Machine Learning project: Classifying EMBER2024 Dataset

Link to the project

Sara Caddeo
Andrea Congiu
Matteo Matta

EMBER2024 dataset

The EMBER2024 dataset contains malicious and benign files uploaded to VirusTotal over 64 weeks.

The dataset is being provided already split in three subset:

- training set
- test set
- challenge set

File Type	Train	Test	Challenge	Total
Win32	1,560,000	360,000	3,225	1,923,225
Win64	520,000	120,000	814	640,814
.NET	260,000	60,000	805	320,805
APK	208,000	48,000	256	256,256
ELF	26,000	6,000	386	32,386
PDF	52,000	12,000	805	64,805

A lightweight subset of PDF files was chosen to decrease training/validation time while maintaining meaningful results.

PDF train	197 MB
PDF test	46 MB

Python code to download and vectorize train and test datasets:

```
import thrember
thrember.download_dataset("/path/to/download/to/")
thrember.create_vectorized_features('/path/to/dataset/')
X_train, y_train = thrember.read_vectorized_features('/path/to/dataset/', subset="train")
X_test, y_test = thrember.read_vectorized_features('/path/to/dataset/', subset="test")
X_challenge, y_challenge = thrember.read_vectorized_features('/path/to/dataset/', subset="challenge")
```

EMBER2024- A Benchmark Dataset for Holistic Evaluation of Malware Classifiers

Project goal: malware or benign?

The aim of this project is to design a machine learning model to classify PDF files, between malicious and benign.

EMBER2024 dataset has the following characteristics:

- balanced
- binary
- medium-large size
- over 2000 features

These caracteristics lead us to consider just a few number of possible models that can be used for our classification problem.

LightGBM, RandomForest and other models

Although LightGBM has been indicated as the used model by the authors of EMBER2024, other models have been considered for classification.

Why *LightGBM*?

- Proposed in the paper
- Great for large datasets
- Low memory usage

Why *RandomForest*?

- Decisional trees: parallelisable process
- Random feature selection
- Handles well large datasets with many features

Why not *K-NN*? Why not *SVM*?

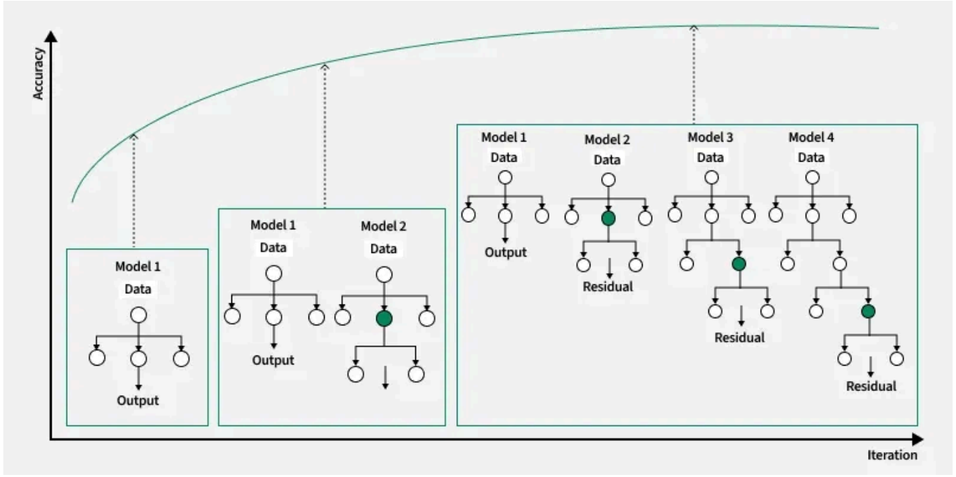
- Computationally expensive and high memory consuming with larger datasets

LightGBM vs RandomForest

LightGBM

How it works:

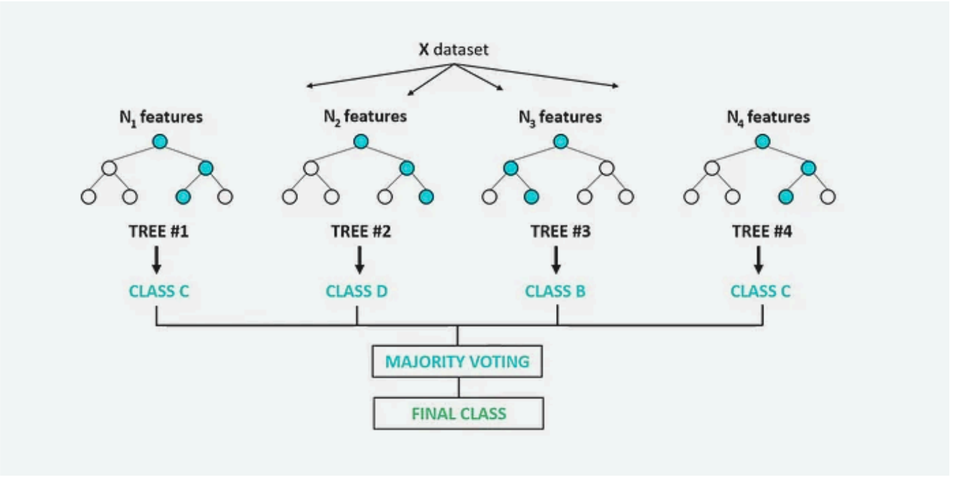
- trees sequential modelling
- each tree is built improving the last one
- focus on previous errors
- precise tuning needed
- higher risk of overfitting



Random Forest

How it works:

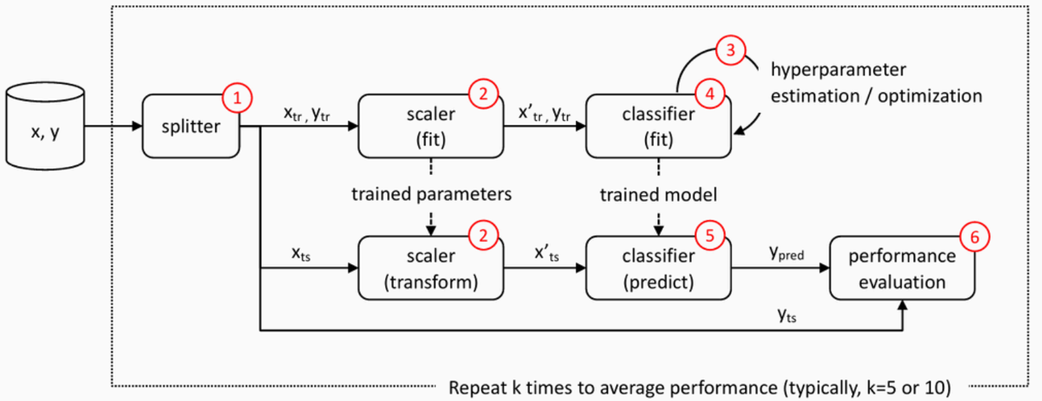
- parallel tree-based modelling
- random samples assigned for each tree
- each tree is independent from the others
- prediction made by votation or mean
- no need of precise tuning



ML System Design

The followed process consisted in these phases:

1. Sampling a train and a test set
2. **Scaling** train and test data
3. Estimating classifier **hyperparameters** on training data
4. **Fitting** the classifier on training data
5. **Predicting** the class labels of testing data
6. Computing **metrics** for performance evaluation



Design choices adopted during:

- Sampling → features already provided with the dataset.
- Scaling → use of “MinMaxscaler” and removal of duplicates.
- Estimating parameters → performed through “GridSearch”
- Predicting → no need to use challenge set
- Evaluation → ROC and PR metrics used in the paper

Hyperparameters estimation

Hyperparameters used for GridSearch function:

- **n_estimators** → number of trees in the “forest”
- **max_depth** → maximum number of node’s levels a tree possesses from the root to the leaves
- **max_features** → number of features considered at each split (level)
- **min_samples_split** → minimum number of samples for splitting a node

Parameters’ possible values have been chosen looking for a trade-off among:

- run-time
- example values on the internet
- accuracy

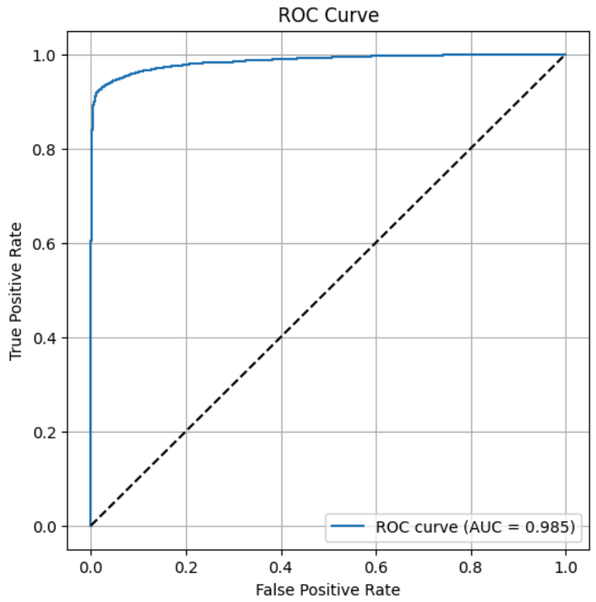
```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)

param_grid = {
    "n_estimators": [100, 200],
    "max_depth": [10, 20, None],
    "max_features": ["sqrt", "log2"],
    "min_samples_split": [2, 5],
}
clf = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=3,
    scoring="accuracy",
    n_jobs=-1
)
```

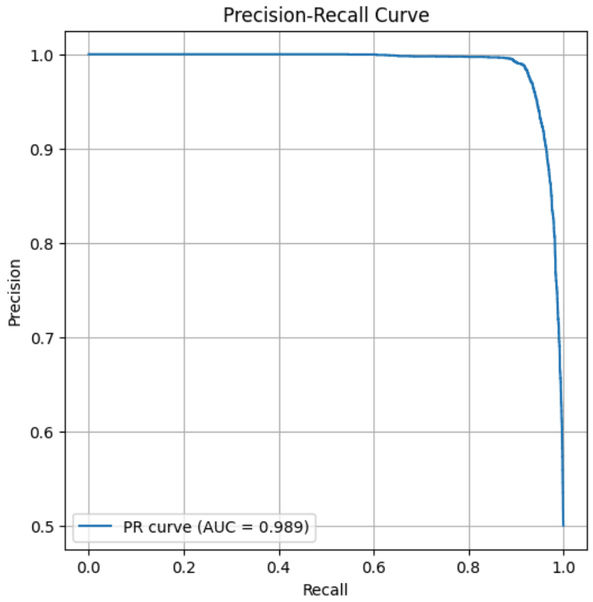
RandomForest metrics

Best parameters for RandomForest model provided by GridSearch:

```
model = RandomForestClassifier(random_state=42,
                               n_jobs=-1,
                               n_estimators= 200,
                               max_depth= 20,
                               max_features= "log2",
                               min_samples_split= 2)
```

Obtained metrics:






```
ROC-AUC:   0.9854
PR-AUC:    0.9885
Accuracy:  0.9427
```

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = TPR = \frac{TP}{TP + FN}$$

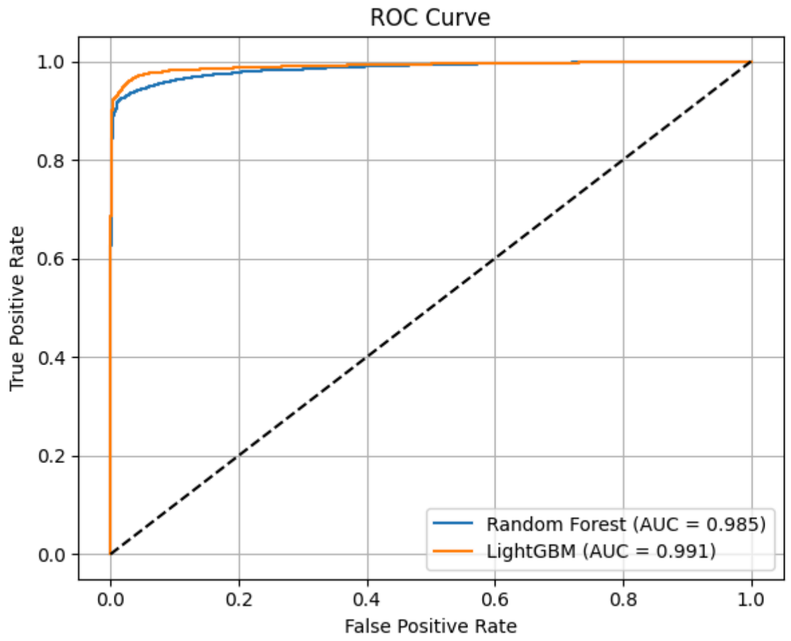
$$FPR = \frac{FP}{FP + TN}$$

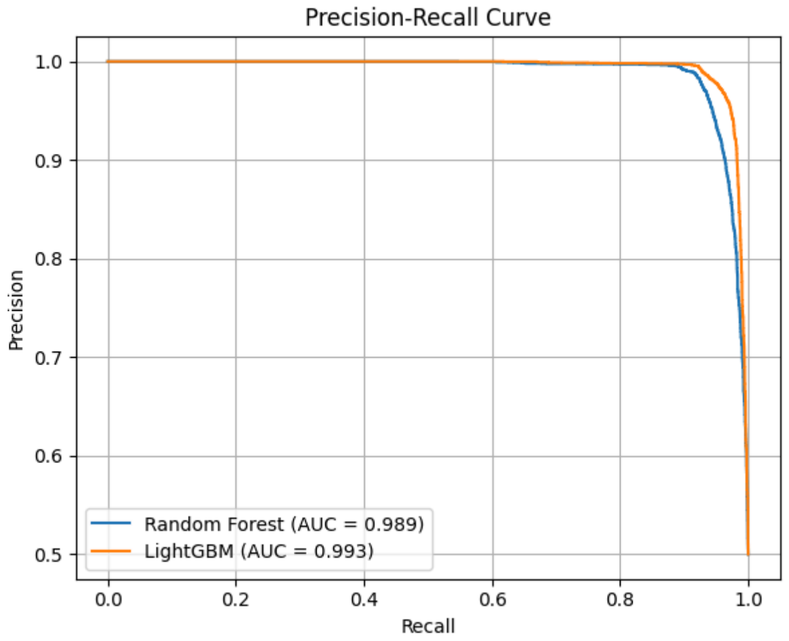
LightGBM vs RandomForest

LightGBM achieves the best results, nevertheless RandomForest is a similar model and shows high values of ROC-AUC, PR-AUC and accuracy.

Turns out LightGBM is better at discriminating between benign and malicious files.
For a FPR = 1% it has gotten:

- TPR = 0.928 (LightGBM)
- TPR = 0.914 (RandomForest)





Random Forest

```
ROC-AUC:   0.9854
PR-AUC:    0.9885
Accuracy:  0.9427
```

LightGBM

```
ROC-AUC:   0.9911
PR-AUC:    0.9932
Accuracy:  0.9590
```

Final conclusions

In conclusion, a machine learning system has been designed to classify the EMBER2024 PDF dataset and its performance has been compared with the LightGBM model proposed in the EMBER2024 paper.

The proposed system is based on a Random Forest model: we optimized its hyperparameters, used it to predict the labels of the test dataset, and then computed evaluation metrics.

As expected, we found that LightGBM achieves better results overall.
However, Random Forest remains a valid alternative thanks to its ease of tuning and lower risk of overfitting.