



# Grupo 4

# Modelo Predictivo de Machine Learning para la Identificación Del Riesgo de Extinción de Especies Terrestres

*Integrantes:*

- **Roly Alvaro Huacre Samaniego - 20196156**
- **Marcelo Daniel Olivares Cornejo - 20221953**
- **Johar Ricarte Cubas Castro -20195806**
- **Karolyn Nayumi Aquiño Torres - 20203608**
- **Mikler Junior Diaz Perez**

# Introducción al peligro de extinción de especies terrestres

- La pérdida de biodiversidad es una amenaza global , con cerca de un millón de especies animales en riesgo de extinción debido a la acción humana
- El principal obstáculo para la clasificación del riesgo de extinción es que una gran parte de las especies carece de suficiente información
- La justificación del proyecto radica en la necesidad de usar Inteligencia Artificial (IA) predictiva para estimar el estatus de amenaza de estas especies DD. El enfoque se centra en la predicción basada en la integración de rasgos biológicos, ecológicos y filogenéticos



# Hipótesis

**¿El modelo Random Forest (RF), al integrar rasgos biológicos, ecológicos y filogenéticos, alcanza un desempeño (medido por Balanced Accuracy y F1-score) que valida su uso como herramienta predictiva robusta y generalizable para clasificar el riesgo de extinción de especies terrestres con datos deficientes?**



# Objetivos

- **Seleccionar los rasgos relevantes para clasificar el riesgo de extinción.**
- **Entrenar y optimizar modelos predictivos supervisados.**
- **Evaluar y comparar el desempeño de los modelos.**

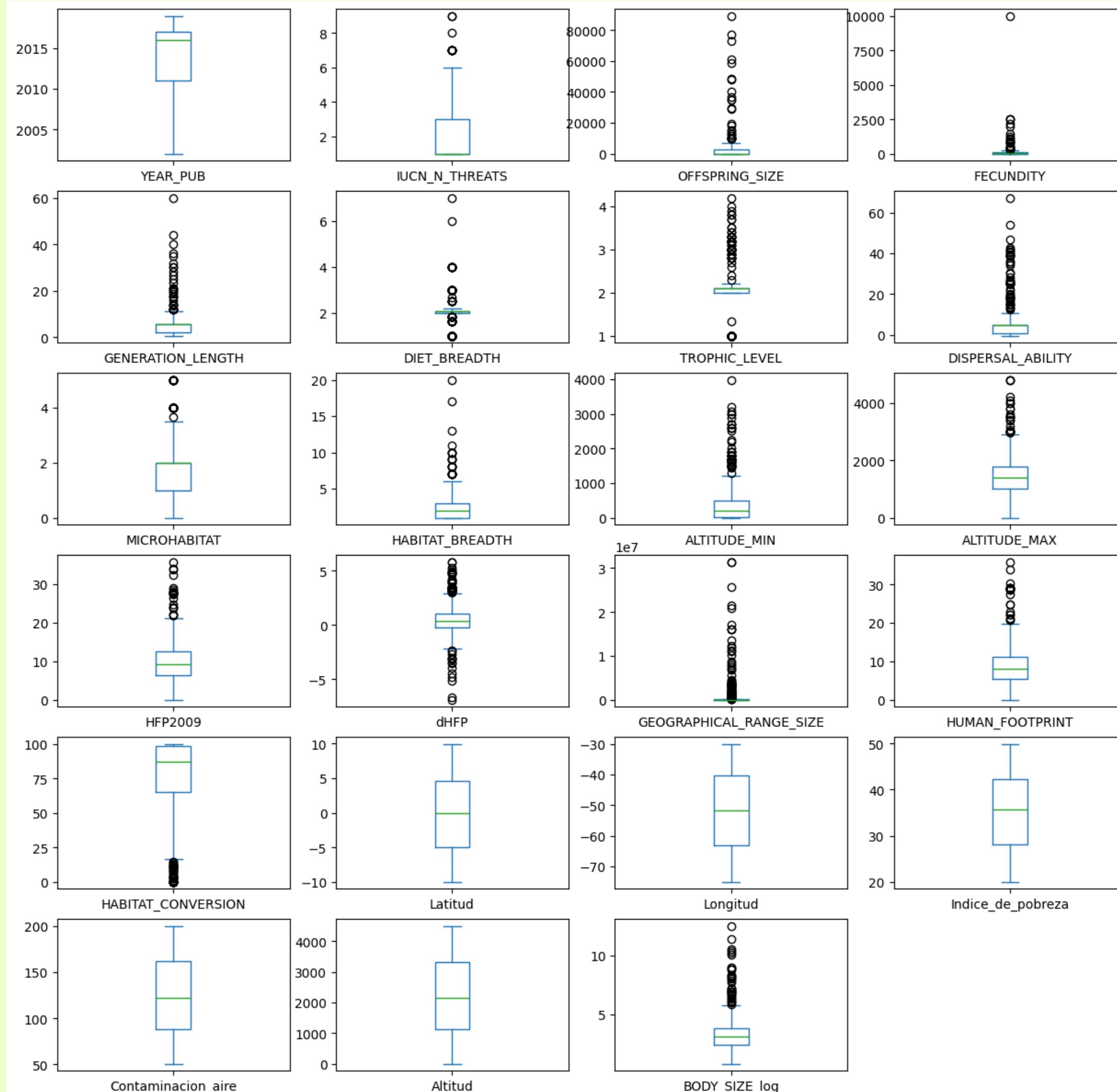
# Preprocesamiento de los datos:

Tras analizar los registros del dataset se elimino los datos de todas las plantas dentro del dataset, ya que solo nos enfocamos en animales.

- Se estandarizan todas la variables numéricas como ALTITUDE\_MIN.
- Se eliminaron instancias sin clasificación de riesgos. Se usaron métodos como IQR y se uso winsorizing para ajustar limites.
- SMOTE → incrementar muestras minoritarias
- El One-Hot Encoding para variables taxonómicas y ambientales
- BODY\_SIZE se transformó con  $\log(x+1)$  para reducir sesgo y atenuar valores extremos.
- Se eliminaron valores redundantes, altamente correlacionadas
- Las variables del target Status se binarizo.
- Se estandarizaron nombres de columnas para compatibilidad con scikit-learn

```
# Vemos información acerca de los atributos del dataset
df.info()
# Descripción estadística de las variables numéricas
df.describe()
```

0	Group	879	non-null	object
1	Kingdom	879	non-null	object
2	Phylum	879	non-null	object
3	Class	879	non-null	object
4	Order	879	non-null	object
5	Family	879	non-null	object
6	Genus	874	non-null	object
7	IUCNName	879	non-null	object
8	NCBIName	876	non-null	object
9	Status	879	non-null	object
10	CRITERIA	331	non-null	object
11	YEAR_PUB	819	non-null	float64
12	Realm	879	non-null	object
13	IUCN_THREATS_CATEG	566	non-null	object
14	IUCN_N_THREATS	569	non-null	float64
15	BODY_SIZE	847	non-null	float64
16	OFFSPRING_SIZE	463	non-null	float64
17	FECUNDITY	334	non-null	float64
18	GENERATION_LENGTH	346	non-null	float64
19	DIET_BREADTH	214	non-null	float64
20	TROPHIC_LEVEL	263	non-null	float64
21	DISPERSAL_ABILITY	334	non-null	float64
22	MICROHABITAT	551	non-null	float64
23	HABITAT_BREADTH	874	non-null	float64
24	ALTITUDE_MIN	547	non-null	object
25	ALTITUDE_MAX	558	non-null	float64
26	HFP2009	558	non-null	float64
27	dHFP	558	non-null	float64
28	GEOGRAPHICAL_RANGE_SIZE	538	non-null	float64
29	HUMAN_FOOTPRINT	553	non-null	float64
30	HABITAT_CONVERSION	565	non-null	float64
31	Latitud	879	non-null	float64

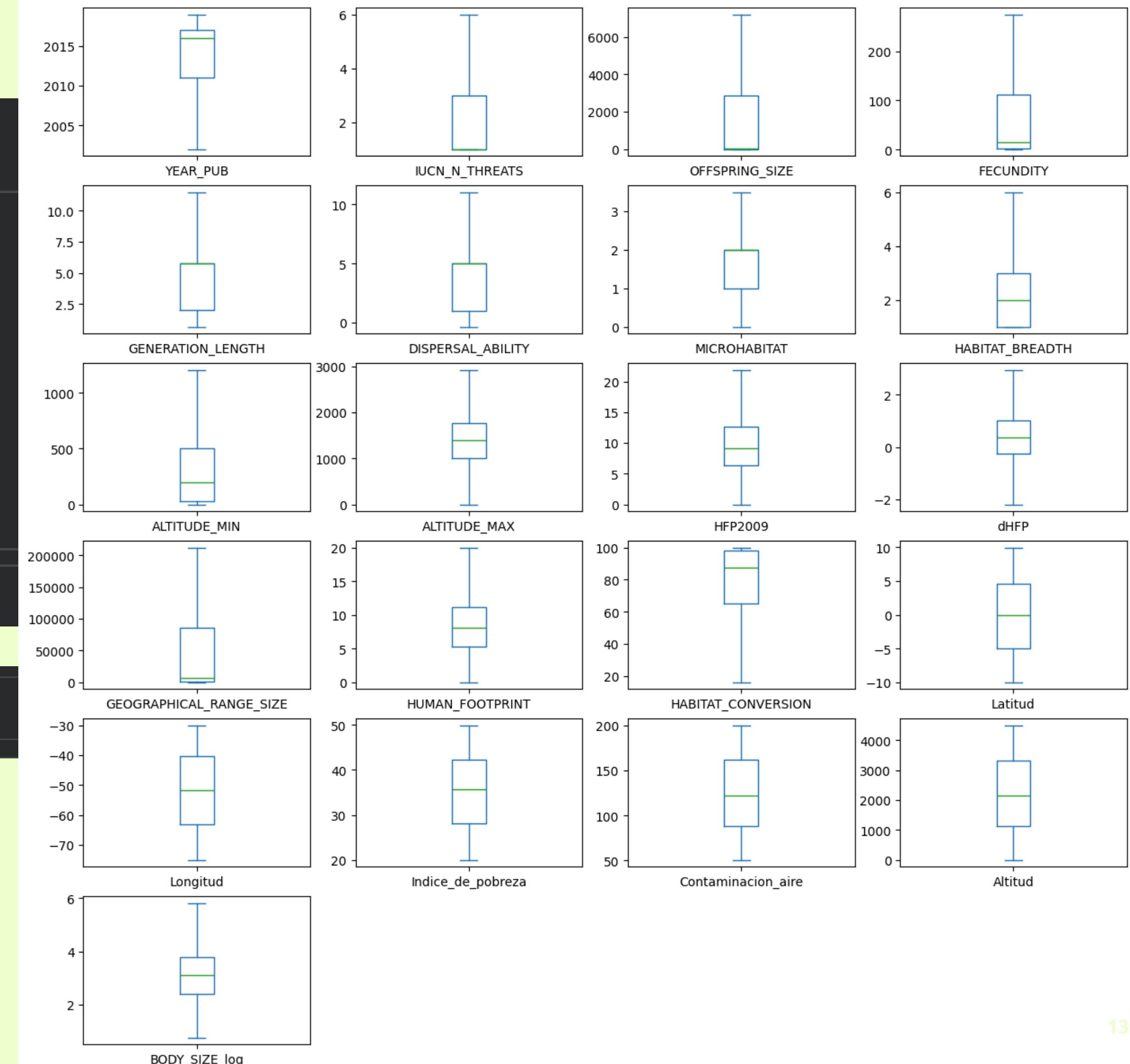


## Variables numéricas

```
def corrige_outliers(dfx, columnas_lst):
    for columna in columnas_lst:
        q1 = dfx[columna].quantile(0.25)
        q3 = dfx[columna].quantile(0.75)
        iqr = q3 - q1
        ul = q3 + 1.5*iqr # upper_limit
        ll = q1 - 1.5*iqr # lower_limit
        dfx.loc[dfx[columna]>ul, columna] = ul
        dfx.loc[dfx[columna]<ll, columna] = ll
    return dfx
```

```
df = corrige_outliers(df, columnas_numericas)
```

```
df = df.drop(columns=['DIET_BREADTH', 'TROPHIC_LEVEL'])
```



```

dropped_corr_cols = df.corr(numeric_only=True).apply(lambda x: round(x,2))
plt.figure(figsize=(15,12))
sns.heatmap(dropped_corr_cols,
            annot=True,
            cmap='Blues')
plt.show()

```

```

for c in columnas_categoricas:
    if c !='IUCN_THREATS_CATEG':
        print("Columna", c)
        print(df[c].unique())
        print()

```

#### One hot encoding

```

onehot_cols = [
    'Group', 'Kingdom', 'Phylum', 'Class', 'Order',
    'Family', 'Genus', 'Realm', 'Uso_del_suelo',
    'Lat_bin', 'Lon_bin'
]

df_encoded = pd.get_dummies(df, columns=onehot_cols, drop_first=True)

```

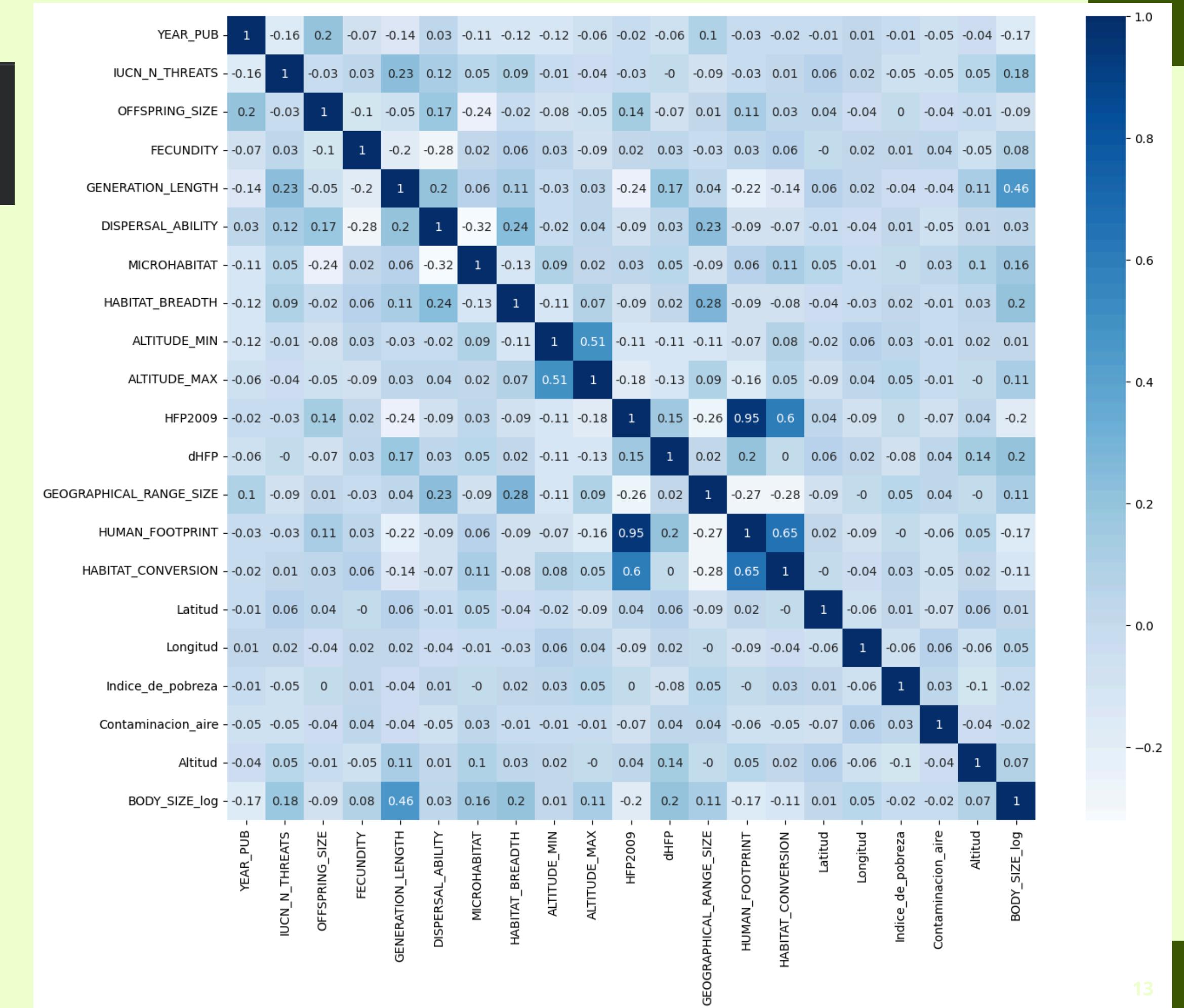
#### Mapeamos la variable target = 'Status' ya que tiene un orden implícito

```

status_map = {
    'LC': 0,
    'NT': 0,
    'VU': 1,#1
    'EN': 1,#1
    'CR': 1,
    'EW': 1,
    'EX': 1
}

df_encoded['Status'] = df['Status'].map(status_map)

```





# Experimentación y Resultados

# DISEÑO Y CONFIGURACION EXPERIMENTAL

## ■ Preprocesamiento y Partición de Datos:

**Filtrado:** Solo Animalia (eliminado Plantae)

**Binarización:**

- Clase 0: LC + NT (356, 61.5%)
- Clase 1: VU + EN + CR + EW + EX (223, 38.5%)

**Eliminación:** Correlación > 0.94 removida

**Partición estratificada 80/20:**

- Train: 463 especies
- Test: 116 especies

**579 especies | 787 features**

## ■ Modelos y Configuración de Hiperparámetros:

- Logistic Regression
- CART
- Random Forest
- XGBoost

```
dataset = df_encoded

array = dataset.values
X = dataset.drop(columns='Status') # atributos
y = dataset['Status'] # target

# realiza el train-test split
vadid_fraction = 0.20    # proporcion de muestras para validacion
seed = 7     # semilla para el generador aleatorio (para reproducibilidad)
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=vadid_fraction, random_state=seed)

print('Training: %d ejemplos. Validation: %d ejemplos' % (len(y_train), len(y_valid)))

Training: 463 ejemplos. Validation: 116 ejemplos
```

```

▶ # Evaluamos cada pipeline de ML en estrategia de 10-fold-CV
results = []
names = []

# genera el particionamiento de 5 folds que seran usados en cada evaluacion
seed = 7
kfold = StratifiedKFold(n_splits=5, random_state=RANDOM_STATE, shuffle= True) # especifica el particionamiento

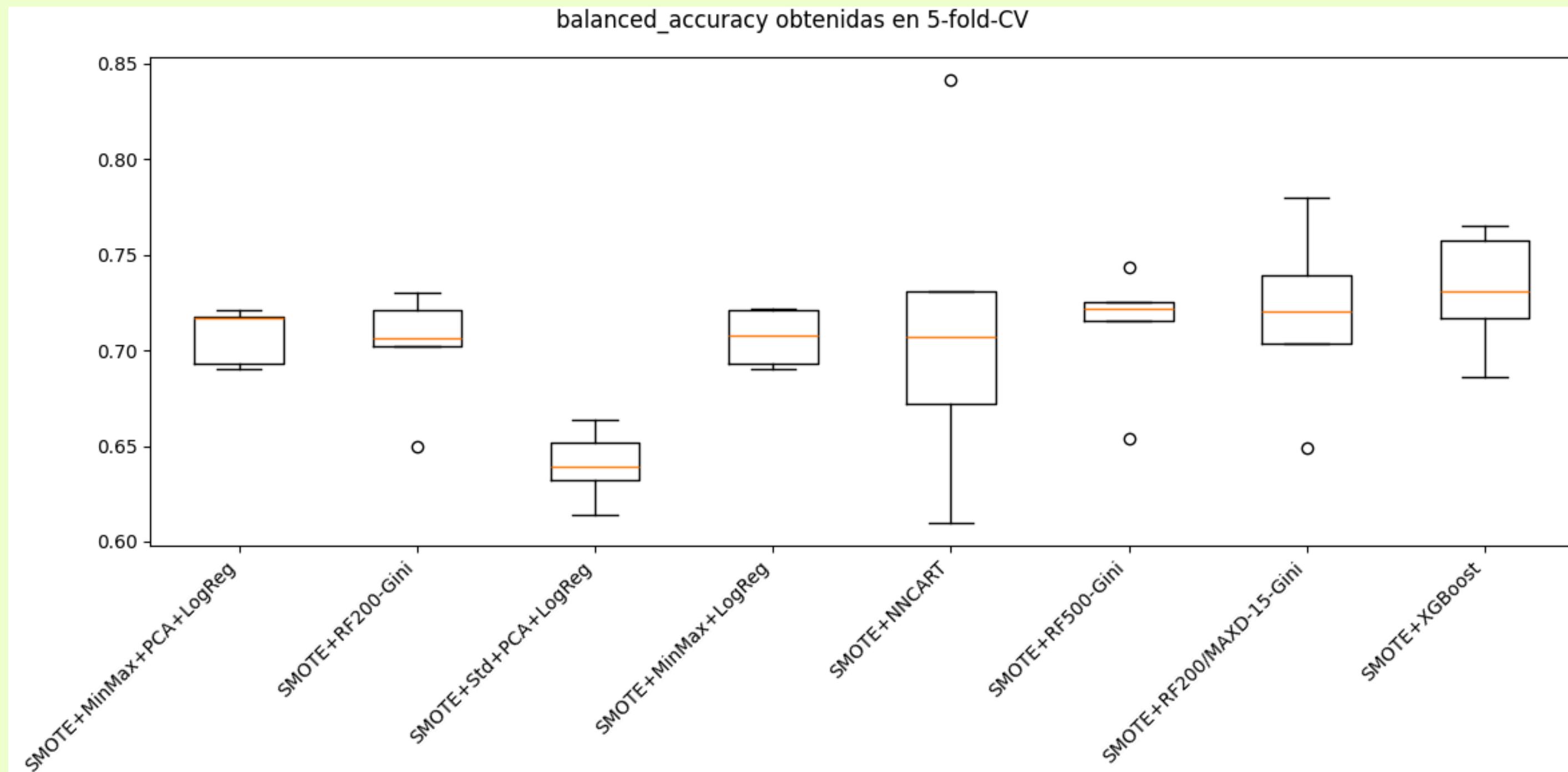
# evalua cada pipeline en crossvalidacion
for pipelinename, pipeline in pipelines:
    cv_results = cross_val_score(pipeline, X_train, y_train, cv=kfold, scoring='balanced_accuracy')
    results.append(cv_results)
    names.append(pipelinename)
    print("{}: {} ({})".format(pipelinename, cv_results.mean(), cv_results.std()))

```

<b>Pipeline</b>	<b>Balanced Accuracy</b>
SMOTE+MinMax+PCA+LogReg	$0.7078 \pm 0.0133$
SMOTE+RF200-Gini	$0.7356 \pm 0.0248$
SMOTE+Std+PCA+LogReg	$0.6419 \pm 0.0180$
SMOTE+MinMax+LogReg	$0.7060 \pm 0.0167$
SMOTE+NNCART	$0.6953 \pm 0.0647$
SMOTE+RF500-Gini	$0.7211 \pm 0.0359$
SMOTE+RF200/MAXD-15-Gini	$0.7174 \pm 0.0416$
SMOTE+XGBoost	$0.7305 \pm 0.0462$

# RESULTADOS Y DISCUSIÓN

## Cuadro Comparativo del accuracy de los modelos



# Cuadro Comparativo del accuracy de los modelos

## Fase 2

```
# Random Forest

nombre_modelo = "Random Forest"
model = make_pipeline(
    SMOTE(random_state=RANDOM_STATE),
    RandomForestClassifier(
        n_estimators=100,
        max_depth = 15,
        criterion='gini',
        max_features='sqrt',
        bootstrap=True,
        min_samples_split=5,
        min_samples_leaf=1,
        random_state=RANDOM_STATE,
        n_jobs=-1,
        class_weight='balanced'
    )
)

# XGBoost

nombre_modelo = "XGBoost"
model = make_pipeline(
    SMOTE(random_state=RANDOM_STATE),
    XGBClassifier(
        random_state=RANDOM_STATE,
        n_jobs=-1,
        colsample_bytree=0.7,
        learning_rate=0.01,
        max_depth=10,
        n_estimators=200,
        subsample=0.8,
        eval_metric='logloss'
    )
)
```

Modelo	Balanced Accuracy	Mejora vs Fase 1	Hiperparámetros óptimos
XGBoost	0.7612	+0.0307 (+4.2%)	n=200, depth=10, lr=0.01, sub=0.8, col=0.7
Random Forest	0.7320	-0.0036 (-0.5%)	n=100, depth=15, gini, sqrt features

**Resumen de métricas de desempeño  
en el conjunto de prueba.**

Modelo	Balanced Accuracy	Precision ▾	Recall	F1-Score
Logistic Regression (LR)	0.64	0.69	0.62	0.65
Classification and Regression Trees (CART)	0.68	0.70	0.65	0.67
Random Forest (RF)	0.73	0.75	0.71	0.73
SMOTE + MinMax + PCA + LogReg	0.71	0.72	0.68	0.70
SMOTE + RF200-Gini	0.70	0.73	0.67	0.70
SMOTE + XGBoost	0.76	0.80	0.74	0.77

**Tiempo de Simulación**

Pipeline	Tiempo (segundos)
SMOTE+MinMax+PCA+LogReg	2.7604
SMOTE+RF200-Gini	1.7481
SMOTE+Std+PCA+LogReg	1.0983
SMOTE+MinMax+LogReg	0.3291
SMOTE+NNCART	0.2672
SMOTE+RF500-Gini	3.1240
SMOTE+RF200/MAXD-15-Gini	0.8284
SMOTE+XGBoost	0.8608

## Comparación de modelo optimizados

<b>Modelo</b>	<b>Balanced Accuracy</b>	<b>Mejora vs Fase 1</b>	<b>Hiperparámetros óptimos</b>
XGBoost	0.7612	+0.0307 (+4.2%)	n=200, depth=10, lr=0.01, sub=0.8, col=0.7
Random Forest	0.7320	-0.0036 (-0.5%)	n=100, depth=15, gini, sqrt features



# CONCLUSIONES



- Este estudio confirma que Random Forest es un modelo adecuado para clasificar el riesgo de extinción de especies con datos deficientes. Alcanzó un Balanced Accuracy cercano a 0.73 y un F1-score mayor a 0.70, superando a modelos como Logistic Regression y CART, lo que valida la hipótesis planteada.
- Los resultados muestran que Random Forest es robusto ante datos incompletos, alta dimensionalidad y desbalanceo, lo que lo convierte en una herramienta confiable para apoyar tareas de conservación.
- Aunque Random Forest cumple la hipótesis, XGBoost obtuvo un rendimiento ligeramente superior tras la optimización, indicando que ambos modelos ensemble representan soluciones sólidas y efectivas para este tipo de problemas.

# ¡Gracias por su atención!

