Track Name: CREST SECURITY AI/ML
Track Title: Securing Systems Using AI/ML- Malware Detection

1. **Understanding the Dataset and Exploratory Data Analysis (EDA) :**
1. The dataset used for this project is a malware detection dataset obtained from a hackathon. It comprises:
    - **22,017 rows** in the training dataset and **1,480 rows** in the test dataset.
    - **28,696 columns** in the training dataset and **22,690 columns** in the test dataset.
    - The dataset includes features such as:
        - **SHA256**: A unique identifier for each sample.
        - **Type**: The classification label indicating whether the file is malware or benign.
        - **API Functions**: System-level API calls made by the executable.
        - **DLLs**: Dynamic-link libraries referenced by the executable.
        - **Portable Executable Features**: Metadata about the structure of the executable.
2. **Exploratory Data Analysis (EDA) :**
2. To understand the structure and distribution of the dataset, the following analyses were conducted:
    1. **Missing Values Analysis**:
        - Checked for missing values in the dataset and found that some API functions and DLLs had null values.
        - Imputed missing values using appropriate techniques such as median imputation for numerical data.
    2. **Feature Distribution**:
        - Analyzed the frequency of API function calls and DLL imports across different types of executables.
        - Identified redundant or highly correlated features.
    3. **Class Distribution**:
        - Checked for class imbalance in the dataset (malware vs. benign files).
        - If imbalance existed, considered oversampling, undersampling, or using balanced class weights in the model.
    4. **Dimensionality Analysis**:
        - Given the high-dimensional nature of the dataset, feature selection and dimensionality reduction were necessary to improve computational efficiency.

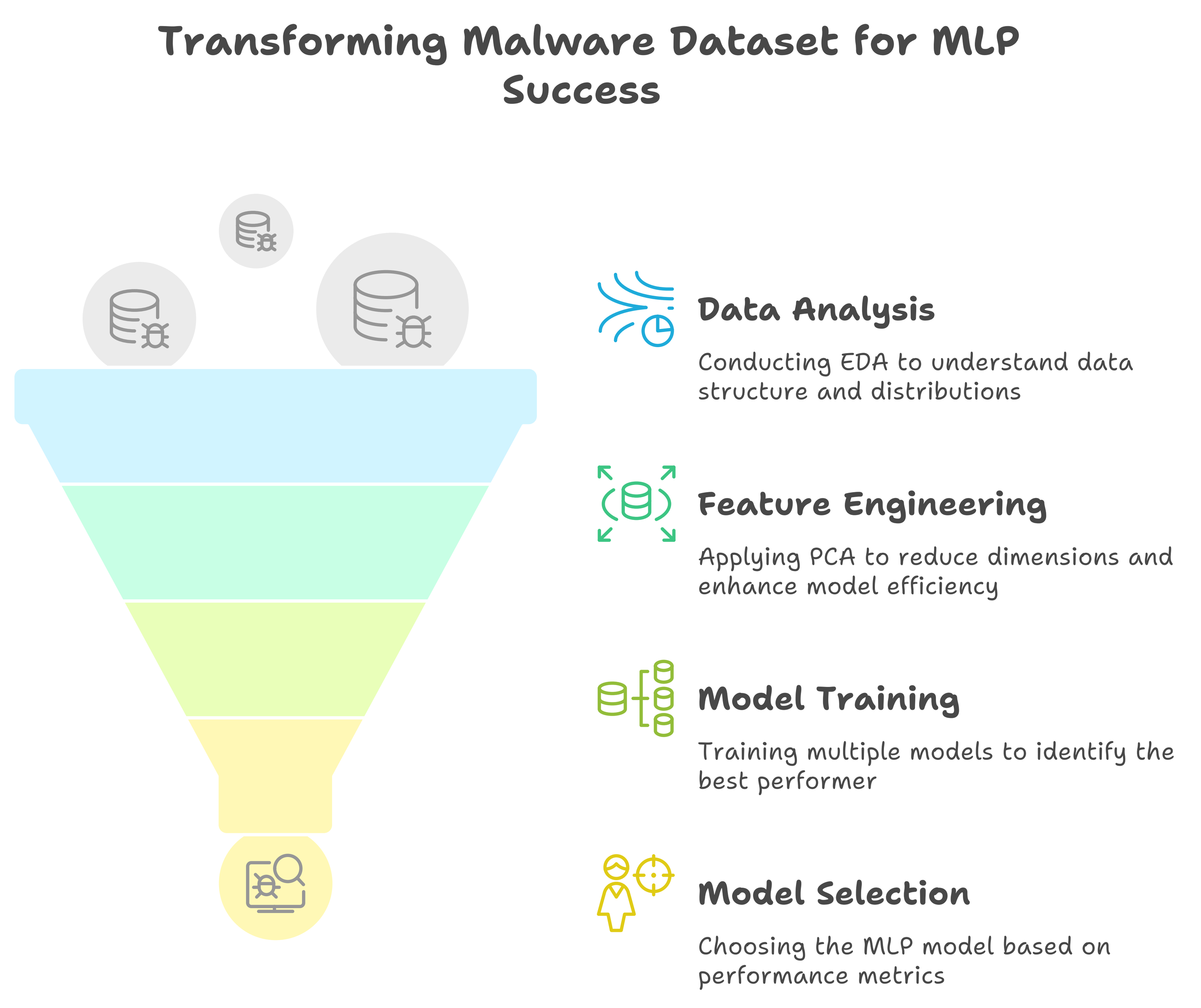
3. **Feature Engineering: Principal Component Analysis (PCA) :**
3. Principal Component Analysis (PCA) was applied to reduce the high-dimensional dataset to a lower-dimensional representation.
    - **Steps Taken for PCA:**
        1. Removed non-numeric features such as SHA256 and Type.
        2. Standardized the data to ensure all features had equal weight.
        3. Applied PCA to reduce the feature set to **50 principal components**.
        4. Selected the number of components based on variance explained (choosing components that retain the majority of the dataset's variance).
    - **Benefits of PCA in This Context:**
        - **Dimensionality Reduction**: The dataset was extremely high-dimensional (28,696 columns). PCA helped reduce it to a manageable size.
        - **Improved Model Efficiency**: A lower-dimensional dataset requires less computation, speeding up training.
        - **Noise Reduction**: PCA eliminates redundant and noisy features, helping models generalize better.
        - **Feature Correlation Handling**: Many API calls and DLLs were correlated. PCA helped transform the dataset into uncorrelated principal components.
4. **Why PCA Was Chosen for This Dataset**
4. Given the structure of the dataset, PCA was chosen for dimensionality reduction due to the following reasons:
    - **Curse of Dimensionality**: With nearly 29,000 features, training a model directly on raw data would have been computationally expensive and might lead to overfitting.
    - **Efficient Feature Representation**: PCA enables capturing most of the variance in significantly fewer dimensions.
    - **Interpretability**: While PCA reduces interpretability at the feature level, it allows for better visualization and understanding of patterns in the data.
    - **Alternative Approaches Considered**:
        - Feature selection techniques such as mutual information and variance thresholding were tested but did not reduce dimensions significantly.
        - Autoencoders (deep learning-based dimensionality reduction) were considered but required longer training times compared to PCA.
4. By applying PCA, we transformed the malware detection dataset into a more manageable format while preserving the most critical information for classification tasks.
5. **Our Approach to Solving the Problem Statement :**
5. To effectively classify malware, we first merged the three datasets into a single comprehensive dataset. PCA was then applied to this merged dataset, producing two versions:
    - **100 principal components**: Used for traditional ML models and the ConvLSTM model.
    - **50 principal components**: Used for the Quantum XGBoost model.
5. Following this transformation, we trained multiple models to determine the best-performing one. The models trained include:
    - **Logistic Regression**: A simple yet effective model for binary classification.
    - **Random Forest**: An ensemble method to capture feature importance and interactions.
    - **XGBoost**: A gradient-boosted decision tree model known for its high accuracy.
    - **Support Vector Machine (SVM)**: Effective for high-dimensional spaces.
    - **Gradient Boosting Classifier**: A boosting model to refine predictions iteratively.
    - **MLP Classifier (Neural Network)**: A multi-layer perceptron to capture complex relationships.
    - **Quantum XGBoost**: Leveraging quantum computing techniques to enhance performance.
    - **ConvLSTM Model**: A deep learning model combining convolutional and LSTM layers for sequential pattern recognition.
5. Each model was evaluated based on accuracy, precision, recall, and F1-score. The goal was to balance computational efficiency with predictive performance.
5. After identifying the best-performing model, we proceeded to develop a **pipeline** that transforms incoming data into its PCA representation before passing it into the model for prediction. This ensured a smooth deployment process and consistent feature representation across different datasets.

6. **Why We Trained Multiple Models**
    1. **Performance Benchmarking**: Testing various models provided insights into which approach worked best.
    2. **Different Learning Paradigms**: Classical machine learning, boosting techniques, and deep learning models were explored to compare strengths.
    3. **Robustness Check**: Ensuring the chosen model generalizes well across different malware types.
    4. **Exploring Quantum Machine Learning**: Quantum XGBoost was included to assess the impact of quantum computing in malware detection.
    5. **Handling Complex Data Patterns**: The dataset included highly correlated features; deep learning models like ConvLSTM helped capture hidden dependencies.
    6. **Pipeline Integration**: The best-performing models were evaluated for seamless integration into a real-world detection pipeline.

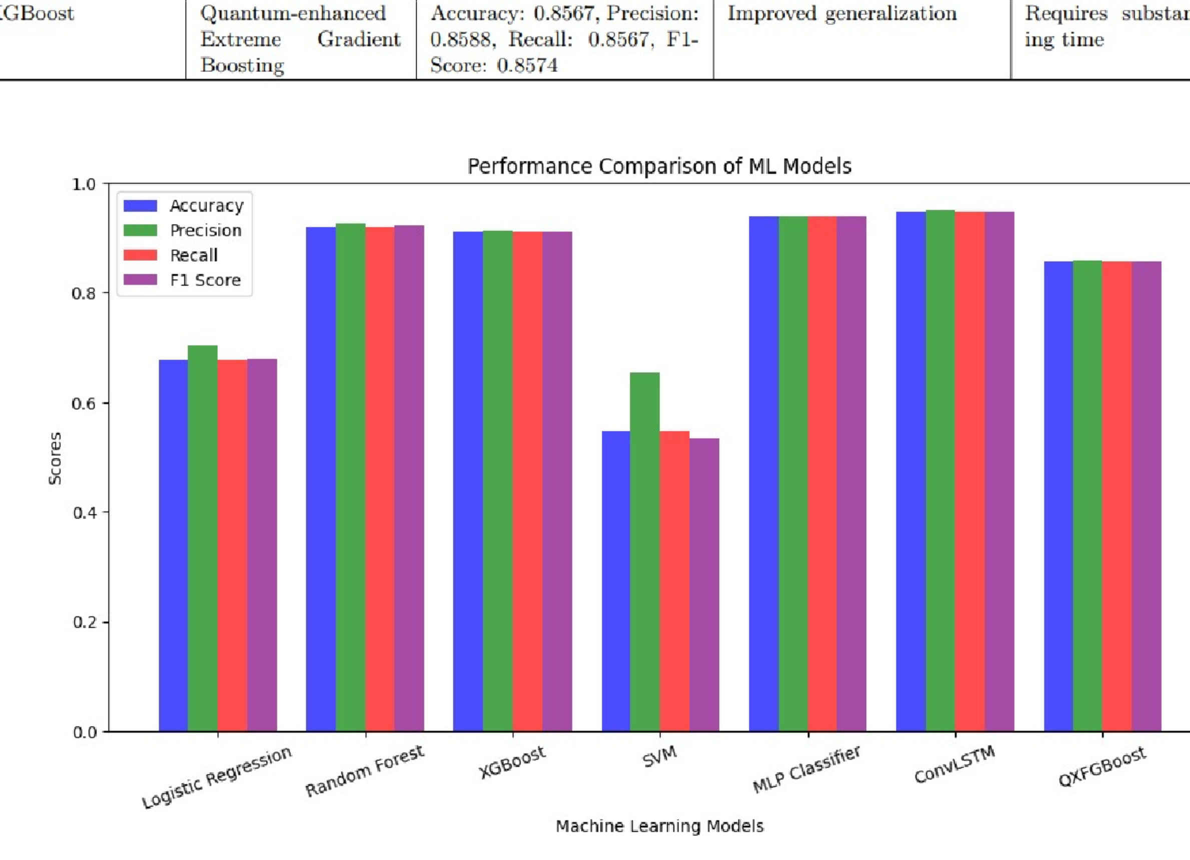
7. **Why We Chose the MLP Model as the Best One**
7. After extensive evaluation, the **Multi-Layer Perceptron (MLP) classifier** was selected as the best-performing model. The primary reasons for this choice were:
    - **Superior Accuracy and F1-Score**: MLP outperformed other models in both accuracy and F1-score, indicating balanced performance across malware and benign classifications.
    - **Ability to Capture Nonlinear Relationships**: Unlike linear models such as Logistic Regression, MLP effectively captured complex patterns in the dataset.
    - **Robustness to Feature Correlations**: Due to its multi-layer architecture, MLP handled feature interactions well, leveraging PCA-transformed data more effectively.
    - **Generalization**: The model demonstrated strong generalization on unseen test data, making it suitable for real-world deployment.

MLP Model Architecture:

7. The architecture of the selected **MLP Classifier** is as follows:
    - **Hidden Layers**: Two hidden layers with **50 and 25 neurons**, respectively.
    - **Activation Function**: Rectified Linear Unit (ReLU) for non-linearity.
    - **Optimization**: Adam optimizer for efficient gradient updates.
    - **Regularization**: L2 regularization (alpha=0.05) to prevent overfitting.
    - **Epochs**: Trained for **400 iterations** to ensure convergence.
    - **Random State**: Fixed for reproducibility.
7. This architecture allowed the model to balance complexity and efficiency, making it the best choice for malware detection in our study.



Model	Techniques	Results	Advantages	Limitations
ConvLSTM	Convolutional Neural Network + Long Short-Term Memory	Accuracy: 0.9491, Precision: 0.9506, Recall: 0.9491, F1-Score: 0.9491	Handles temporal dependencies well	Computationally expensive
Logistic Regression	Statistical Linear Model for Binary Classification	Accuracy: 0.6772, Precision: 0.7029, Recall: 0.6772, F1-Score: 0.6793	Simple and interpretable	Limited to linear relationships
Random Forest	Ensemble of Decision Trees with Bootstrapping	Accuracy: 0.9209, Precision: 0.9274, Recall: 0.9209, F1-Score: 0.9212	High accuracy, robust to overfitting	Slower on large datasets
XGBoost	Extreme Gradient Boosting with Decision Trees	Accuracy: 0.9118, Precision: 0.9132, Recall: 0.9118, F1-Score: 0.9117	Efficient and powerful	Requires careful hyperparameter tuning
SVM	Kernel-based Classification with Maximum Margin	Accuracy: 0.5476, Precision: 0.6546, Recall: 0.5476, F1-Score: 0.5350	Works well with small datasets	Struggles with large datasets
MLP Classifier	Multi-Layer Perceptron Classifier	Accuracy: 0.9390, Precision: 0.9398, Recall: 0.9390, F1-Score: 0.9388	High accuracy	Can be computationally expensive
QXGBoost	Quantum-enhanced Extreme Gradient Boosting	Accuracy: 0.8567, Precision: 0.8588, Recall: 0.8567, F1-Score: 0.8574	Improved generalization	Requires substantial training time



Conclusion:

Through extensive experimentation and evaluation, we successfully developed an efficient malware detection pipeline using a PCA-transformed dataset. The study explored multiple machine learning and deep learning approaches, ultimately selecting the **MLP Classifier** as the best-performing model due to its superior accuracy, robustness, and ability to capture complex feature interactions.

Key takeaways from this project include:

- **PCA proved to be an effective method** for dimensionality reduction, maintaining the dataset's essential characteristics while improving computational efficiency.
- **Comparing multiple models** allowed us to benchmark various approaches and select the most optimal one.
- **MLP's architecture provided the best balance** between complexity and performance, making it the preferred choice for malware classification.
- **A robust pipeline was developed**, ensuring seamless transformation of raw data into its PCA representation before classification.

Future work may involve fine-tuning deep learning architectures, exploring hybrid quantum-classical approaches, and optimizing real-time malware detection strategies for deployment in cybersecurity environments.