Predicción de brotes de *Astaena pygidialis* en el cultivo de aguacate has, utilizando métodos de aprendizaje automático.

Oscar Pulgarin
Edna Rincon

Aprendizaje Automático

Junio, 2025

INTRODUCCIÓN

- El aguacate *Peresea americana cv* Hass está tomando gran relevancia en los mercados internacionales.
- La producción mundial fue de 6,4 % anual.
- En Colombia las exportaciones son del126%
- La producción y calidad se está viendo afectado por plagas de importancia económica como lo son el complejo Melolonthidae.

L. Díaz Ramírez et al., (2021).

- *Astaena pygidialis Kirsch* conocido como cucarrón marceño es una de las plagas de mayor importancia
- Generan daños cuando el insecto perfora el material vegetal y dejando lesiones en los frutos.
- Ocasiona perdidas del 30 % de la fruta y ataca entre el 40 % y el 60 % de la fruta formada.

Carabalí Muñoz et al., (2021). T. Kondo et al., (2020)

METODOLOGIA

1. Descripción del Conjunto de Datos

Periodo 2022 y 2023

Recurso abierto disponible en Melendy

Listado de tipos de datos de cada dataframe

```
[ ]
print("pest_marceno ------------------------------------")
print(pest_marceno.dtypes)
print("------------------------------------------------")
print("raw_climate_part1 ------------------------------------")
print(raw_climate_part1.dtypes)
print("------------------------------------------------")
print("raw_climate_part2 ------------------------------------")
print(raw_climate_part2.dtypes)
print("------------------------------------------------")
print("data_harvest ------------------------------------")
print(data_harvest.dtypes)
print("------------------------------------------------")
print("data_trees ------------------------------------")
print(data_trees.dtypes)
print("------------------------------------------------")
```

Finca Rionegro, Antioquia (Colombia).

METODOLOGIA

1. Descripción del Conjunto de Datos

Tabla 1. Datos Variables Climáticas

rawClimatePart1.csv; rawClimatePart2.csv		
Variable	Nomenclature	Unit
Record timestamp	timestamp	YYYY-MM-DD HH:MM
Relative humidity	relativeHumidity	Percent (%)
Air temperature	airTemp	Degrees Celsius (°C)
Rainfall	Rainfall	Millimeters of rainfall (mm)
Wind speed	speedWind	Meters per second (m/s)
Solar radiation	radiationSolar	Watts per square meter (W/m^2)
Raw milivolts value	rawSolarValue	Milivolts (mV)
Solar total flux density	fluxSolar	Megajoules per square meter (MJ/m2)

Tabla 2. Datos Arboles

dataTrees.csv		
Variable	**Nomenclature**	**Unit**
Tree identifier	idTree	
Planting date	datePlanting	YYYY-MM-DD
Latitude	latitude	Decimal degrees (°)
Longitude	longitude	Decimal degrees (°)
Altitude	altitude	Meters above sea level (masl)
Slope orientation	orientSlope	Cardinal and ordinal directions
Slope direction	dirSlope	Degrees (°)
Type of plantation land	typePlantation	
Drainage distance	distDrainage	Meters (m)
Drainage depth	depthDrainage	Meters (m)

Tabla 3. Datos Plaga Marceño

pestMarceno.csv		
Variable	Nomenclature	Unit
Tree identifier	idTree	
Observation date	dateObservation	YYYY-MM-DD
Pest presence	presenceMarceno	Binary (0-1)
Damage to old leaves	oldLeavesMarceno	NA = No damage BA = Low damage ME = Medium damage
Damage to new leaves	newLeavesMarceno	NA = No damage BA = Low damage ME = Medium damage
Damage to fruits	fruitDamageMarceno	NA = No damage BA = Low damage ME = Medium damage

METODOLOGIA

2. Observación y limpieza de datos

Visualización de los datos

```
print("pest_marceno ------------------------------------")
print(pest_marceno.head().to_string(index=False))
print("-------------------------------------------------")
print("raw_climate_part1 ------------------------------------")
print(raw_climate_part1.head().to_string(index=False))
print("-------------------------------------------------")
print("raw_climate_part2 ------------------------------------")
print(raw_climate_part2.head().to_string(index=False))
print("-------------------------------------------------")
print("data_harvest ------------------------------------")
print(data_harvest.head().to_string(index=False))
print("-------------------------------------------------")
print("data_trees ------------------------------------")
print(data_trees.head().to_string(index=False))
print("-------------------------------------------------")
```

```
pest_marceno ------------------------------------
    idTree dateObservation  presenceMarceno oldLeavesMarceno newLea
avocado_01      2022-09-06                0              NaN
avocado_02      2022-09-06                0              NaN
avocado_03      2022-09-06                0              NaN
avocado_04      2022-09-06                0              NaN
avocado_05      2022-09-06                1              NaN
-------------------------------------------------
raw_climate_part1 ------------------------------------
      timestamp  relativeHumidity  airTemp  rainfall  speedWind  d
2022-06-09 00:00              99.1    13.93     0.508        0.0
2022-06-09 00:15              99.3    13.87     0.000        0.0
2022-06-09 00:30              99.4    13.83     0.000        0.0
2022-06-09 00:45              99.5    13.82     0.000        0.0  2
2022-06-09 01:00              99.6    13.82     0.000        0.0  2
```

```
print("Nulos")
print(f"******Pest
print(f"Dimensione
print(f"******pest
print(f"Dimensione
print(f"******raw_
print(f"Dimensione
print(f"******raw_
print(f"Dimensione
print(f"******data
print(f"Dimensione
print(f"******data
print(f"Dimensione
```

```
Nulos
******Pestmarceno
idTree                0
dateObservation       0
presenceMarceno       0
oldLeavesMarceno    101
newLeavesMarceno    163
fruitDamageMarceno  168
dtype: int64
Dimensiones (300, 6)
```

```
# Unificación de archivos de clima (2022-2023)
raw_climate_part1_diario = raw_climate_part1.groupby('timestamp').mean(numeric_only=True).reset_index()
raw_climate_part2_diario = raw_climate_part2.groupby('timestamp').mean(numeric_only=True).reset_index()

# Obtener solo las columnas comunes
columnas_comunes = raw_climate_part1.columns.intersection(raw_climate_part2.columns)

# Filtrar los DataFrames por las columnas comunes
climate1_filtrado = raw_climate_part1[columnas_comunes]
climate2_filtrado = raw_climate_part2[columnas_comunes]

#Combinaciòn de ambas fechas de clima
# raw_climate = pd.merge(raw_climate_part1_diario,raw_climate_part2_diario,on='timestamp',how='outer',suffixes=("_c1","_c2"))
raw_climate = pd.concat([climate1_filtrado, climate2_filtrado], axis=0, ignore_index=True)
print(raw_climate.describe())
print(raw_climate.shape)
```

```
                           timestamp  relativeHumidity       airTemp  \
count                          47432      47358.000000  47358.000000
mean   2023-02-11 20:03:15.665373696         86.783426     16.738648
min              2022-06-09 00:00:00         39.830000      7.820000
25%              2022-10-10 16:41:15         75.710000     13.980000
50%              2023-02-11 05:07:30         94.400000     15.600000
75%              2023-06-15 01:18:45         98.400000     19.580000
max              2023-11-22 14:30:00        100.000000     26.910000
std                              NaN         14.465450      3.557884

           rainfall     speedWind       dirWind
count  47432.000000  47432.000000  47432.000000
```

METODOLOGIA

2. Observación y limpieza de datos

```
# Unión de dataframees de peste y características del árbol
df_marceno_tree = pd.merge(pest_marceno,data_trees,on='idTree',how='l
df_marceno_tree['timestamp'] = df_marceno_tree['dateObservation'].dt
```

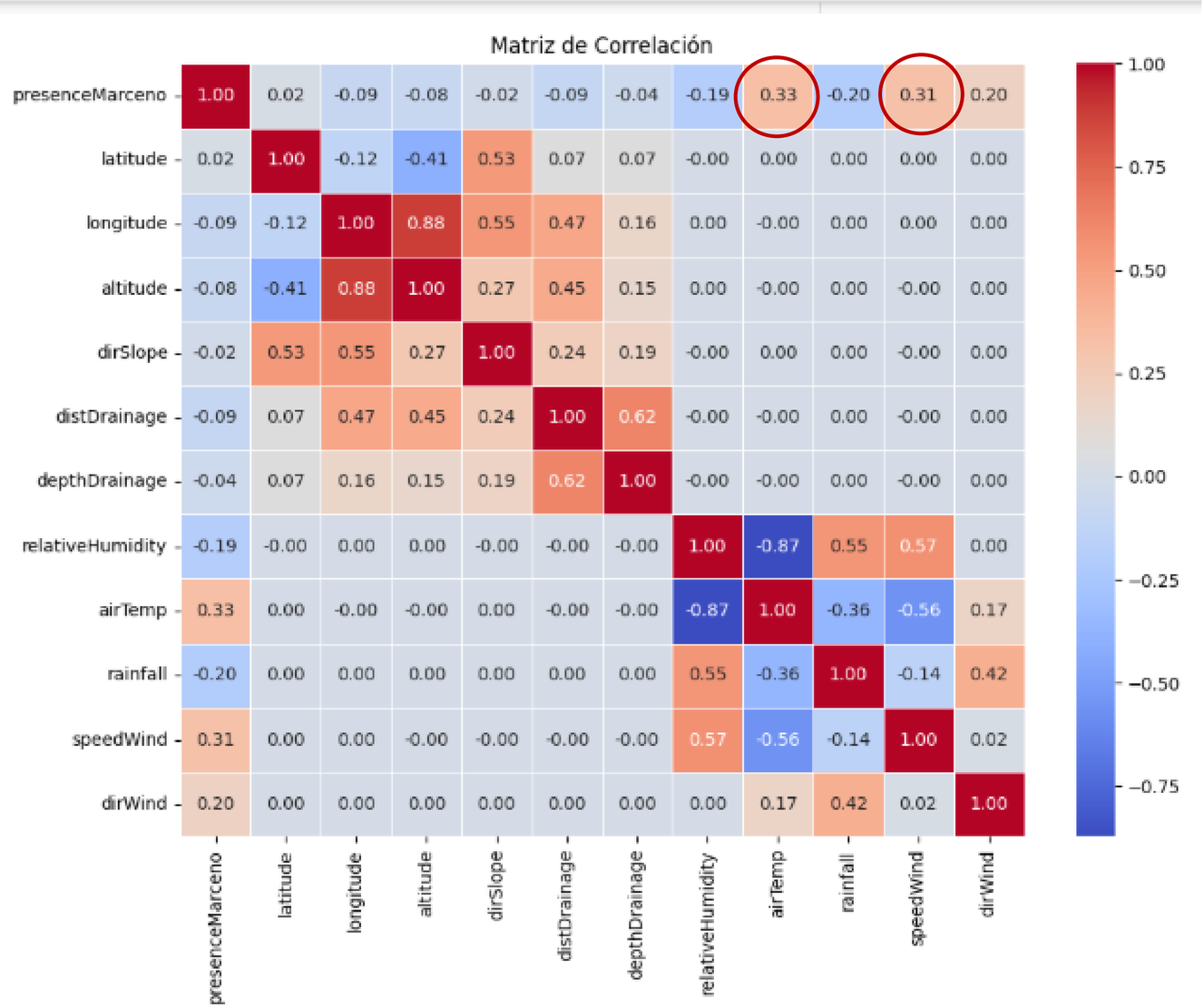
```
from tqdm import tqdm

# Se ordena por arbol y fecha
df_marceno_tree_ordenado = df_marceno_tree.sort_values(['idTre

#Calculo de la fecha anterior a la observación por arbol
df_marceno_tree_ordenado['prev_date'] = df_marceno_tree_ordena

# variable para promedios del clima
promedios_clima = []

# Se agrupa el clima de manera diaria (Se elige esta resolució
raw_climate_daily = raw_climate.groupby('timestamp').mean(nume
raw_climate_daily['date'] = pd.to_datetime(raw_climate['timest

# Iterar por cada fila de pest_df_sorted
for _, row in tqdm(df_marceno_tree_ordenado.iterrows(), total=
    id_arbol = row['idTree']
    fecha_actual = row['dateObservation']
    fecha_anterior = row['prev_date']

    if pd.isna(fecha_anterior):
        # Si no hay fecha anterior, usar un periodo fijo anter
        fecha_anterior = fecha_actual - pd.Timedelta(days=7)
```

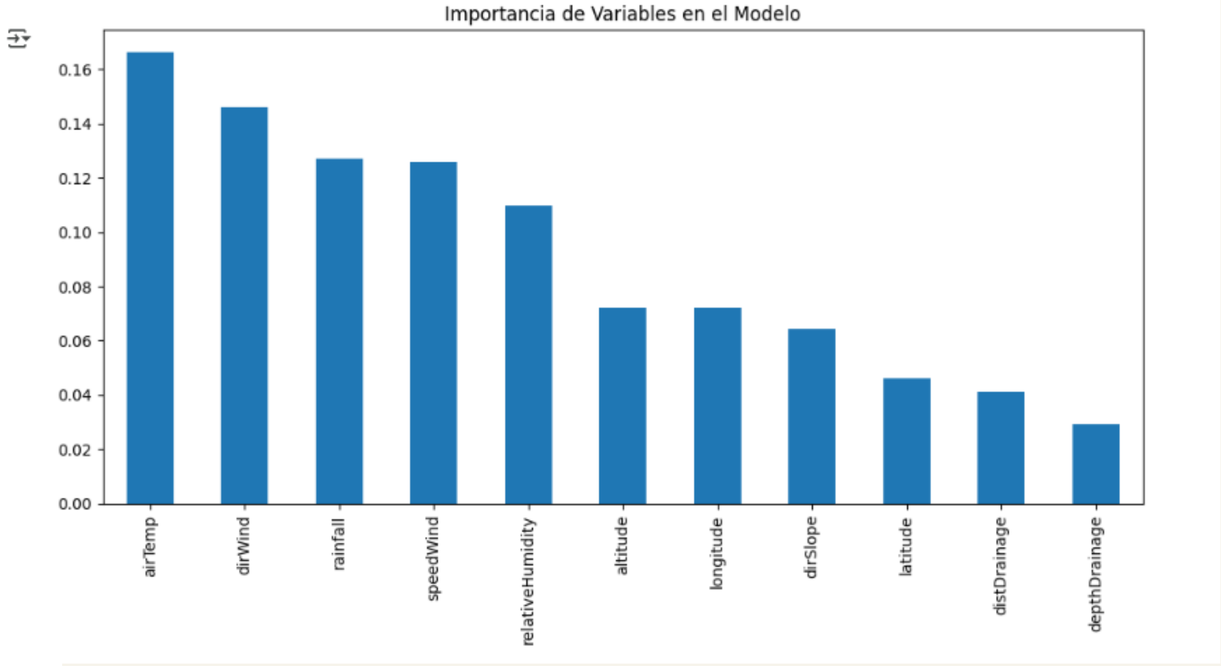
```
    # Filtrar el clima entre fechas (exclusiva-inclusiva)
    mask = (raw_climate_daily['date'] > fecha_anterior) & (raw_climate_daily['date'] <= fecha_actual)
    clima_intervalo = raw_climate_daily.loc[mask]

    # Calcular promedios
    promedios = clima_intervalo.mean(numeric_only=True)

    # Añadir idTree y dateObservation para unir luego
    promedios['idTree'] = id_arbol
    promedios['dateObservation'] = fecha_actual

    promedios_clima.append(promedios)

# Se construye un dataframe de los promedios de clima para unirlo al dataframe de arboles
df_promedios_clima = pd.DataFrame(promedios_clima)

# Se une los dataframes
raw_climate_tree_pest = pd.merge(df_marceno_tree_ordenado,df_promedios_clima,on=['idTree','dateObservation'

print(_tree_pest.describe())

```

```
100%|██████████| 300/300 [00:00<00:00, 350.90it/s]
       dateObservation  presenceMarceno         datePlanting    latitude  \
count              300       300.000000                  300  300.000000
mean   2023-01-28 16:48:00         0.646667  2019-01-18 12:48:00    6.209149
min    2022-04-01 00:00:00         0.000000  2018-12-01 00:00:00    6.208690
25%    2022-10-01 00:00:00         0.000000  2018-12-01 00:00:00    6.208830
50%    2023-01-24 00:00:00         1.000000  2018-12-01 00:00:00    6.209040
75%    2023-05-30 00:00:00         1.000000  2018-12-01 00:00:00    6.209420
```

RESULTADOS

Correlación entre el Clima y las Plagas





Factores que impactan la presencia de la peste Marceño:

- La temperatura ambiente es la variable con mayor impacto en la predicción de la peste.
- Aunque comúnmente se destaca la humedad relativa, en este análisis no es la más influyente.
- La humedad relativa depende del microclima del cultivo.
- La lluvia tiene un impacto más alto y está moderadamente correlacionada con la humedad relativa.

RESULTADOS

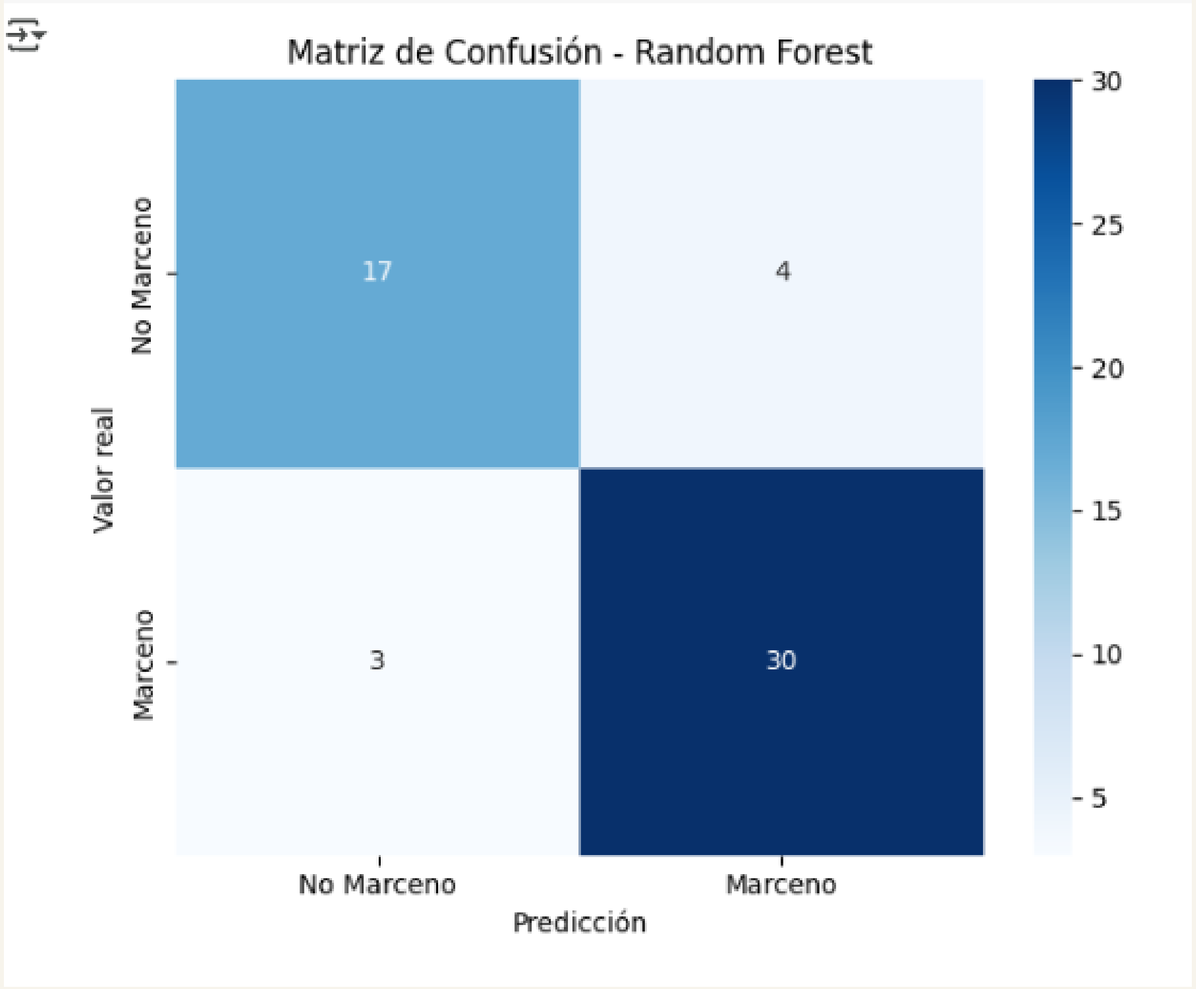
Evaluación de Algoritmos

Random Forest

Evaluación del modelo

```
[ ]  1 y_pred = modelo.predict(X_test)
     2
     3 # Reporte de métricas
     4 print("=== CLASIFICATION REPORT ===")
     5 print(classification_report(y_test, y_pred, digits=3))
     6
     7 # Matriz de confusión
     8 cm = confusion_matrix(y_test, y_pred)
```

```
=== CLASIFICATION REPORT ===
              precision    recall  f1-score   support

           0      0.850     0.810     0.829        21
           1      0.882     0.909     0.896        33

    accuracy                          0.870        54
   macro avg      0.866     0.859     0.862        54
weighted avg      0.870     0.870     0.870        54
```



RESULTADOS

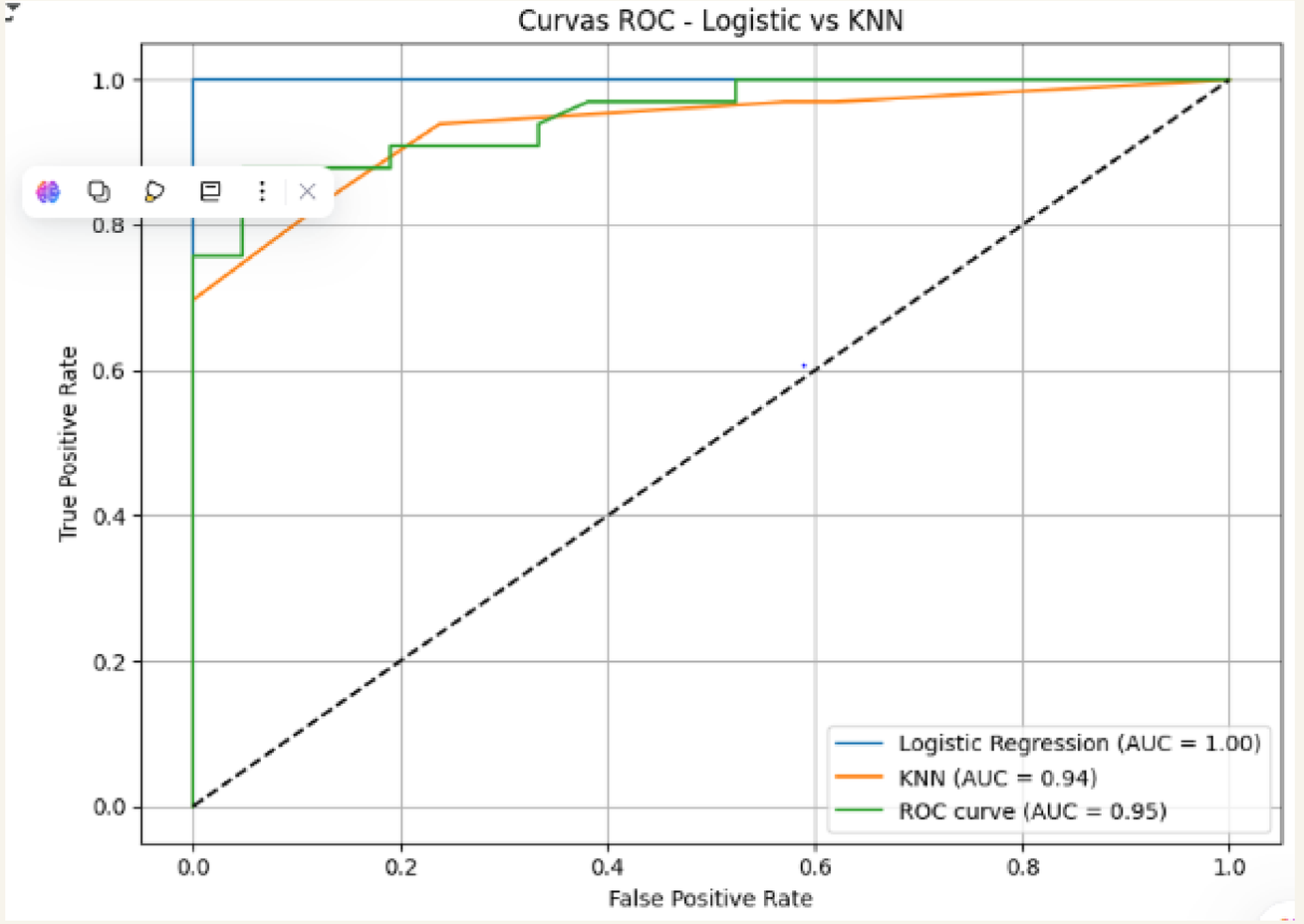
Regresión Logística – KNN

```
=== Logistic Regression ===
Accuracy: 1.000
Recall: 1.000
F1-score: 1.000
AUC-ROC: 1.000

=== K-Nearest Neighbors ===
Accuracy: 0.852
Recall: 0.848
F1-score: 0.875
AUC-ROC: 0.935
```

Validación de overfitting para Logistic Regresion

```
[ ]
1 from sklearn.model_selection import cross_val_score, StratifiedKFold
2
3 cv = StratifiedKFold(n_splits=6,shuffle=True,random_state=42)
4
5 auc_scores = cross_val_score(logreg,X_pca,y,cv=cv,scoring='roc_auc')
6
7 #AUC
8 print("AUC scores:", auc_scores)
9 print("AUC promedio: {:.3f} ± {:.3f}".format(np.mean(auc_scores), np.std(auc_scor
10
11 # F1-score
12 f1_scores = cross_val_score(logreg, X_pca, y, cv=cv, scoring='f1')
13 print("\nF1 scores:", f1_scores)
14 print("F1 promedio: {:.3f} ± {:.3f}".format(np.mean(f1_scores), np.std(f1_scores)
```

```
AUC scores: [0.72839506 0.75720165 0.84033613 0.86554622 0.78991597 0.84453782]
AUC promedio: 0.804 ± 0.050

F1 scores: [0.77777778 0.75       0.78571429 0.79310345 0.81967213 0.86206897]
F1 promedio: 0.798 ± 0.035
```



CONCLUSIONES

En el análisis de correlación, la velocidad del viento se identifica como una variable climática relevante que puede influir en el diseño agroforestal del cultivo. Esta información resulta útil para la implementación de estrategias como la instalación de barreras naturales o árboles en los bordes del cultivo, que actúan como protección frente a condiciones adversas asociadas al viento y propragacion de la plaga.

De los algoritmos evaluados el modelo Random Forest fue el que logro un mejor equilibrio, una alta precision y una alternativa robusta para la detección de la plaga, comparado con los demas modelos que uno de ellos presento sobre ajuste, esto se valida y sustenta con la literaratura mencionada en apartados anteioriores.

Estos enfoques permiten desarrollar sistemas predictivos más eficientes y sostenibles en el manejo agrícola, estas tecnologías pueden reducir la dependencia de pesticidas, minimizando así los impactos ambientales, los costos de producción que facilitan decisiones agrícolas más sostenibles y eficientes.