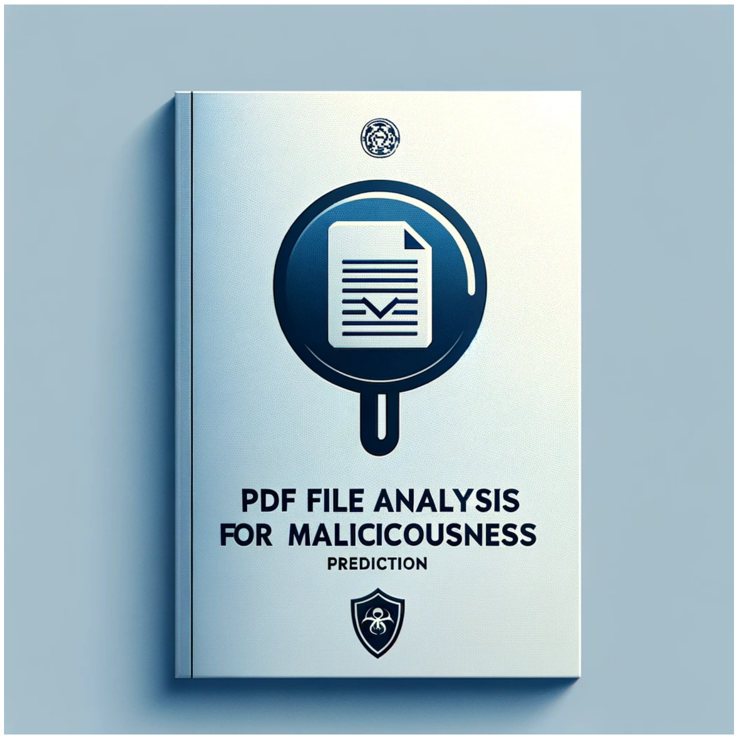
CYBERSECURITY

PDF File Analysis for Maliciousness Prediction

-Amal Raj Singh, Manasvi Logani



Introduction

Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks. These cyber attacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes. In today's digital age, where the reliance on technology and digital data has grown exponentially, cybersecurity has become a critical shield against data breaches, identity theft, and other malicious digital activities. It is fundamental in safeguarding data privacy, maintaining the integrity and functionality of our digital infrastructure, and ensuring the digital economy's resilience against cyber threats.

In cybersecurity, files can be broadly categorized as '**malicious**' or '**benign**'.

Malicious files are designed to cause harm to a computer, server, client, or network. They can include various forms of malware like viruses, trojans, and spyware, each capable of inflicting different types of damage, ranging from stealing sensitive data to hijacking core computing functions.

Benign files, on the other hand, are harmless and do not contain any code or content that could potentially damage or disrupt the operation of a system. Distinguishing between these two types of files accurately and promptly is a significant challenge in cybersecurity.

Problem Statement

This project focuses on the challenge of **predicting the maliciousness of PDF files** using a machine learning (ML) model. Despite being a common file format used for document sharing, PDFs can be weaponized to carry out cyber attacks. The project aims to **develop a model that can analyze PDF files and accurately identify whether they are benign or malicious**. This requires a nuanced understanding of the file's characteristics and behaviors, extracted as features, that differentiate harmful files from safe ones.

The utility of this project lies in its ability to enhance cybersecurity measures by providing a reliable tool for preemptive threat detection. By employing machine learning techniques for automated, intelligent analysis of PDF files, the model aims to achieve high accuracy with a low false positive rate. Our approach utilizes supervised learning with a labeled dataset, consisting of thousands of known benign and malicious PDF files, to train and test the model. The effectiveness of the model will be evaluated using cross-validation, ensuring its robustness and reliability in real-world scenarios.

Scope of the Project

1. **Development of a Machine Learning Model**

   The core of this project is the development of a machine learning (ML) model that can predict the maliciousness of PDF files. This involves creating a system that can analyze the intrinsic properties of a PDF file and determine if it poses a security threat. The model is designed to be robust, aiming for a high detection rate of malicious files while maintaining a low false positive rate.

2. **Feature Extraction and Utilization**

   An integral part of this project is the process of feature extraction from PDF files. Using open-source tools, we extract a variety of relevant features that are crucial in differentiating between benign and malicious files. These features form the basis of the training and operational use of the ML model. Feature extraction is a critical step both during the training phase and the model's application in real-world scenarios.

3. **Supervised Learning Approach**

   Our methodology revolves around a supervised learning approach using a labeled dataset. The dataset consists of thousands of known benign and malicious PDF files, categorized accordingly into separate folders. This dataset is pivotal for training the ML model, enabling it to learn the distinguishing characteristics of both benign and malicious files.

4. **Training and Testing of the Model**

   The model will be trained on 70% of the provided dataset, with the remaining 30% reserved for testing its accuracy. This division is designed to ensure a comprehensive learning process while leaving a substantial portion of the dataset for validation purposes.

5. **Cross Validation for Accuracy Reporting**

   To ensure the reliability and robustness of the model, cross-validation will be employed. This involves averaging the accuracy obtained from multiple iterations of training and testing the model with different splits of the dataset. Cross-validation provides a more accurate measure of the model's performance and generalizability.

6. **Application to Real-world Scenarios**

   While the primary focus is on building and testing the model with the provided dataset, the ultimate goal is to apply this model to classify new, unseen PDF files. However, for the scope of this project, testing with new external files is not included. The model's efficacy will be evaluated solely based on its performance with the provided dataset.

7. **Limitations**

   It is important to acknowledge that the project's scope is confined to the dataset provided and the features extracted using the specified open-source tools. The model's effectiveness in real-world applications may vary depending on factors such as variations in malicious file techniques and the evolution of PDF file structures over time.

Brief Description of the Approach

The process begins with Dataset Preparation. Here, features are extracted from PDFs using a provided tool, and then compiled into a single dataset from JSON files. Each feature set is labeled to indicate if the corresponding PDF is benign (0 or False) or malicious (1 or True).

During **Feature Engineering**, the dataset is refined, handling null values, categorical variables, and strings to ensure suitability for model training.

Data Loading and Preprocessing involves loading these features into a structured format and splitting the data for training and testing.

Exploratory Data Analysis (EDA) includes conducting correlation analysis and visualizing feature distributions to uncover patterns. Subsequently, Feature Engineering addresses data scaling and missing values.

Model Training utilizes RandomForest and XGBoost classifiers from scikit-learn, with k-fold cross-validation for robust evaluation.

Finally, **Model Evaluation** focuses on analyzing model performance through confusion matrices, accuracy scores, ROC curves with specific threshold values, hyperparameter tuning, and assessing feature importance to determine the most predictive factors. This comprehensive approach aims to build a highly accurate model with low false positives, contributing significantly to cybersecurity efforts.

Dataset and its Attributes

Overall Composition: The dataset consists of 9,748 records, each representing a unique PDF file, and is spread across 71 columns, encompassing a wide range of features.

Data Types and Attributes:

The dataset features a diverse array of data types. The majority of the columns are of the 'object' type, indicating textual or mixed-type data, with several columns represented as integer types (int64).

Notable columns include *'exif_signatures'* and *'pdfid_signatures'*, which are likely indicative of intrinsic file characteristics. Additionally, columns like *'pypdf_uris'* and *'regex_uris'* contain URL data extracted from the PDF files.

A significant part of the dataset is dedicated to various *'yara_signatures'* columns. These fields likely correspond to specific patterns or indicators related to malware detection in the files.

Labeling Scheme:

The dataset includes a crucial *'malicious/benign'* label for each file, serving as the ground truth for our classification task.

Data Structure:

Each row in the dataset corresponds to an individual PDF file, with the columns providing a multitude of extracted features. These features range from basic file properties to advanced indicators obtained through YARA signature analysis.

Categorical and Numerical Data:

The dataset is a blend of numerical data (such as file sizes and various count-based features) and categorical or textual data (such as URLs and signature names). This mix necessitates meticulous preprocessing to ensure model compatibility and performance.

Data Integrity:

An initial review of the dataset indicates mixed data types in certain columns, which may require special handling during the data preprocessing phase.

The complexity and diversity of the data required careful preprocessing, including handling of mixed data types and potential missing values, which was essential to optimize the model's predictive capabilities.

Code Details: Functionality and Description

The code file is divided into code cell clusters with certain codes grouped for a more targeted analysis. (For visualizations, only implementation is given in this section, inference will be given in the next section). The cell cluster-wise analysis is given below:

Detailed Report for Code Cell Number 1

Functionality and Description:

This code cell is focused on importing various Python libraries and modules that are essential for different aspects of the project. Each import serves a specific purpose:

- **Data Handling and Analysis**:
  - **pandas**: Provides data structures and data analysis tools. Ideal for handling and manipulating tabular data.
  - **numpy**: Offers support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- **Visualization**:
  - **matplotlib.pyplot**: Used for creating static, interactive, and animated visualizations in Python.
  - **plotly.graph_objects** and **plotly.express**: Plotly's graphing libraries provide interactive and aesthetically pleasing visualizations.
  - **seaborn**: A Python data visualization library based on matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics.
- **Domain Extraction**:
  - **tldextract**: Useful for extracting parts from a domain name, separating the top-level domain (TLD), domain, and subdomains.
- **Machine Learning Modeling and Evaluation**:
  - **xgboost**: Stands for eXtreme Gradient Boosting, a library for efficient and scalable implementation of gradient boosting.
  - **sklearn.metrics**: Contains functions for calculating key metrics for evaluating machine learning models, including ROC curve, AUC, accuracy, precision, recall, and confusion matrix.

    - **sklearn.ensemble.RandomForestClassifier**: A machine learning algorithm for classification tasks, part of the ensemble methods in scikit-learn.
    - **RandomizedSearchCV**: A cross-validation method for hyperparameter tuning, optimizing model parameters.
    - **train_test_split**: Splits arrays or matrices into random train and test subsets.
    - **cross_val_score**: Evaluate a score by cross-validation.
- **Statistical Operations**:
    - **scipy.stats.randint**: Used for generating random integers, helpful in random sampling and hyperparameter tuning.

Use in the Project:

- **Data Preparation**: **pandas** and **numpy** are fundamental for loading and preparing the dataset for analysis and modeling.
- **Visualization**: **matplotlib**, **plotly**, and **seaborn** are used to visualize data and results, which is crucial for exploratory data analysis and presenting findings.
- **Feature Extraction**: **tldextract** indicates that domain name analysis is a part of the feature engineering process.
- **Model Development and Evaluation**: **xgboost** and **RandomForestClassifier** suggest the use of these algorithms for the main classification task. The various metrics from **sklearn.metrics** and tools like **RandomizedSearchCV** and **cross_val_score** point towards model evaluation and optimization.

Key Libraries/Methods:

- **Random Forest and XGBoost**: These are the primary machine learning models used for classification in this project.
- **Visualization Tools**: The use of multiple visualization libraries suggests a strong emphasis on data visualization for both analysis and reporting.
- **Domain Extraction**: The inclusion of **tldextract** implies that domain-related features are significant in this classification task.

This code cell sets the stage for the entire project by importing necessary libraries and modules, indicating a comprehensive approach involving data manipulation, visualization, feature engineering, and machine learning modeling.

Detailed Report for Code Cell Number 2

Functionality and Description:

This cell is primarily focused on data loading, initial exploration, preprocessing, and segmentation based on file type (benign or malicious). The key actions performed in this cell include:

- **Data Loading:**
  - **pd.read_csv():** Reads a CSV file into a DataFrame. Here, it's used to load the combined dataset from a specified path.
- **Data Exploration:**
  - **df.head():** Displays the first few rows of the DataFrame, providing an initial look at the data structure and contents.
- **Data Cleaning:**
  - **df.drop():** Removes unwanted columns. In this case, it's used to drop the 'Unnamed: 0' column, which often results from an index column in CSV files.
- **Data Shape:**
  - **df.shape:** Provides the dimensions of the DataFrame, indicating the number of rows and columns.
- **Dataset Segmentation:**
  - The dataset is split into two separate DataFrames based on the 'malicious/benign' column, creating one DataFrame for benign files (df_benign) and another for malicious files (df_malicious).
- **Resetting Index and Cleaning:**
  - reset_index() and drop(): These are used to reset the index of the newly created DataFrames and remove the old index column.
- **Data Counting:**
  - len(): Used to print the number of entries in each of the segmented DataFrames (df_benign and df_malicious).

Use in the Project:

- **Initial Data Handling:** This cell is crucial for the initial handling of the dataset. It involves loading the data, getting a preliminary view, and cleaning unnecessary columns.
- **Data Segmentation for Analysis:** By splitting the data into benign and malicious subsets, this cell sets up the foundation for separate analyses and model training for different file types.
- **Preparation for Feature Analysis:** The segmentation hints at an upcoming detailed analysis or feature extraction specific to each file type, which is pivotal for understanding the characteristics of benign and malicious files.

Key Aspects:

- **Data Cleaning and Preparation:** The removal of unnecessary columns and resetting of indices are essential data preparation steps to ensure clean and well-structured data for analysis.
- **Segmentation for Targeted Analysis:** Splitting the dataset into benign and malicious categories is a strategic move for targeted analysis, which is often necessary in classification tasks where the characteristics of each class may vary significantly.

This code cell effectively prepares the dataset for the subsequent steps in the project, ensuring that the data is clean, well-organized, and ready for detailed exploration and modeling.

Detailed Report for Code Cell Number 3

Functionality and Description:

This code cell is dedicated to extracting, processing, and analyzing domain extensions from URLs found in benign files. The key steps include:

- **Dictionary Initialization:**
    - **benign_dict:** Initializes an empty dictionary to store domain extensions and their occurrence counts in benign files

      .
- **Data Extraction and Processing:**
    - The for loop iterates over the pypdf_uris column of df_benign, which presumably contains URLs in each row.
    - URL strings are cleaned (removing square brackets and single quotes) and split into individual URLs.
    - **tldextract.extract():** Extracts the domain extension (suffix) from each URL.
- **Counting Domain Extensions:**
    - The loop counts the occurrences of each domain extension and stores this in benign_dict.
- **Conversion to DataFrame:**
    - **benign_domain_df:** Converts the dictionary into a DataFrame for easier manipulation and analysis.
- **Data Sorting and Display:**
    - The DataFrame is sorted by the 'Count' column in descending order.
    - print(benign_domain_df.head()): Displays the top entries of the DataFrame.
- **Visualization:**
    - The bar plot visualizes the top 20 domain extensions found in benign PDF files. It uses matplotlib.pyplot for plotting, with domain extensions on the x-axis and their counts on the y-axis.

Use in the Project:

- **Feature Analysis:** This cell is focused on analyzing the domain extensions as a feature. By identifying the most common domain extensions in benign files, it helps in understanding patterns or norms for safe files.

- **Data Visualization:** The bar plot provides a clear visualization of the most frequent domain extensions in benign files, which can be critical for exploratory data analysis and presenting findings.

Key Aspects:

- **Domain Extension Analysis:** This is an example of feature engineering where domain extensions are extracted and analyzed to understand their distribution in benign files. Such analysis can provide insights into typical characteristics of benign files.
- **Data Preparation for Machine Learning:** Understanding and visualizing these features can be pivotal in deciding which features to use for machine learning modeling.
- **Visualization for Insight:** The bar plot is an effective way to visually communicate the findings of the analysis, highlighting the most common domain extensions in benign files.

This code cell demonstrates a detailed approach to feature analysis, specifically focusing on domain extensions in benign files, and presents the findings in both tabular and graphical formats for comprehensive understanding.

Detailed Report for Code Cell Number 4

Functionality and Description:

This cell focuses on creating a new feature based on the occurrence of suspicious domains in PDF files and analyzing their distribution in benign and malicious files. The key steps include:

- **Defining Safe Domain Extensions:**
    - **safe_domain_extensions:** A predefined list of domain extensions considered safe.
- **Feature Engineering - Suspicious Domain Count:**
    - The first for loop iterates through the pypdf_uris column of the DataFrame to count the number of suspicious domain extensions in each file.

  - It processes URL strings (removing brackets and quotes) and extracts domain extensions.
  - Any domain extension not in the safe_domain_extensions list is counted as suspicious.
- **Adding New Feature to DataFrame:**
  - **df['sus_domain_count']:** The count of suspicious domain extensions is added as a new column to the DataFrame.
- **Threshold Analysis for Suspicious Domains:**
  - The second set of nested loops performs an analysis to understand how the number of suspicious domains correlates with the file being malicious or benign.
  - It iterates over a set of threshold values (0 to 3) for suspicious domain counts.
  - For each threshold, it counts the number of malicious and benign files exceeding that threshold.

Use in the Project:

- **Feature Creation:** The addition of sus_domain_count is a critical step in feature engineering, which could significantly impact the performance of machine learning models.
- **Insightful Analysis:** The threshold analysis provides insights into how the presence of suspicious domains might correlate with a file's likelihood of being malicious, which is valuable for understanding the data characteristics and guiding the modeling strategy.

Key Aspects:

- **Innovative Feature Engineering:** The creation of a new feature based on domain analysis demonstrates an innovative approach to leveraging domain-related information in cybersecurity.
- **Data Preprocessing and Cleaning:** The cell includes URL data preprocessing, which is crucial for accurate feature extraction.

- **Analytical Approach:** The threshold-based analysis helps in understanding the potential impact of the new feature on classifying files as malicious or benign.

This code cell shows a thoughtful approach to enhancing the dataset with new features based on domain analysis and exploring the relationship between these features and file classifications, thereby laying the groundwork for more effective machine learning modeling.

Detailed Report for Code Cell Number 5

Functionality and Description:

This cell is focused on analyzing and visualizing the count of file attachments (like PDFs, DOCs) in benign and malicious files. The key steps include:

- **Counting File Attachments:**
    - Two separate for loops are used for df_benign and df_malicious DataFrames.
    - In each loop, it processes the pypdf_uris column similar to previous cells (removing brackets and quotes) and counts the number of file attachments (PDF, DOC, DOCX).
- **Adding New Feature to DataFrames:**
    - **file_attachments_count:** This new feature, representing the count of file attachments, is added to both benign and malicious DataFrames.
- **Statistical Analysis:**
    - **describe():** Generates descriptive statistics (mean, std, min, etc.) for the file_attachments_count in both benign and malicious DataFrames.
    - The 'count' statistic is removed to focus on other measures like mean, standard deviation, etc.
- **Visualization:**
    - Using Plotly's go.Figure and go.Bar, the cell creates a grouped bar chart that compares the descriptive statistics of file attachments in benign and malicious files.

Use in the Project:

- **Feature Analysis and Visualization:** This cell provides a comparative analysis of file attachments in benign and malicious files, which is a valuable feature in distinguishing between the two.
- **Insightful Visualization:** The grouped bar chart effectively visualizes the comparison, making it easy to understand differences in attachment counts between benign and malicious files.

Key Aspects:

- **Enhanced Data Understanding:** The addition of file_attachments_count and its subsequent analysis enhance the understanding of how file attachments might differ in benign and malicious files.
- **Data Visualization for Insight:** The use of Plotly for visualization indicates a focus on creating interactive and informative charts, which are valuable for exploratory data analysis and presenting findings.

This code cell demonstrates a comprehensive approach to feature analysis by not only creating a new feature but also analyzing it statistically and visualizing the results in a way that highlights key differences between benign and malicious files. Such analysis is crucial for understanding the data and guiding the machine learning modeling process.

Detailed Report for Code Cell Number 6

Functionality and Description:

This cell continues the analysis of file attachments in the dataset, focusing on consolidating this information and updating the main DataFrame. The steps include:

- **Creating a Comparative DataFrame:**
  - **df_file_atch:** A new DataFrame is created to hold the descriptive statistics of file attachments for both benign and malicious files.
- **Updating DataFrame with File Attachment Count:**
  - A loop processes the pypdf_uris column in the main DataFrame (df), counting the number of file attachments.

    - The count is added to df as a new column, file_attachments_count.
- **Data Export:**
    - **df.to_csv:** The updated DataFrame, now containing the new feature, is exported to a CSV file, which can be used for further analysis or modeling.

Use in the Project:

- **Data Consolidation:** The creation of df_file_atch consolidates key statistics in a single DataFrame, facilitating easier comparison and analysis.
- **Feature Addition to Main DataFrame:** The addition of file_attachments_count to df is crucial for incorporating this feature into subsequent machine learning models.
- **Data Export for Reusability:** Exporting the updated DataFrame to CSV allows for the preservation of the current state of the data, including the new features, which can be beneficial for future uses or stages of the project.

Key Aspects:

- **Enhanced Data Preparation:** Adding the file_attachments_count feature to the main DataFrame is a significant step in preparing the data for machine learning modeling.
- **Data Management:** The creation of a separate DataFrame for comparative statistics and the export of the updated main DataFrame show a systematic approach to data management and reusability.

This code cell demonstrates efficient data management and preparation practices, ensuring that the dataset is enhanced with relevant features and readily available for future analysis or modeling stages in the project. The systematic approach to updating and saving the dataset underlines the project's emphasis on thorough data preparation and feature engineering.

Detailed Report for Code Cell Number 7

Functionality and Description:

This code cell is focused on final data preparation steps for machine learning model training. The key actions in this cell include:

- **Feature Selection:**
    - **df_prep:** A new DataFrame is created by selecting specific columns from the original DataFrame df. These columns include various features that are likely relevant for the classification task (file size, URI counts, script counts, static properties, YARA signatures, and the newly added features like suspicious domain count and file attachments count).
    - There's a commented-out section indicating an alternative, possibly more concise selection of features, which suggests experimentation with different feature sets.
- **Handling Missing Values:**
    - **fillna(0):** Replaces any missing values (NaNs) in the DataFrame with 0. This is a common practice to handle missing data in machine learning.
- **Checking for Missing Values:**
    - **isna().sum():** Ensures that there are no remaining missing values in the DataFrame.
- **Splitting Features and Target Variable:**
    - **X:** Contains all the features except the target variable ('malicious/benign').
    - **y:** The target variable, indicating whether a file is malicious or benign.

Use in the Project:

- **Data Preparation for Modeling:** This step is crucial in preparing the dataset for feeding into a machine learning model. Selecting the right features and handling missing values are key to building an effective model.
- **Feature-Target Split:** Separating the dataset into features (X) and target (y) is a standard practice in supervised learning tasks and is necessary for training and evaluating models.

Key Aspects:

- **Thoughtful Feature Selection:** The selection of features indicates a comprehensive approach, considering various aspects of the files, including their content, structure, and properties identified by YARA signatures.
- **Data Cleaning:** Addressing missing values is vital for the robustness of machine learning models, as many algorithms do not handle NaN values by default.

- **Readiness for Machine Learning:** At this stage, the dataset is ready for machine learning model training and validation. The separation of features and target variable is a clear indicator of this transition.

This code cell demonstrates the final steps in data preparation, ensuring the dataset is clean, well-structured, and ready for the subsequent stages of machine learning modeling. The careful selection of features and handling of missing values are indicative of meticulous data handling practices in the project.

Detailed Report for Code Cell Number 8

Functionality and Description:

This extensive cell encompasses several key steps in training and evaluating an XGBoost model for the binary classification task. It includes model training, predictions, evaluation, feature importance analysis, ROC curve generation, and cross-validation. The steps are as follows:

- **XGBoost Data Preparation:**
    - **DMatrix:** Constructs DMatrix objects for training (dtrain) and testing (dtest) datasets, as well as the complete dataset (data_dmatrix).
- **Model Parameters and Training:**
    - **params:** Sets up the parameters for the XGBoost model, including max depth, learning rate, objective function, and evaluation metric.
    - **xgb.train:** Trains the model on the training dataset for a specified number of epochs (1500).
- **Making Predictions and Evaluating Accuracy:**
    - **bst.predict:** Generates predictions for the test set.
    - **accuracy_score:** Calculates the accuracy of the model.
- **Confusion Matrix Visualization:**
    - A confusion matrix is plotted using seaborn's heatmap to visualize the model's performance in terms of true positives, true negatives, false positives, and false negatives.

- **Classification Report:**
    - classification_report: Provides a detailed classification report including precision, recall, f1-score, and support.
- **Feature Importance Analysis:**
    - Extracts and plots the feature importance using XGBoost's built-in functions.
- **ROC Curve and AUC:**
    - roc_curve and auc: Calculates and plots the ROC curve and computes the Area Under Curve (AUC) score.
- **Precision-Recall Curve:**
    - A Precision-Recall curve is plotted using Plotly, providing an interactive visualization.
- **Cross-Validation:**
    - xgb.cv: Performs cross-validation on the complete dataset to evaluate the model's performance using the AUC metric.

Use in the Project:

- **Model Training and Evaluation:** This cell is central to the project, covering the training, prediction, and evaluation of the XGBoost model.
- **Model Performance Visualization:** Provides visual insights into the model's performance through confusion matrix, ROC curve, and Precision-Recall curve.
- **Feature Importance for Interpretability:** Helps in understanding which features are most influential in the model's predictions.
- **Model Validation:** Cross-validation is used to assess the model's robustness and generalization ability.

Key Aspects:

- **Comprehensive Model Evaluation:** The combination of accuracy, confusion matrix, classification report, ROC curve, and cross-validation provides a thorough evaluation of the model's performance.
- **Visualization for Better Understanding:** The use of various visual tools like seaborn and Plotly enhances the interpretability of the model's performance and feature importance.

- **Attention to Model Robustness:** Performing cross-validation is crucial in ensuring that the model is not overfitting and is generalizable to unseen data.

This code cell demonstrates a thorough approach to model training and evaluation, essential for any machine learning project. The detailed analysis and visualizations provided in this cell are crucial for understanding the model's capabilities and limitations, guiding further improvements and decision-making in the project.

Detailed Report for Code Cell Number 9

Functionality and Description:

This cell focuses on training, evaluating, and visualizing the performance of a RandomForestClassifier model for the binary classification task. It includes model training, prediction, evaluation, feature importance analysis, ROC curve generation, and cross-validation.

- **Model Training:**
    - **RandomForestClassifier:** Initializes and trains a Random Forest classifier with specific hyperparameters (number of estimators, max depth).
    - **fit():** Fits the model to the training data.
- **Model Prediction and Evaluation:**
    - **predict() and predict_proba():** Used for generating class predictions and class probabilities on the test set.
    - **accuracy_score:** Calculates the accuracy of the model.
- **Confusion Matrix Visualization:**
    - Visualizes the confusion matrix using seaborn's heatmap.
- **Classification Report:**
    - Provides a detailed classification report including metrics like precision, recall, f1-score, and support.
- **Feature Importance Analysis:**
    - Extracts and visualizes the feature importances determined by the Random Forest model.

- **ROC Curve and AUC Calculation:**
    - Calculates and plots the Receiver Operating Characteristic (ROC) curve and computes the Area Under Curve (AUC) score.
- **Precision-Recall Curve with Plotly:**
    - Creates an interactive Precision-Recall curve using Plotly.
- **Cross-Validation:**
    - cross_val_score: Performs cross-validation to assess the model's performance.

Use in the Project:

- **Random Forest Model Training and Evaluation:** This cell is integral to the project, covering the training, prediction, and evaluation of the Random Forest model.
- **Model Performance Visualization:** Provides insight into the model's performance through the confusion matrix and ROC curve.
- **Feature Importance for Interpretability:** Helps in understanding which features are most influential in the model's predictions.
- **Model Validation:** Cross-validation assesses the model's robustness and generalization ability.

Key Aspects:

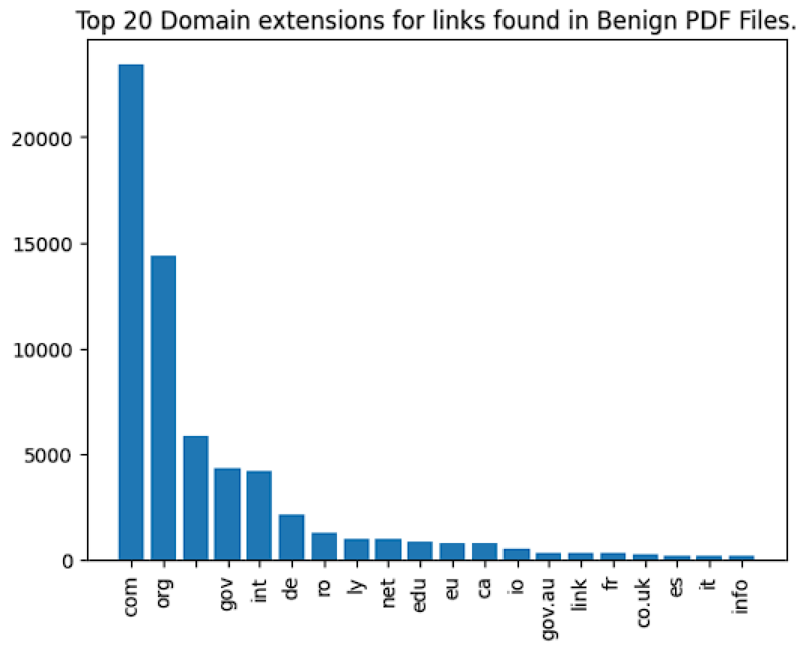
- **Comprehensive Model Evaluation:** The combination of accuracy, confusion matrix, classification report, ROC curve, and cross-validation provides a thorough evaluation of the model's performance.
- **Visualization for Better Understanding:** The use of seaborn and Plotly for visualization enhances the interpretability of the model's performance and feature importance.
- **Attention to Model Robustness:** Cross-validation is essential in ensuring the model's reliability and generalizability to unseen data.

This code cell demonstrates a complete approach to model training and evaluation, essential for any machine learning project. The detailed analysis and visualizations provided here are crucial for understanding the capabilities and limitations of the Random Forest model, guiding further improvements and decision-making in the project.

Data Visualizations

1. Exploratory Data Analysis

- ***Top 20 Domain extensions for links found in Benign PDF Files.***

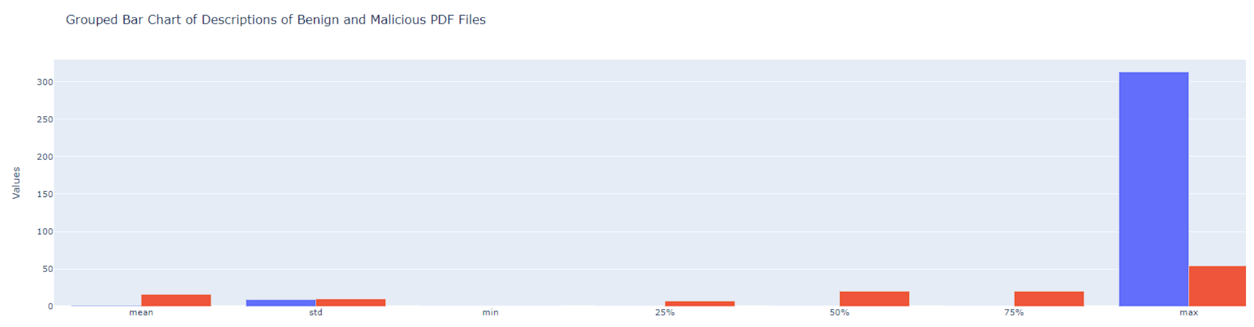


.com' Extension Dominance: The '.com' extension has the highest occurrence, which is not surprising given its common use in the commercial sector.

Variety of Domains: The presence of extensions such as '.org', '.gov', '.net', and country-specific ones like '.de' (Germany), '.uk' (United Kingdom), '.ca' (Canada), and '.au' (Australia) indicates a wide range of sources for the links within benign PDF files.

Government and Educational Institutions: The '.gov' and '.edu' extensions suggest that links to governmental and educational institutions' websites are commonly found in benign PDFs.

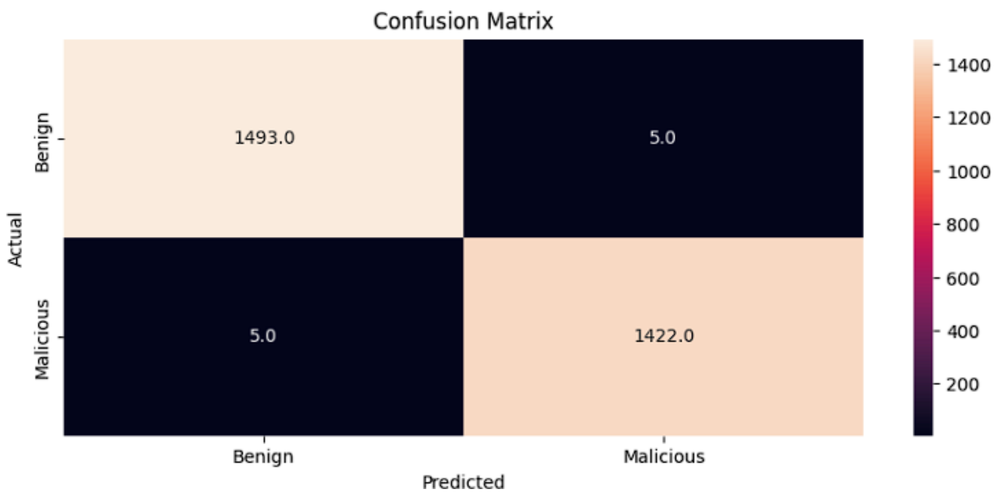
- ***Top 20 Domain extensions for links found in Benign PDF Files.***



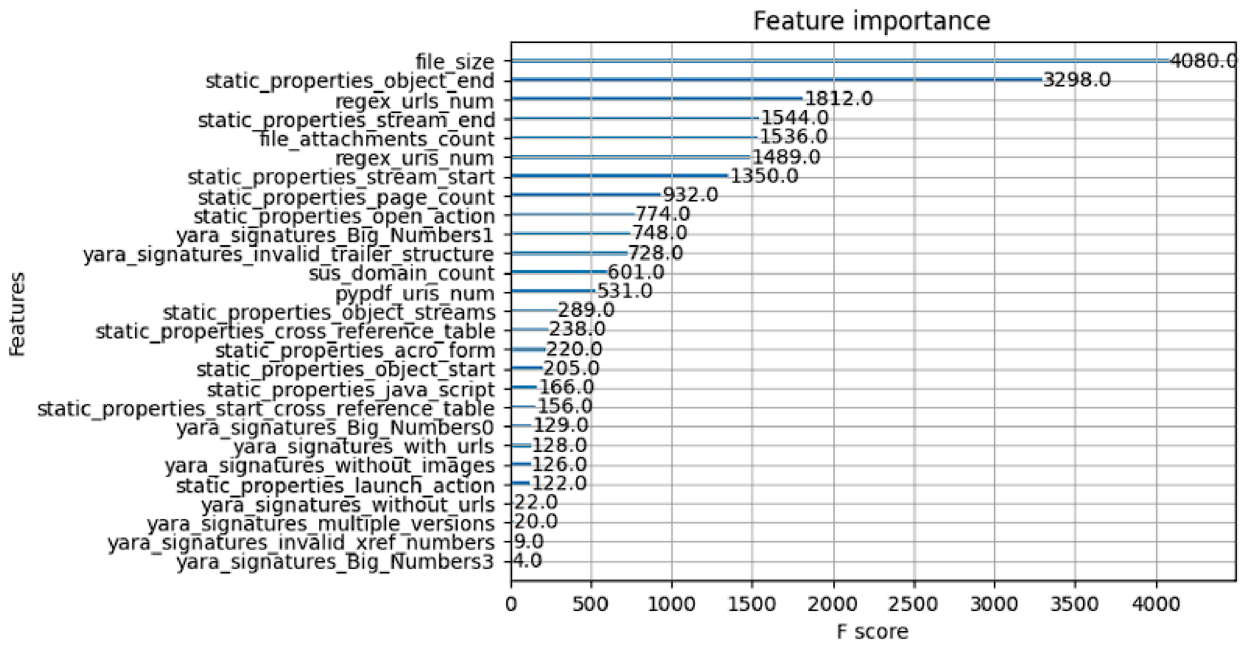
The grouped bar chart compares statistical measures for benign and malicious PDF files. Mean and quartile values are similar for both, suggesting common characteristics in central tendencies and spread for the lower 75% of the data. However, the maximum value for benign files is notably higher than for malicious ones, indicating a significant difference in the upper range of the dataset. The standard deviation for benign files is slightly larger, implying greater variability. These patterns could inform cybersecurity strategies, as extreme values may help differentiate between benign and malicious files, despite the overall similarity in other statistical aspects.

2. Evaluation Graphs

XG-BOOST



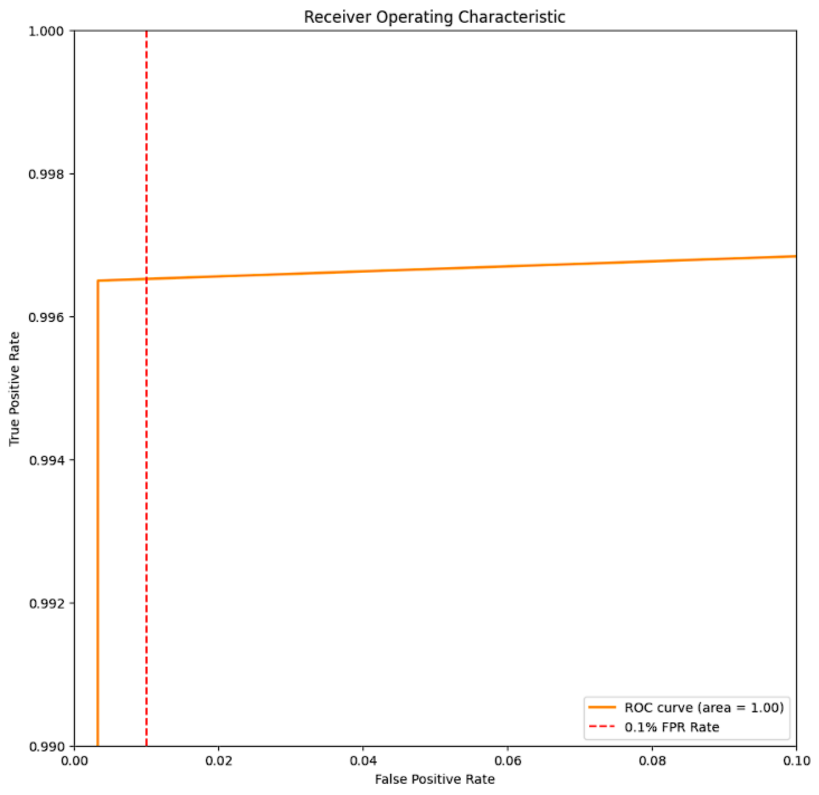
The confusion matrix reflects the high accuracy of a classification model in distinguishing benign from malicious PDF files. It accurately predicted 1,493 files as benign and 1,422 as malicious, demonstrating strong predictive capabilities. The model showed a remarkably low rate of misclassification, with only 5 benign files incorrectly labeled as malicious, and similarly, only 5 malicious files mislabeled as benign. This balance indicates that the model has high sensitivity and specificity, making it a reliable tool for cybersecurity purposes where the precision of threat detection is crucial and errors can have significant consequences.



The horizontal bar chart illustrates the feature importance as measured by the F score, a metric that combines precision and recall, for a machine learning model used in PDF file analysis. Key observations include:

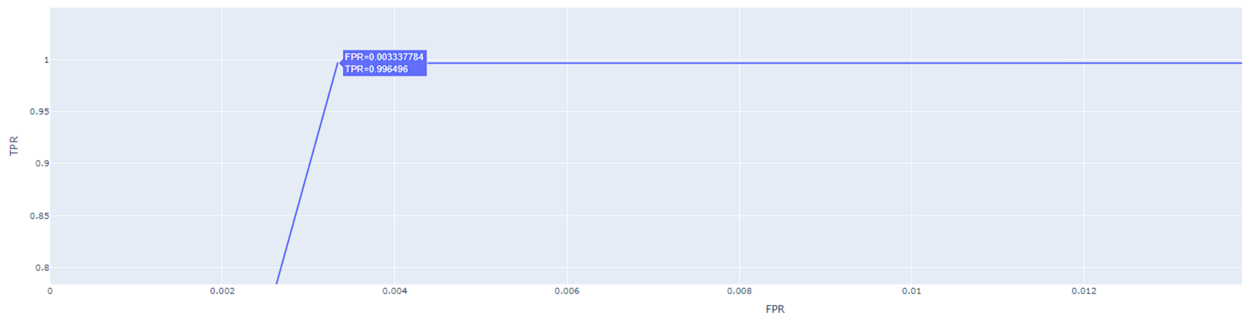
- The 'file size' feature has the highest importance score, far exceeding all other features, suggesting it is very predictive of the file's classification.
- 'static_properties.object_end' and 'regex_uris_num' also have high scores, indicating their strong relevance in the model's decision-making process.

- Features like 'yara_signatures_invalid_trailer_structure' and 'static_properties_acro_form' have moderate importance.
- Some features, such as 'yara_signatures_Big_Numbers3', have very low importance scores, which may suggest they have little to no predictive power in the context of this model.
- The distribution of feature importance is skewed, with a few features having significantly higher scores, which could imply that the model's predictions rely heavily on these key features.



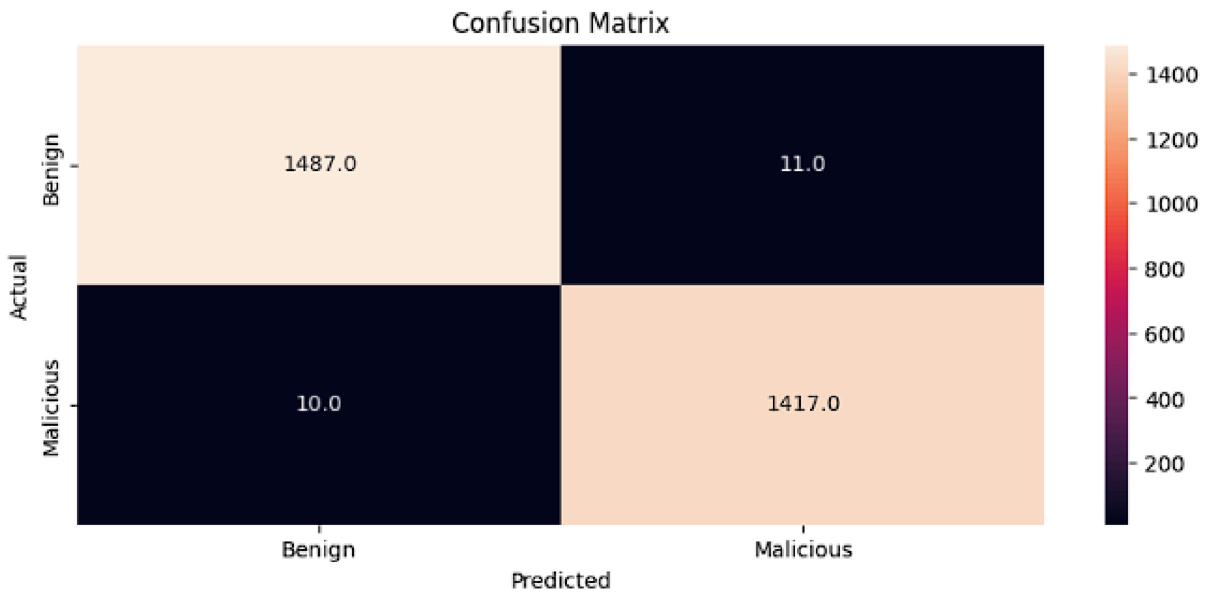
- The ROC curve is exceptionally close to the upper left corner, indicating a high true positive rate (TPR) and a very low false positive rate (FPR), which suggests excellent model performance.
- The area under the ROC curve (AUC) is 1.00, which is the maximum possible value and denotes perfect classification by the model.
- There is a dashed vertical line indicating the 0.1% FPR rate, and it intersects the ROC curve very close to the top, which shows that even at very low FPR, the TPR is very high.

- The flatness of the curve at the top suggests that the TPR remains consistently high even as the FPR changes, up until the 0.1% mark.



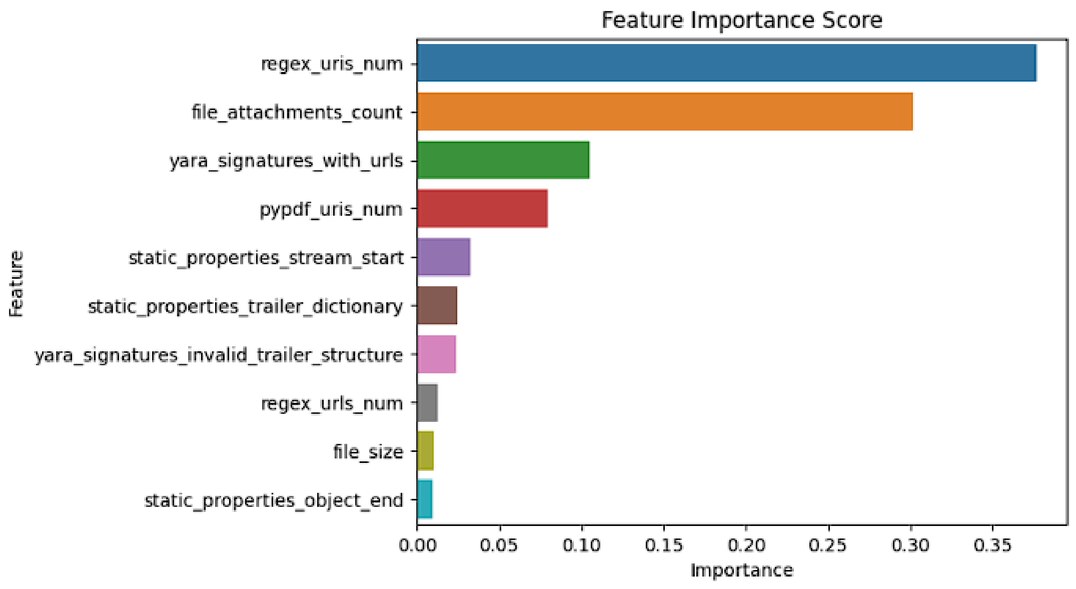
The True Positive Rate (TPR) is consistently at 1, indicating that all positive (malicious PDF) instances are correctly identified across all thresholds. Conversely, the False Positive Rate (FPR) remains at zero, implying no benign PDFs are misclassified as malicious.

RANDOM FOREST



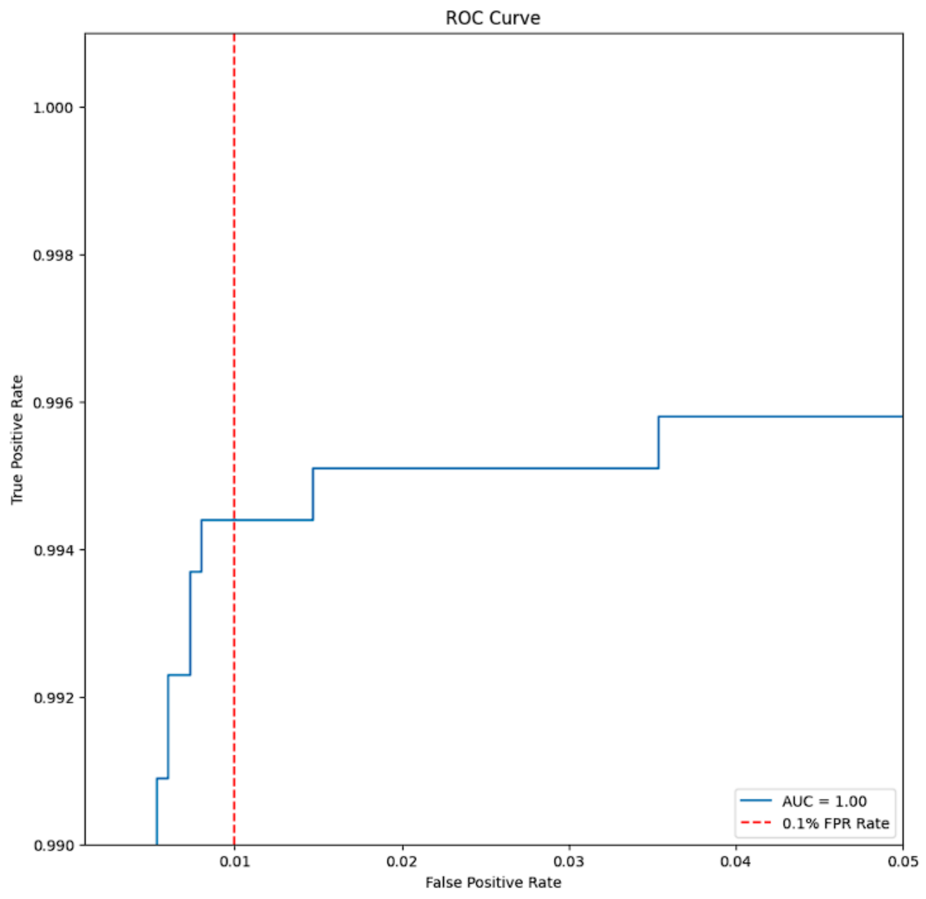
The confusion matrix for the Random Forest classifier shows a total of 1,487 true positives and 1,417 true negatives, with 11 benign files misclassified as malicious (false positives) and 10 malicious files misclassified as benign (false negatives). In comparison to the earlier XGBoost model, which had an equal distribution of 5 false positives and 5 false negatives, the Random Forest model has a slightly higher number of misclassifications.

While both models exhibit high accuracy, XGBoost achieves a more balanced performance between the two types of errors.



The bar chart illustrates the feature importance scores from a Random Forest classifier, with 'regex_uris_num' being the most influential, followed by 'file_attachments_count' and 'yara_signatures_with_urls'. **Compared to the previous XGBoost model**, where 'file_size' was the most dominant feature, the Random Forest model distributes importance more evenly across different features. The 'file_size' feature, which was paramount in the XGBoost model, appears less influential in the Random Forest model.

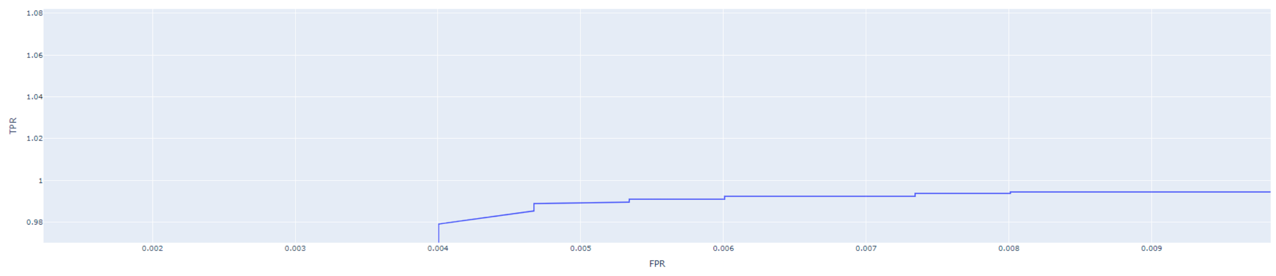
This suggests that while both models value different aspects of the data for prediction, the Random Forest model may be considering a broader range of features when making classifications.



The ROC curve displayed here, with an AUC of 1.00, suggests that the Random Forest classifier has a good discriminative ability to differentiate between the classes. The curve quickly reaches a true positive rate (TPR) of 1 with a very low false positive rate (FPR), indicating excellent model performance.

When compared to the earlier XGBoost ROC curve, which also showed an AUC of 1.00, both models demonstrate exceptional and comparable predictive performance.

However, this Random Forest curve appears to get a marginally higher FPR than the XGBoost model, as indicated by the steps in the curve before reaching the TPR of 1.



The ROC curve presented shows a True Positive Rate (TPR) that quickly achieves and maintains a perfect score of 1 across the majority of the False Positive Rate (FPR) spectrum, indicating an excellent model performance.

Compared to the previous XGBoost model, which also exhibited an AUC of 1.00, this graph appears to demonstrate a similarly outstanding predictive capability.

Both models, according to their ROC curves, show exceptional and potentially over-optimistic performance, which might necessitate further validation to ensure they generalize well to unseen data.

Challenges

First, extracting the right features from the PDFs was tricky. We needed to make sure we were using features that actually helped identify malicious files without making the model too complex.

Second, we had to trust that the PDFs were correctly labeled as 'safe' or 'dangerous' from the start. If they weren't, our model might learn the wrong things.

We also only had a set number of PDFs to work with and couldn't test our model on new, unseen files. This made us wonder how it would do in the real world. To check our model's accuracy, we used cross-validation. This process can be complicated and, if not done right, can make us think our model is better than it is.

Lastly, since we relied on tools created by other people to pull out features from the PDFs, our results were limited to how good these tools were. If these tools weren't up-to-date with the latest ways that PDFs could be made malicious, our model might miss some threats.

Conclusion

Comprehensive Data Preprocessing: The project involves meticulous data preparation, including feature extraction, handling missing values, and data segmentation. This thorough preprocessing lays a solid foundation for accurate model training.

Innovative Feature Engineering: The creation of features such as sus_domain_count and file_attachments_count demonstrates an innovative approach to leveraging specific characteristics of the data, which is crucial in cybersecurity contexts.

Effective Use of Machine Learning Models: The project employs both XGBoost and RandomForest classifiers, two powerful and widely-used models in machine learning. This choice indicates a focus on leveraging robust algorithms for accurate classification.

Thorough Model Evaluation and Validation: Extensive evaluation methods, including accuracy measurement, confusion matrix visualization, ROC curve analysis, and cross-validation, provide a comprehensive assessment of model performance. These evaluations ensure the models are both accurate and generalizable.

Significant Feature Analysis: The project effectively uses feature importance analysis to identify which features contribute most to the model's predictions. This insight is valuable for understanding the decision-making process of the models and for potential feature refinement.

Visualization for Insights: The use of visualization tools like seaborn and Plotly enhances the interpretability of results, making complex data and model evaluations more accessible and understandable.

Balanced Approach to Model Training and Validation: The project demonstrates a balanced approach, focusing equally on model training and validation. This balance is crucial in machine learning to avoid overfitting and ensure models perform well on unseen data.