

01001001 01101110 01100110 01110010 01100001 01101110 01100100 01100101 01110010



EmberEye

Production-Ready Binary Detector & Analysis Toolkit

For Network/Traffic Data Analysis



CatBoost



2.2M Rows



99.9% Accuracy

Project Overview and Vision

EmberEye is a **production-ready binary detector and analysis toolkit** for network data.

- 🕒 Historically known as **Fruty** or **EmbeReye**
- ✅ Emphasizes **reproducibility** with clear artifacts
- 🔗 Identify network events using **CatBoost model**
- 🗄️ Primary dataset: **combine.csv** (2.2M rows)

Key Components

- 📦 Trained models in **models/**
- 📈 Experiment results in **results/**
- 🔧 Scripts in **src/**
- 📄 EDA in **notebooks/**



Core Objectives



Production-Ready Detector

- ✓ Produce **single, high-performing CatBoost** detector
- ✓ Package as **models/final_detector.joblib**
- ✓ Ensure readiness for production environments



Robustness Validation

- ✓ Validate through **sampled Cross-Validation**
- ✓ Conduct **permutation importance** analysis
- ✓ Perform **leakage scans** and reproducibility tests



Simplified Tooling

- ✓ Provide **straightforward inference tooling**
- ✓ Include **reproducibility checklist** for auditors
- ✓ Facilitate deployment and auditing processes

Dataset Foundation


combine.csv Dataset

 **Filename:** combine.csv





 **Size:** Approximately 2.2 million rows


 **Features:** Around 78 features


 **Target Column:** Label

 Always confirm header names, as some CSVs may contain leading or trailing whitespace. Explicit checks are recommended.

Features Breakdown

Feature	Type
feature_1	 Numeric
feature_2	 Categorical
feature_3	 Numeric
feature_4	 Categorical

 Numeric
Features

 Categorical
Features

*Preview of feature
types*

Technology Stack



</> Programming

Python 3.x

Core language for project



⚙️ ML Frameworks

CatBoost

High performance model

scikit-learn

For utilities



🗄️ Data Tools

pandas

Data manipulation

numpy

Numerical computing



💾 Persistence

joblib

Model saving/loading



📊 Visualization

matplotlib

Static plots

seaborn

Statistical graphics

plotly

Interactive visualizations



🔧 Optional

shap

Model interpretability

>_ Environment

environment-catboost.yml

Conda env for CatBoost

requirements.txt

Pip dependencies

Project Architecture

The EmberEye repository is organized to ensure clarity and ease of navigation for users. The structure separates different aspects of the project, from raw data to final model artifacts.



EmberEye



models



results



src



notebooks

models/

Stores trained models, including [final_detector.joblib](#).

results/

Contains experiment outputs like JSON summaries and figures.

src/

Houses training and validation scripts for CatBoost model.

notebooks/

Contains exploratory data analysis and diagnostic notebooks.

combine.csv

The primary dataset used for training, residing in root.

Other Assets

Includes environment files and documentation.

Environment Setup and Dependencies

1 Conda Environment Setup

For Windows users, use the Conda environment file to avoid CatBoost build issues:

```
$ conda env create -f environment-catboost.yml -n embereye  
$ conda activate embereye
```



Creates a dedicated environment with necessary dependencies.

2 Pip Dependencies

For additional utilities, install packages from **requirements.txt**:

```
$ pip install -r requirements.txt
```



Recommended for full functionality of notebooks and scripts.

Key Points to Remember



Windows users: Use Conda to avoid CatBoost build issues



Environment isolation: Use dedicated environments



Verify installation: Check all dependencies are installed

Core Execution Pipeline

Five-step workflow ensuring reproducibility and facilitating auditing:

1



Train CatBoost Model

Trains the CatBoost model on combine.csv

```
python  
src\train_catboost_on_  
--data combine.csv  
--out_dir results
```

2



Run Diagnostics

Performs permutation importance and cross-validation

```
python  
src\catboost_checks.py  
--model  
models/catboost_raw.jo  
--data combine.csv
```

3



Finalize Production Bundle

Prepares the model as production-ready detector

```
python  
src\finalize_model.py  
--src  
models/catboost_raw.jo  
--dst  
models/final_detector.
```

4



Smoke-Load Final Model

Quick test to ensure the model loads successfully

```
python  
src\_smoke_load_catboo  
--model  
models/final_detector.
```

5



Perform Inference

Enables predictions on new input data

```
python  
src\predict_with_catbo  
--model  
models/final_detector.  
--input  
sample_input.csv  
--output  
results\preds.csv
```



Model Training Process

⚙️ Training Workflow

1 Prepare Data



2 Configure Training



3 Train Model



4 Save Artifact

> Training Command

```
python src\train_catboost_on_raw.py --data  
combine.csv --out_dir results --model_out  
models/catboost_raw.joblib
```

⚙️ Key Parameters

- data**: Path to training dataset
- out_dir**: Directory for experiment outputs
- model_out**: Path to save the trained model
- sampling**: Optional sampling parameters
- seed**: Random seed for reproducibility

Diagnostic and Validation Framework

Validation Methods



Permutation Importance

Evaluates feature significance by randomizing values



Sampled Cross-Validation

5-fold CV on 100k rows: **0.99872** (std = 0.00019)



Leakage Detection

Identifies data leakage through mutual information analysis

Key Validation Results

Best Threshold

0.496

Best Accuracy

0.99905 (\approx 99.9046%)

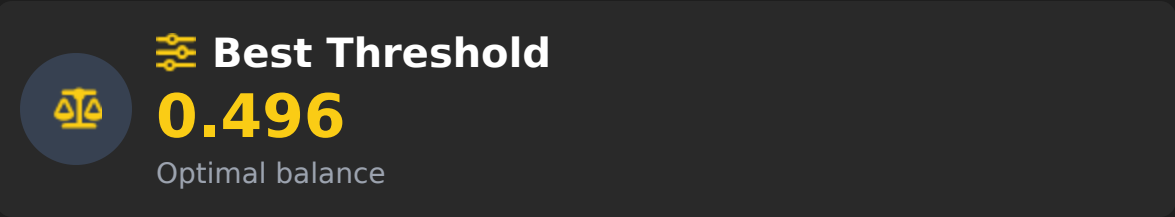
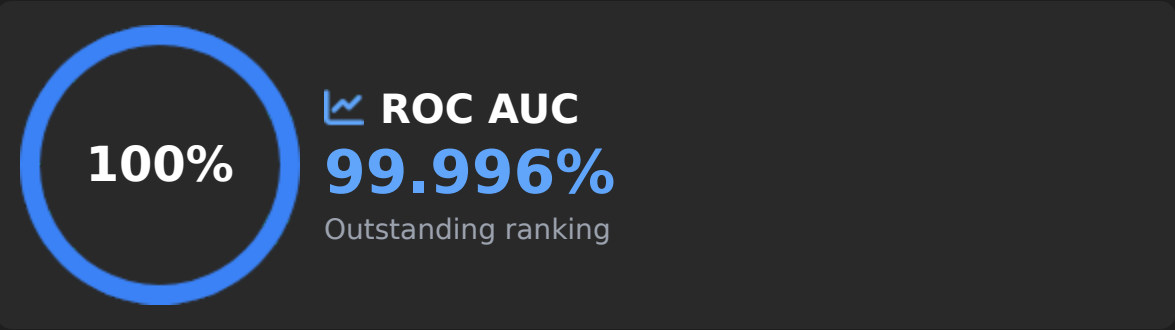
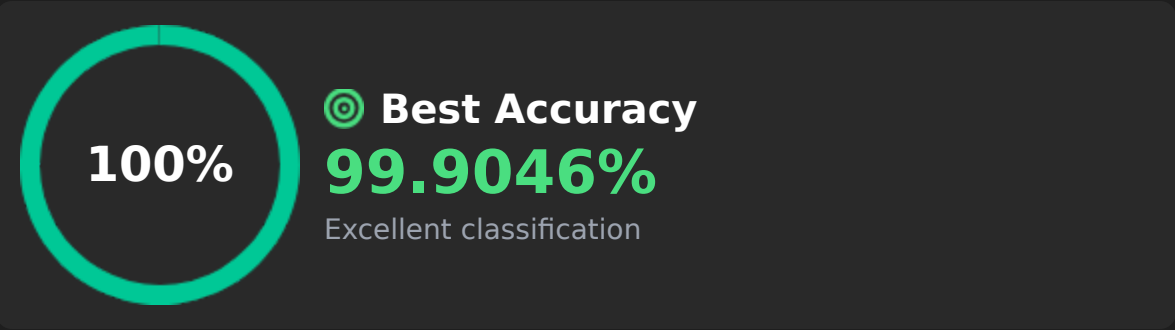
ROC AUC

0.99996

Confusion Matrix

TN: 250,691	FP: 234	FN: 83	TP: 81,161
True Negatives	False Positives	False Negatives	True Positives

Performance Metrics Excellence



📊 Confusion Matrix

Predicted	Class 0	Class 1
Actual Class 0	250,691	234
Actual Class 1	83	81,161

🔄 Cross-Validation Score

5	100K	99.872%	0.019%
Folds	Samples	Mean Acc.	Std Dev.

Visualization and Analysis Tools

Interactive Visualization Notebook



Jupyter notebook for:

- Load sampled subsets and model artifacts
- Plot class balance and feature distributions
- Render correlation heatmaps and projections

 Use [notebooks/visualization.ipynb](#) for exploration

Script-driven Exports

Automated scripts for CI-friendly exports:

-  Non-interactive execution
`jupyter nbconvert --to notebook --execute visualization.ipynb`
-  Output formats
PNG/HTML figures exported to `results/figs/`

Supported Visualization Types



Class Balance

Visualize class distribution



Correlation Heatmap

Display feature correlations



Permutation Importance

Rank features by importance




SHAP Summary Plot

Show feature impacts


Model Comparison and Selection

During development, several models were evaluated to identify the most suitable candidate for production deployment.

Model	Key Characteristics	Performance	Selection Status
✓ CatBoost	<ul style="list-style-type: none">High performance with categorical features	Best accuracy: 99.9046% ROC AUC: 0.99996	Selected for production Optimal balance of performance and reliability
⌚ LightGBM / XGBoost	<ul style="list-style-type: none">Used in Optuna tuning and stacking experiments	Did not outperform CatBoost at chosen threshold	Not selected Higher AUC observed in some configs




Rationale for CatBoost Selection




Consistently High Metrics

Reproducible high accuracy across validation tests



Ease of Packaging

Simplified deployment into production environments



Inference Efficiency

Optimal balance of performance and practicality

Large File Management



Git LFS Implementation

This repository tracks large assets with Git LFS.

- Tracks combine.csv (dataset)
- Tracks models/*.joblib (models)
- Tracks data/processed/*.npz



Collaboration Setup

After cloning the repository, run:

```
git lfs install
git lfs pull
```

This installs Git LFS and downloads large files.



Usage Notes

GitHub enforces LFS quotas. Monitor usage in:
Repository Settings → Packages/LFS



Troubleshooting

- "File exceeds 100 MB" - Ensure LFS installed
- "LFS bandwidth exceeded" - Reduce usage

Production Deployment Considerations



Windows Compatibility

For CatBoost on Windows, use the provided conda environment:

```
conda env create -f environment-catboost.yml
-n embereye
conda activate embereye
```



CSV Header Validation

Always confirm header names in your CSV files:

- Some CSVs may contain **leading/trailing whitespace**
 - The **Label** column is used for binary detection
- ⚠️ **Verify target column naming**



Sampling Strategies

For large datasets, sampling is recommended:

- Many operations use **100k-200k rows** samples
- Use **--sample** flag in scripts for consistent sampling



Model Persistence

The final model is saved as:

```
models/final_detector.joblib
```

This file contains the complete production-ready detector

Future Enhancements and Contributions



SHAP Analysis Expansion

Run SHAP TreeSHAP on sampled subsets (10-20k rows) to generate detailed feature contribution plots and save them to [results/figs/](#).



Model Comparison Tables

Create comprehensive per-model comparison tables by extracting metrics from [results/*.json](#) and adding them to [README_FULL.md](#).



CI/CD Integration

Add GitHub Actions smoke tests to validate model loading and perform tiny inference checks, ensuring ongoing model integrity.



Community Contributions

Contributions welcome! Open an issue or submit a pull request with reproducible tests for any code changes.



Your Input Matters

suggest new features or improvements by opening an issue on the repository. The team welcomes your feedback to enhance EmberEye's capabilities.

Project Impact and Conclusion

Key Achievements

🎯 Exceptional Detection

99.9% accuracy with ROC AUC of 0.99996

🔄 Complete Reproducibility

Validation through CV, permutation importance, and leakage scans

🔑 Production-Ready

Single CatBoost model packaged as `final_detector.joblib`

🚀 Deployment Framework

Simplified inference tooling with reproducibility checklist

Best Accuracy

99.9046%

on test dataset

ROC AUC

0.99996

highly accurate ranking

Best Threshold

0.496

optimal operating point