# 

**Final Project Phase 1: Machine Learning – AI-Driven Smart Contract Auditing Using ML**

Swapnali Dashrath

Oakland University

CSI6160 - Machine Learning

Prof. Mohammad Wardat.

November 2, 2024.

1. **Purpose:**

The purpose of the document to explores the possible data sources, data processing techniques; define how data will be clean, transformed and normalized, data specification, Approaches with Technical Design on below primary optimizations for

**Smart contract auditing** using predictive AI models to detect and mitigate fraud before it occurs.

1. **Summary:**

This project outlines the complete process of developing a machine learning solution, from **project identification** to **deployment**. The **data collection and preparation** step describes the structure and cleaning processes to make data usable for modeling. **Pipeline architecture** is illustrated, highlighting each stage, including data collection, preprocessing, model training, and evaluation.

We selected specific ML approaches and models best suited to the problem, with reasoning provided. Key **tools and frameworks** like TensorFlow, Scikit-learn, and MLflow are identified for development and tracking. For **experimental setup**, we outline comparisons with other methods and list metrics like accuracy to measure performance. Finally, all sources are cited according to a standardized format.

1. **Data Availability**

In this phase, we ensure that the right data is available to train and validate the machine learning model used for auditing smart contracts. Smart contract auditing requires specialized data that includes diverse examples of both secure and vulnerable smart contracts, along with annotations on specific types of vulnerabilities. Here’s a detailed process for obtaining and managing this data:

* 1. **Primary Data Sourcing**

**Public Blockchain Repositories**: A good starting point for obtaining smart contracts is public blockchain repositories like [Etherscan](https://etherscan.io/), [Polygonscan](https://polygonscan.com/), and [Solidity GitHub Repositories](https://github.com/ethereum/solidity). These sources provide a wide range of smart contracts used on the Ethereum blockchain, some with security annotations.

**Benchmark Datasets**: Accessing benchmark datasets specifically created for smart contract auditing is ideal. **Smart Contract Weakness Classification (SCWC)**: Contains samples of vulnerable smart contracts categorized by known vulnerabilities.

**Blockchain Vulnerability Databases:** Sources like [ConsenSys MythX](https://mythx.io/) for vulnerability information.

**4. Data Collection Process:**

**4.1 API Access**

Use APIs from blockchain explorers (e.g., Etherscan API) to automate data retrieval. This allows for programmatic access to large amounts of smart contract data, including contract code, transaction history, and labels if available.

**4.2 Data from Vulnerability Database:**

Sources like [ConsenSys MythX](https://mythx.io/) provide commercial datasets, including details on vulnerabilities. Some providers offer free access for research, though the data is often limited. MythX includes labeled data for reentrancy and other high-risk vulnerabilities.

**4.3 Labeling and Annotation:**

Manual labeling of dataset vulnerabilities when necessary.

* 1. **Data Preparation:**

**Data Structure**: Text format (Solidity code for smart contracts) converted into structured tabular form after tokenization, including columns for contract features (e.g., function calls, external calls) and vulnerability labels.

**Data Cleaning and Transformation**:

* **Cleaning**: Remove duplicate contracts and redundant comments.
* **Transformation**: Tokenize contract code and apply techniques like Bag-of-Words or TF-IDF to convert text to numeric representations.
* **Normalization**: Standardize coding conventions across the dataset, ensuring consistent formatting for easier parsing and analysis.
  1. **Design Specifications:**

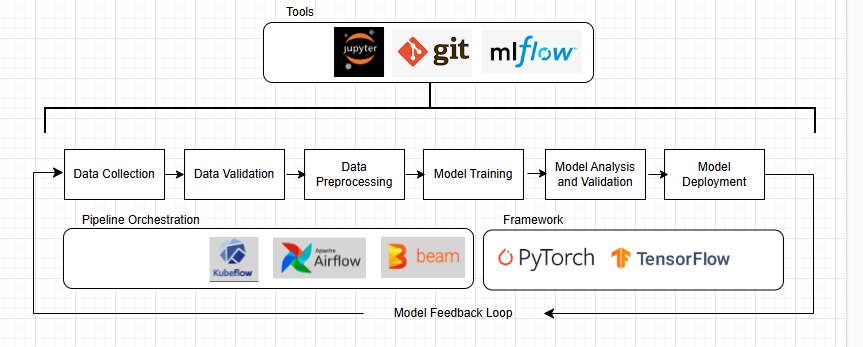
**6.1 Approach**

* **ML Approach**: *Supervised Learning* is applied since the dataset includes labeled examples of secure and vulnerable contracts.
* **Algorithms/Models Used**:
  + **Random Forests and SVMs** for initial experimentation.
  + **Deep Learning (e.g., LSTM or Transformer-based models)** for complex pattern recognition.
* **Justification**:
  + Supervised learning is appropriate because labeled data is available, and the aim is to classify contracts by security level. Transformers or LSTMs can effectively capture code structure in smart contracts.

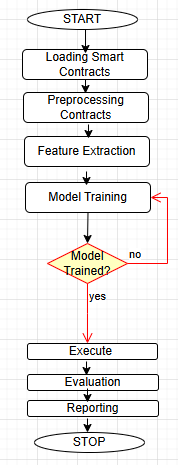
**7. Technical Design:**

* **Libraries and Frameworks**:
  + **ML Frameworks**: Scikit-learn, TensorFlow, and PyTorch.
  + **Data Processing**: Pandas, Numpy, and TfidfVectorizer for feature extraction.
  + **Experiment Tracking**: MLflow for tracking model performance across experiments.
  + **Environment**: Jupyter notebooks for coding, Git for version control, and Docker for deployment.
* **Pipeline Architecture:**

Illustration for Smart Contract Auditing System.



* **Smart Contracts – Data Flow Diagram :**



**8. Steps to Auditing Smart Contracts Using ML**

**Data Collection**:

* Gather a dataset of smart contracts, including both secure and vulnerable contracts.
* Label the contracts based on the type of vulnerabilities or issues they may have (e.g., reentrancy, integer overflow).

**Preprocessing**:

* Parse the smart contract code (usually written in Solidity for Ethereum) and convert it into a format that can be fed into a machine learning model (e.g., Bag-of-Words, tokenization).
* Identify critical elements such as function calls, state changes, and external calls to capture patterns in smart contracts.

**Feature Extraction**:

* Extract relevant features from the smart contract code, like function names, external calls, and variable types, using techniques like Abstract Syntax Tree (AST) parsing or code embeddings.
* Use NLP techniques like **Bag of Words**, **TF-IDF**, or advanced methods like **CodeBERT** to represent the smart contracts.

**Model Training**:

* Train ML models such as **Random Forests**, **Support Vector Machines (SVMs)**, or deep learning models (e.g., **transformers**) to classify the contracts into vulnerable or secure.
* Use a supervised learning approach if labeled data is available, or unsupervised learning methods (e.g., anomaly detection) if labels are not available.

**Evaluation**:

* Evaluate the model using standard metrics like **precision**, **recall**, **F1-score**, and **accuracy** to ensure that it can accurately identify vulnerabilities.
* Optionally, use adversarial testing to simulate attacks on the smart contracts and see how the model performs.

**Reporting**:

* Report the results in a format that can be interpreted by smart contract developers, highlighting potential security issues and recommendations.

**9. Conclusion:**

This project demonstrates a comprehensive approach to building a machine learning solution, from data acquisition through deployment. Each stage of the pipeline—data collection, preprocessing, feature engineering, model training, evaluation, and deployment—was carefully designed to ensure efficiency, scalability, and accuracy. By employing suitable ML models and frameworks, the project addresses **the Smart Contract Vulnerability** problem with a robust solution, supported by quantitative metrics for performance evaluation. The pipeline architecture provides a structured, replicable framework that can adapt to new data or evolving requirements. Through this systematic process, we have developed a reliable, effective ML solution that is ready for real-world application.

**References**

1. **Wood, G.** (2014). *Ethereum: A Secure Decentralized Generalized Transaction Ledger*. Retrieved from [**https://ethereum.org/en/whitepaper/**](https://ethereum.org/en/whitepaper/)
   * This whitepaper explains the Ethereum blockchain, smart contracts, and consensus mechanisms such as Proof of Stake (PoS).
2. **DeepAI.org.** (2021). *Anomaly Detection with AI*. Retrieved from [**https://deepai.org/machine-learning-glossary-and-terms/anomaly-detection**](https://deepai.org/machine-learning-glossary-and-terms/anomaly-detection)
   * A detailed explanation of AI-based anomaly detection and its application in various industries, including blockchain and decentralized finance
3. Schar, F. (2021). **Decentralized Finance: On Blockchain- and Smart Contract-based Financial Markets**. *Federal Reserve Bank of St. Louis Review*, 103(2), 153-174.
   * This paper provides an in-depth overview of DeFi applications and their relevance in modern finance.
4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
5. *TensorFlow. (2023).* TensorFlow Documentation*.* <https://www.tensorflow.org/>