 2: File Format Classification

**Objective:** Automate categorization of file formats.

**Tasks:**

1. ***Model Design***

* Study and understand the provided file signatures dataset.
* Choose appropriate machine learning algorithms (e.g., SVM, Random Forest).

1. ***Model Training***

* Train the model using the dataset of file signatures.
* Validate accuracy with cross-validation techniques.

1. ***Deployment***

* Deploy the model to classify files into formats like PDF, DOCX, JPEG, HTML, etc.
* Integrate the model with the workflow to automatically classify incoming files.

1. ***Evaluation***

* Evaluate the model's performance and make necessary adjustments.

**Algorithms:** Decision Trees, Support Vector Machines (SVM), or Neural Networks are suitable for classifying files based on their signatures.

**Libraries:** Scikit-learn for machine learning models, Pandas for data handling, and NumPy for numerical operations.

**Implementation Tips:**

* Extract file signatures efficiently and ensure your dataset includes a diverse range of file types for training.
* Consider using a confusion matrix to evaluate the model's performance and identify which file types are frequently misclassified.
* Regularly update the model with new file types as they emerge.

# Folder 2: Malware Analysis

## Challenge 3: Malware Detection with Virus Total AI

**Objective:** Identify malware in files from RAM dump.

**Tasks:**

1. ***Integration with VirusTotal API***

* Set up and authenticate API access.
* Create a script to send files to VirusTotal for scanning.

1. ***Malware Analysis***

* Analyze the response from VirusTotal to determine if files are malicious.
* Compile results into a CSV format detailing the malware findings.

1. ***Model Development***

* Use findings to train a malware detection model.
* Implement machine learning techniques suitable for binary classification.

1. ***Testing***

* Test the model on a separate set of files to evaluate effectiveness.

**Algorithms:** Use API responses to train a binary classifier or use anomaly detection if dealing with unknown malware types.

**Libraries:** Requests library for API interaction, Pandas for handling the output data, and Scikit-learn for creating the classification model.

**Implementation Tips:**

* Automate the scanning process with batch requests to VirusTotal if dealing with large numbers of files.
* Store the API responses efficiently to avoid re-scanning files unnecessarily.
* Develop a robust error handling system to manage potential API rate limits or failures.

# Folder 3: File Integrity

## Challenge 4: Detecting File Manipulation and Text Classification for File Reconstruction

**Objective:** Detect file tampering and reconstruct files with corrupted extensions.

**Tasks:**

1. ***Anomaly Detection Model***

* Develop a model to analyze file headers and footers.
* Train the model to identify anomalies indicating unauthorized modifications.

1. ***Reconstruction Model***

* Use machine learning to match corrupted files with correct formats using the headers and footers dataset.
* Implement classification algorithms to predict the original file type.

1. ***Integration and Testing***

* Integrate both models into a single workflow.
* Conduct thorough testing to ensure models are functioning as expected.

**Algorithms:** For detecting file manipulation, use anomaly detection techniques. For file reconstruction, classification algorithms like Random Forest or Neural Networks can predict the correct file type from headers/footers.

**Libraries:** Scikit-learn for machine learning, Pandas for data manipulation, and possibly TensorFlow or PyTorch if using deep learning methods.

**Implementation Tips:**

* Develop a feature extraction method that accurately captures the essential characteristics of file headers and footers.
* Test the model with both tampered and untampered files to ensure it can reliably identify anomalies.
* For reconstruction, ensure that your training dataset covers all possible file types encountered in the environment.

# Folder 4: Visual Analysis

## Challenge 5: Facial Recognition in Video Clips

**Objective:** Develop an ML model for facial recognition to identify specific individuals.

**Tasks:**

1. ***Facial Recognition Model Setup***

* Select and set up a facial recognition library (e.g., OpenCV, Face Recognition API).
* Train the model using known faces from a cataloged dataset.

1. ***Video Processing***

* Develop a script to extract frames from video clips.
* Apply the facial recognition model to identify and tag faces.

1. ***Output Handling***

* Store identified faces with metadata (e.g., time in video, confidence score).
* Generate a report or visual representation of the results.

**Algorithms:** Use Deep Learning models like Convolutional Neural Networks (CNNs) tailored for facial recognition tasks.

**Libraries:** OpenCV for video processing, Dlib or Face Recognition for facial detection and recognition, and TensorFlow or Keras for deep learning implementations.

**Implementation Tips:**

* Preprocess video clips to adjust resolution and frame rate to optimize facial recognition accuracy.
* Implement face detection in video streams efficiently to handle large volumes of video data.
* Test the recognition system with different environmental conditions and angles to ensure robustness across various scenarios.

# Folder 5: Predictive Analysis (Browsing Habits)

## Challenge 6: Predictive Insights through Machine Learning

**Objective:** Predict the most frequently visited category in the suspect's browsing history.

**Tasks:**

1. ***Data Preprocessing***

* Collect and clean the browsing history data.
* Categorize each website visit into predefined categories (e.g., Shopping, Entertainment, social media).

1. ***Model Development***

* Select appropriate machine learning models for classification (e.g., Decision Trees, Naive Bayes, Neural Networks).
* Train the model using the categorized browsing data.

1. ***Model Validation***

* Use cross-validation to assess the model’s accuracy.
* Adjust parameters and techniques based on validation results.

1. ***Deployment and Reporting***

* Deploy the model to predict browsing categories for new data entries.
* Generate a report detailing predictions and insights into the suspect’s browsing habits.

**Algorithms:** For predicting browsing categories, you might consider algorithms like Logistic Regression, Random Forest, Gradient Boosting Machines (GBMs), or Neural Networks if the dataset is large and complex.

**Libraries:** Use Python libraries like Scikit-learn for model building, Pandas for data manipulation, and Matplotlib or Seaborn for data visualization.

**Implementation Tips:**

* Ensure the dataset is balanced for each category to avoid biases in the model.
* Feature engineering is crucial: extract features like the time of day for browsing and frequency of visits to each category.
* Regularly update the model as new browsing data becomes available to keep predictions accurate.

# Folder 6: Network Analysis (IP Address Classification)

## Challenge 7: Classification: Network Analysis

**Objective:** Predict the region associated with an input IP address for network analysis.

**Tasks:**

1. ***Dataset Preparation***

* Examine and preprocess the dataset containing IP address ranges and associated regions.
* Include additional data on CyberSec Solutions hosts’ vulnerabilities and their regions.

1. ***Model Training***

* Choose and train a classification model (e.g., Random Forest, Support Vector Machines).
* Ensure the model can accurately predict the region and identify vulnerabilities associated with IP addresses.

1. ***Model Evaluation***

* Evaluate the model’s accuracy using a test set.
* Adjust the model based on performance metrics like precision and recall.

1. ***Implementation***

* Integrate the model into the network analysis workflow.
* Use the model to support real-time decisions about network security.

**Algorithms:** Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) are good choices for geographical region prediction due to their effectiveness in handling high-dimensional space.

**Libraries:** TensorFlow or PyTorch can be used if you're implementing deep learning techniques, or Scikit-learn for more traditional machine learning approaches.

**Implementation Tips:**

* Use feature scaling to normalize IP address data which often improves the performance of models like SVM.
* Incorporate additional network features like latency or connection type if available, as they might improve predictive performance.
* Continuously train your model with new data to adapt to potential changes in IP address allocations or regional attributes.

# Folder 7: Data Security

## Challenge 8: Steganography Detection

**Objective:** Develop a specialized ML model to detect hidden messages or data within a digital image.

**Tasks:**

1. ***Research and Tool Selection***

* Investigate existing algorithms and tools capable of detecting steganography.
* Select appropriate methods (e.g., statistical analysis, pattern recognition algorithms).

1. ***Model Development***

* Train the model using a dataset of images known to contain steganographic content and clean images.
* Focus on detecting anomalies or patterns indicative of hidden data.

1. ***Testing and Optimization***

* Test the model with both known and unknown images to evaluate effectiveness.
* Optimize the detection algorithm to reduce false positives and false negatives.

1. ***Deployment***

* Implement the model as part of a larger digital forensics toolkit.
* Provide a mechanism for investigators to use the tool easily and obtain results quickly.

**Algorithms:** For detecting steganography, consider using statistical methods or machine learning techniques such as Convolutional Neural Networks (CNNs) that can detect subtle changes in image patterns.

**Libraries:** TensorFlow or Keras for building and training CNN models. For initial experiments, you might use the StegExpose tool for quick steganalysis based on statistical tests.

**Implementation Tips:**

* Training data should include a diverse set of images with and without steganography to teach the model to recognize various embedding techniques.
* Consider implementing image preprocessing steps to enhance model accuracy, like adjusting contrast or using edge detection filters.
* Deploy the model into a user-friendly application that allows investigators to upload images and receive immediate feedback on potential steganography.

**Implementation Plan**

# Challenge 1: NLP for Cyber-Attack Investigation