EcoFinder Team — Probabilistic Prioritization of Exoplanet Candidates

Authors: Matias Graña • Manuel Krapf • Lucas Jarpa

Abstract

We present **EcoFinder**, a reproducible pipeline and web service for *probabilistic prioritization* of exoplanet candidates from Kepler/TESS mission catalogs. EcoFinder exposes (i) a frontend UI for interactive scoring, (ii) a backend API (Gradio + Flask) for programmatic access, and (iii) training notebooks for two supervised models: a **Multilayer Perceptron (MLP, Keras)** and an **XGBoost** classifier. Using transit-derived features (period, duration, depth, stellar/planetary properties, SNR, transit count), EcoFinder estimates the likelihood that a candidate will become a *confirmed exoplanet*, thus accelerating scientific vetting and follow-up.

1 System Overview

EcoFinder helps researchers and citizen scientists prioritize **Kepler** candidates for follow-up by estimating the probability that a "candidate" becomes a **confirmed exoplanet**. It provides a **web UI**, a **public API**, and **reproducible training** notebooks for two models: **MLP (Keras)** and **XGBoost**.

Live Links

- **Frontend (live demo):** https://e58733b4f52c223cfa.gradio.live/
- **Backend API — Keras classifier (Hugging Face Space):**
  Live Web API: https://huggingface.co/spaces/jarpalucas/echo-finder-api
  Source: https://github.com/lucasjarpadev/echo-finder-api-exoplanets/tree/main/backend-in-hugg
  BackendAPIWithKeras
- **Backend API — XGBoost classifier (Hugging Face Space):**
  Live Web API: https://huggingface.co/spaces/jarpalucas/eco-finder-api-xgboost
  Source: https://github.com/lucasjarpadev/echo-finder-api-exoplanets/tree/main/backend-in-hugg
  BackendAPIWithXGBoosting

The Spaces expose both a **Gradio** UI and a **Flask** REST API.

2 Data & Features

2.1 Data Sources

EcoFinder consumes **NASA Exoplanet Archive** catalogs via official REST/TAP endpoints:

- Training labels from **KOI (Kepler Objects of Interest)**: *CONFIRMED*, *CANDIDATE*, *FALSE POSITIVE*.
- "Unseen/unknown" objects for live tests from **TOI** (TESS Objects of Interest) and **TCE** (Threshold Crossing Events).

2.2 Transit-Based Features (KOI-like template)

`koi_period, koi_duration, koi_depth, koi_prad, koi_srad, koi_teq, koi_steff, koi_slogg, koi_smet, koi_kepmag, koi_model_snr, koi_num_transits`

Normalization & Imputation: Field-name *synonym mapping* (TOI/TCE → KOI-like schema), *median imputation* for missing values, and *standardization* (scaler).

Reproducibility Artifacts:

- Keras (default): `modelo_tabular.h5`, plus `scaler.pkl`, `label_encoder.pkl`, `feature_stats.json`.
- XGBoost (optional): `xgb_model.pkl`, re-using the same scaler/encoder/stats.

3 Models

3.1 MLP (Keras) — Default Deployed Model

- **Task:** Multiclass classification (`CONFIRMED` / `CANDIDATE` / `FALSE POSITIVE`).
- **Architecture:** `Dense(128)-Dropout(0.3)-Dense(64)-Dropout(0.2)-Dense(32)-Dense(softmax)`.
- **Training:** Stratified split, 25 epochs (tunable), standardized inputs.
- **Export:** `modelo_tabular.h5` (+ shared scaler/label artifacts).

Training snippet (Keras):

```
num_classes = len(np.unique(y_tr))
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(X_tr.shape[1],)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['
    accuracy'])
history = model.fit(X_tr, y_tr, validation_data=(X_te, y_te), epochs=25, batch_size=32,
    verbose=1)
model.save("modelo_tabular.h5")
```

Recommended visuals: accuracy/loss curves; confusion matrix on held-out test set.

3.2 XGBoost — Alternative Deployed Model

- **Task:** Same labels; strong tabular performance and feature importances.
- **Baseline hyperparameters:** `n_estimators=400, max_depth=5, learning_rate=0.05, subsample=0.9, colsample_bytree=0.9`.
- **Export:** `xgb_model.pkl` (+ shared scaler/label artifacts).

Training snippet (XGBoost):

```python
from xgboost import XGBClassifier
xgb = XGBClassifier(
    n_estimators=400, learning_rate=0.05, max_depth=5,
    subsample=0.9, colsample_bytree=0.9, random_state=42, eval_metric='mlogloss'
)
xgb.fit(X_tr, y_tr)
import joblib
joblib.dump(xgb, "xgb_model.pkl")
```

Operationally we can **switch** between MLP and XGBoost in the web UI or via deployment flags.

4 Backend API (Gradio + Flask on Hugging Face)

4.1 Endpoints

Method	**Path**	**Description**
GET	`/health`	Service status and model info
GET	`/features`	Feature names, medians, descriptions
POST	`/predict`	Single prediction (JSON body)
POST	`/predict-batch`	Batch prediction (`{"objects": [...]}`)

Expected JSON (KOI-like)

```json
{
  "koi_period": 10.0,
  "koi_duration": 5.0,
  "koi_depth": 1000.0,
  "koi_prad": 2.0,
  "koi_srad": 1.0,
  "koi_teq": 1000.0,
  "koi_steff": 6000.0,
  "koi_slogg": 4.5,
  "koi_smet": 0.0,
  "koi_kepmag": 12.0,
  "koi_model_snr": 10.0,
  "koi_num_transits": 3.0
}
```

cURL Examples

Health

```bash
curl -s https://<space>.hf.space/health
```

Single prediction

```bash
curl -s -X POST https://<space>.hf.space/predict \
  -H "Content-Type: application/json" \
  -d '{
    "koi_period": 10.0,
    "koi_duration": 5.0,
```

```
    "koi_depth": 1000.0,
    "koi_prad": 2.0,
    "koi_srad": 1.0,
    "koi_teq": 1000.0,
    "koi_steff": 6000.0,
    "koi_slogg": 4.5,
    "koi_smet": 0.0,
    "koi_kepmag": 12.0,
    "koi_model_snr": 10.0,
    "koi_num_transits": 3.0
  }'
```

Batch prediction

```
curl -s -X POST https://<space>.hf.space/predict-batch \
  -H "Content-Type: application/json" \
  -d '{
    "objects": [
      {"koi_period": 10.0, "koi_duration": 5.0, "koi_depth": 1000.0, "koi_prad": 2.0,
       "koi_srad": 1.0, "koi_teq": 1000.0, "koi_steff": 6000.0, "koi_slogg": 4.5,
       "koi_smet": 0.0, "koi_kepmag": 12.0, "koi_model_snr": 10.0, "koi_num_transits":
           3.0}
    ]
  }'
```

4.2 Deployment Artifacts (per Space)

- **Keras (default):** `modelo_tabular.h5`, `scaler.pkl`, `label_encoder.pkl`, `feature_stats.json`.
- **XGBoost (optional):** `xgb_model.pkl` (reuses the same scaler/encoder/stats).

Ensure real binaries (not Git LFS/Xet pointers). If a file is only a few bytes or shows "pointer", re-upload the actual artifact.

5 Reproducibility & Evaluation

- **Training notebooks (place in `/notebooks`):**
  `Exoplanet_Train_Export_Colab-USADO.ipynb` (Keras MLP)
  `Exoplanet_Train_Export_Colab-XGB.ipynb` (XGBoost)
- **Evaluation artifacts:** accuracy/loss learning curves (MLP), confusion matrix, class-wise precision/recall/F1, optionally ROC-AUC macro.
- **Operational logging:** model versioning, feature medians, and scaler snapshot included with each prediction.

6 Roadmap

- **Scale the backend:** Containerize (Docker) and deploy to **Kubernetes** with N replicas and load balancing; centralize logs/metrics.

- **Model zoo & on-demand training:** Add LightGBM/TabNet/calibrated models; enable dataset uploads and training jobs from the web; surface validation metrics and artifact versioning.
- **Richer data sources:** Integrate **TESS** feeds and direct TAP for **TOI/TCE**; automate name unification and missing-value handling.
- **CSV UX for non-experts:** Enable CSV downloads from TOI/TESS in a human-readable template; allow CSV re-uploads for scoring; provide probability reports; build a **Hall of Fame** crediting users who surface promising candidates.

7 Local Execution (Optional)

```
# create env
python -m venv .venv && source .venv/bin/activate

# install
pip install -r requirements.txt

# place artifacts next to app.py
# - modelo_tabular.h5 (or xgb_model.pkl)
# - scaler.pkl, label_encoder.pkl, feature_stats.json

# run
python app.py
# Flask API on :5000, Gradio on :7860
```

8 Acknowledgments

NASA Exoplanet Archive (Kepler/TESS catalogs: KOI, TOI, TCE; official APIs). We gratefully acknowledge the teams and infrastructure behind these datasets and tools.

9 License

Choose an OSI-approved license (e.g., MIT/Apache-2.0) and add it as `LICENSE` in the repository.