Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores

Unidad Morelia

Proyecto Final

Predicción del Crecimiento Significativo en Plantas

PRESENTA:

Alexis Uriel Aguilar Uribe

PROFESORES:

Dra. Marisol Flores Garrido

Dr. Luis Miguel García Velázquez

GRADO

Licenciatura en Tecnologías para la Información en Ciencias

Número de Cuenta: 424060075

Asignatura: Sistemas basados en conocimiento [Machine Learning]

A: 27 de Mayo del 2025

Índice

1. Introducción

En la agricultura, como cualquier otra industria, se vuelve relevante la optimización de los recursos y ganancias, es decir, reducir los insumos consumidos mientras se incrementa la producción (tanto en calidad como en cantidad); todo lo anterior se traduce en aplicar mejoras en diferentes áreas y aspectos que convergen y se relacionan para generar ganancias y reducir costos en la agricultura. Para el caso de este proyecto, el interés se encuentra en el crecimiento de las plantas, bajo qué factores ambientales y de cuidado propician un crecimiento significativo en las plantas.

Para lograr el último punto, se tiene como objetivo el crear un modelo de aprendizaje supervisado para la clasificación del crecimiento significativo en base a los factores y mediciones relacionadas a su cuidado y ambiente. Esto se encuentra desarrollado en el repositorio en GitHub dedicado para el proyecto: Plant Growth Model.

2. Descripción de los Datos

El conjunto de datos que se emplearán para el proyecto se encuentra disponibles en [1], que es un conjunto de datos publicados en Kaggle por la propia comunidad. Se cuenta con siete columnas, donde seis de ellas son atributos y la otra el target, referenciando a la fuente del conjunto de datos, se tienen los siguientes atributos junto con su descripción y tipo de dato:

- **Soil_Type** [*String*]: El tipo o composición del suelo en el que las plantas están creciendo o se plantan.
- **Sunlight_Hours** [*Float*]: La duración o intensidad de la luz solar que las plantas reciben.
- **Water_Frequency** [*String*]: Qué tan seguido se riegan las plantas, se indica la frecuencia del riego.
- **Fertilizer_Type** [*String*]: El tipo de fertilizante usado para nutrir a las plantas.
- **Temperature** [*Float*]: Las condiciones de la temperatura ambiental bajo las cuales las plantas están creciendo.
- **Humidity** [*Float*]: El nivel de humedad en el ambiente alrededor de las plantas.
- **Growth_Milestone** [*Integer, Target*]: Descripción o marcadores que indican la etapa o eventos significativos en el proceso de crecimiento de las plantas.

Por último, el conjunto de datos consta de 193 instancias (filas), las diferentes instancias lucen de la siguiente manera:

Soil_Type	Sunlight_Hours	Water_Frequency	Fertilizer_Type
sandy	9.228	daily	none
sandy	9.774	weekly	chemical
clay	7.392	bi-weekly	none
clay	6.462	bi-weekly	organic
clay	8.846	weekly	organic
loam	5.985	bi-weekly	chemical

Temperature	Humidity	Growth_Milestone
33.804	32.815	0
32.549	61.377	1
31.100	68.600	0
27.517	34.175	1
27.700	56.800	1
29.757	57.476	0

3. Análisis Exploratorio de Datos

Los tipos de datos en base a la descripción proporcionada en [1] con la mostrada al momento de la lectura de los datos en Python, por lo que no es necesario realizar una transformación sobre los tipos de datos de cada atributo.

Debido a que los valores que toma y representan el target, *Growth_Milestone*, son enteros, se tiene que el modelo que se generará será un clasificador binario; cuyas clases representan si hay un crecimiento significativo, bajo ciertos criterios, en las plantas. Como primera observación se tiene que el conjunto de datos está balanceado respecto a las clases, por lo que se podría usar cualquiera de las métricas bajo una justificación válida o apropiada al problema:

Growth_Milestone	
Clases	Conteo
0 [No Milestone]	97
1 [Milestone]	96

Se presenta un análisis univariado sobre los atributos numéricos y categóricos, y por último se prueban algunas hipótesis relevantes y relacionadas sobre las observaciones en los apartados anteriores.

3.1. Atributos Numéricos

Generando la descriptiva básica (medidas centrales y de dispersión) de los datos se obtienen los siguientes resultados:

Medida	Sunlight_Hours	Temperature	Humidity
Media	6.8264	25.0760	58.0989
Desviación Estándar	1.5995	5.3541	12.6317
Mínimo	4.0331	15.2000	30.5676
Q_1	5.4770	20.6370	49.3000
Q_2	6.8332	25.9123	59.1828
Q_3	8.2411	29.7579	69.1000
Máximo	9.9139	34.8101	79.6482

Primero se destaca que siguen diferentes rangos de valores, por lo que se tendrán que estandarizar o blanquear para su adecuado uso para el entrenamiento de los modelos que se crearán, además de permitir realizar comparativas entre las distribuciones. Al estandarizar los valores se tienen los siguientes resultados:

Medida	Sunlight_Hours	Temperature	Humidity
Media	0	0	0
Desviación Estándar	1	1	1
Mínimo	-1.746	-1.8445	-2.1795
Q_1	-0.843	-0.8290	-0.6965
Q_2	0.004	0.1561	0.0858
Q_3	0.884	0.8744	0.8709
Máximo	1.930	1.8180	1.7059

Destacándose que el atributo *Sunlight_Hours* figura que sigue una distribución debido a que su Q_2 se aproxima a 0 junto que sus Q_1 y Q_3 se parecen, salvo el signo. Mientras que en *Humidity* su Q_2 se encuentra equidistante a Q_1 y a Q_3, lo que significa que no tiene sesgo más no es simétrica. Y en *Temperature*, por sus cuartiles, tiene un sesgo negativo considerable. En base a lo mencionado se tiene que los tres atributos siguen diferentes distribuciones, implicando que estos atributos surgen de diferentes fenómenos, interacciones y procesos que impactan en el crecimiento de las plantas, por lo que los tres atributos se vuelven relevantes para la clasificación debido a que cada uno condensa diferentes procesos para lograr un crecimiento significativo.

De los box plots, se puede observar que las distribuciones de los atributos no son tan diferentes según el target, según si hay un crecimiento significativo, esto al ver que las cajas están superpuestas, haciendo que no exista una diferencia Significativa entre las medias de las distribuciones. Lo que se destaca es que cuando existe un crecimiento significativo tiende a tomar valores más bajos y además de que su desviación estándar también tiende a decrecer.

Este último hecho se podría relacionar con que existe un control sobre los factores ambientales, haciendo que las distribuciones sean más restrictivas y reguladas, dejando afuera posibles fenómenos que generen un ambiente irregular para el crecimiento de las propias plantas. Haciendo que la planta esté en un ambiente ideal para su crecimiento pero, por las distribuciones,

existe la posibilidad de que aunque esté en las condiciones idóneas no logre un crecimiento significativo.

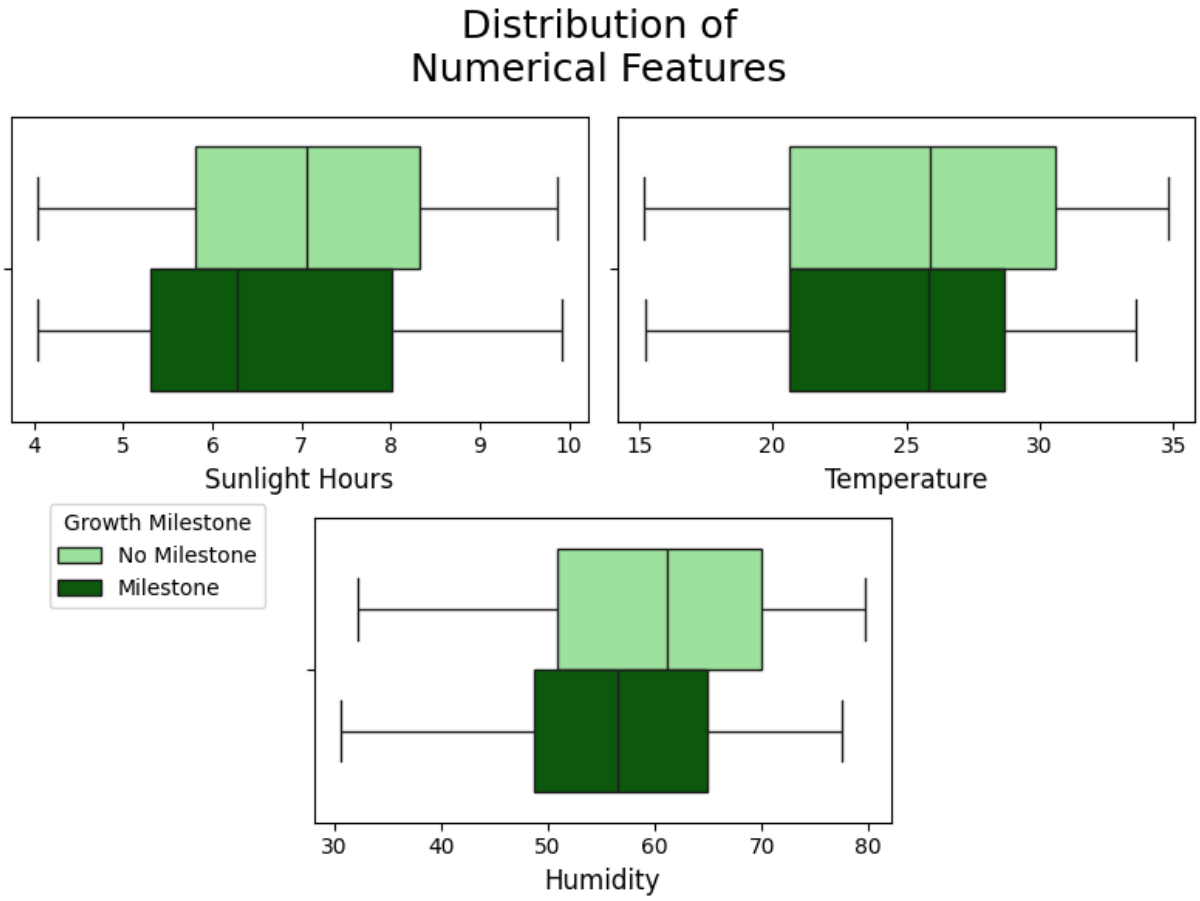

Figura 1: Distribución de los atributos Numéricos según el Crecimiento Significativo

De manera visual, no se cuenta con valores atípicos derivados de la Regla del Rango Intercuartil ni valores faltantes, por lo que estos atributos numéricos se encuentran preparados para la fase de entrenamiento.

3.2. Atributos Categóricos

Cada atributo categórico tiene tres valores únicos, con las siguientes cantidades y valores:

Soil_Type	Cantidad
clay	67
sandy	64
loam	62

Water_Frequency	Cantidad
daily	74
bi-weekly	60
weekly	59

Fertilizer_Type	Cantidad
none	74
chemical	65
organic	54

Siendo *Soil_type* el atributo que está más balanceado de los tres, por lo que el modelo tendrá suficientes instancias por cada valor como para distinguir entre los posibles casos donde este atributo sea crítico. Mientras que los otros atributos, *Water_Frequency* y *Fertilizer_Type* no están balanceadas en sus valores, esto podría estar relacionado a que no se realizaron los suficientes experimentos para generar todos los posibles resultados por igual o existe un factor que provocara un cierto sesgo o preferencia sobre ciertos valores, como lo son la falta de insumos suficientes.

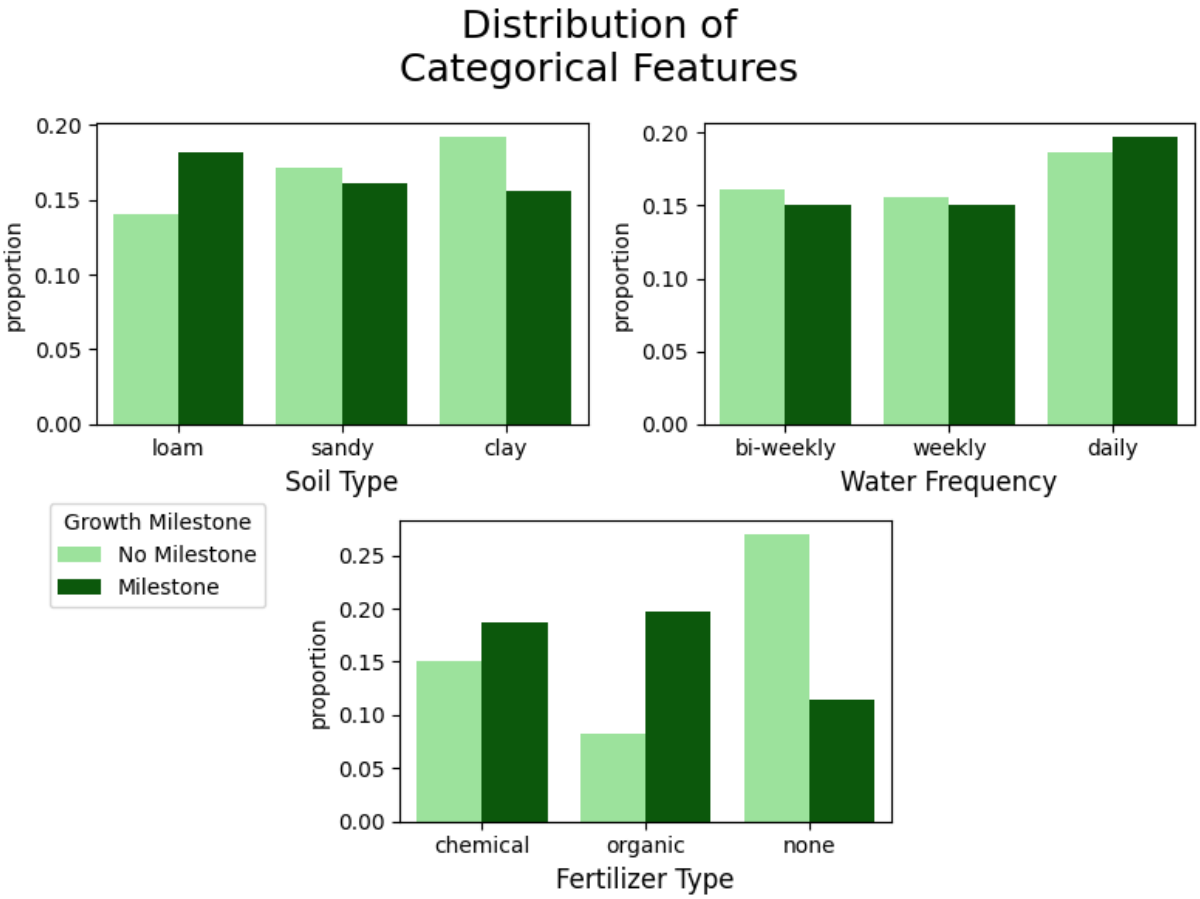

Figura 2: Distribución de los atributos Categóricos según el Crecimiento Significativo

Del gráfico, se puede destacar dos hechos. El atributo de *Soil_type* tiene un impacto significativo en el crecimiento de las plantas, debido a que los suelos de tipo franco (loam) afecta positi-

vamente este fenómeno, mientras que un suelo de tipo arcilla (clay) tiene el efecto negativo, esto es un indicio de una posible interacción fuerte que repercute en el crecimiento significativo de una planta según el suelo en el que está plantada. Mientras que al atributo *Fertilizer_Type* muestra un también un impacto en el crecimiento de las plantas, y este hecho es natural debido a que de aquí, la planta, obtiene los diferentes nutrientes para su adecuado crecimiento, siendo así el como favorece al crecimiento de una planta según el fertilizante que se use.

4. Metodología del Proyecto

4.1. Preprocesamiento

Del análisis realizado en la sección previa, en 3 Análisis Exploratorio de Datos, se puede determinar que es necesario aplicar un adecuado preprocesamiento de los atributos, así cómo también derivar otros que puedan crear otras posibles interacciones. Por lo que se propone realizar lo siguiente:

- **Atributos Numéricos**: Los atributos que se generan los que se obtienen después de aplicar las funciones $\sqrt{x}$, $\log_{10} x$, $1/x$, x^2 y x (denotando la función identidad), donde x son los posibles valores que pueden tomar los atributos numéricos. Después de generar los valores, se la aplica la transformación de estandarización en cada uno de los atributos derivados, esto se hace para que los valores estén en un rango de entre -3 y 3 o, equivalentemente, estén en unidades estándar.
- **Atributos Categóricos**: Debido a que los atributos pueden tomar tres valores que no tienen una relación de orden plausible o clara, se tiene que realizar una codificación haciendo uso de la codificación One-Hot, para obtener una información más granular sobre las diferentes instancias. Después de esta codificación no se le aplica ninguna otra transformación.

Debido a que el espacio de atributos creció a una dimensión relativamente grande, se tiene que como parte del preprocesamiento se realiza una selección de atributos con el fin de reducir el problema de la dimensionalidad y hacer que los modelos tengan un mejor aprendizaje (capturen las relaciones importantes) mientras se reduce el coste computacional tanto de entrenamiento como de predicción. Para ello:

- **Atributos Numéricos**: Se selecciona los mejores 10 atributos por medio de Mutual Information (MI) [2].
- **Atributos Categóricos**: Se seleccionan los mejores 6 atributos por medio del Estadístico χ^2 [3].

Ambas métricas permiten medir la misma noción de independencia respecto al target (*Growth_Milestone*), es decir, se conversan o se usan los atributos que son menos independientes del target con el

fin de que el modelo pueda capturar y aprender estas relaciones.

Como parte del preprocesamiento, se incluye la separación del conjunto de datos original en conjuntos de entrenamiento y de prueba, se hace uso del 25 % de las instancias como instancias del conjunto de prueba.

4.2. Modelos de Machine Learning

Para realizar la tarea de clasificación del crecimiento de las plantas se hace uso, principalmente, de modelos que permiten generar o propiciar bordes de decisión no lineales, debido a que, por como se puede observar en 3 Análisis Exploratorio de Datos, las clases del target no son linealmente separables, haciendo que los modelos que hacen uso de bordes de decisión naive o lineales no puedan capturar adecuadamente los patrones en los datos para clasificar una instancia de manera adecuada. Por ello, se hace uso de los siguientes modelos e hiperparámetros a ajustar:

Support Vector Machine (SVM)

El usar SVM permite añadir un capa adicional de interacción entre atributos debido a que hace uso del truco del kernel que esto permite propiciar un mejor ambiente para la separabilidad entre clases. Esto último hace que SVM sea un modelo predilecto para tratar problemas de clasificación con clases sin separabilidad lineal, como lo es en este caso. De los hiperparámetros que se encuentra en [4], se ajustan los siguientes:

- *kernel*: Se escoge entre Poly (polinómico), RBF (Radial Basis Function) y Sigmoid.
- C: Parámetro de regularización. Se escogen valores reales en $[1e-10, 5]$
- *gamma*: Coeficiente del kernel. Se escogen valores reales en $[0, 2]$
- *degree*: Grado del polinomio. Ajustado únicamente en el kernel poly. Se escogen valores enteros en $[1, 4]$
- *coef0*: Valor del coeficiente x^0. Ajustado únicamente en el kernel poly. Se escogen valores reales en $[0, 2]$

Random Forest

El usar Random Forest permite aprovechar de mejor manera las decisiones tomadas por árboles

de decisión simples, de profundidades bajas, gracias a que cuando se combinan sus decisiones se reduce el sesgo y, por su entrenamiento, también se reduce su varianza. Además de ello, al combinar varios árboles simples permite el crear bordes de decisión más complejas, no lineales, beneficiándose así la separabilidad entre clases. De los hiperparámetros que se encuentran en [5], se ajustan los siguientes:

- *n_estimators*: Cantidad de estimadores. Se escogen valores enteros en $[1, 100]$
- *criterion*: Función para decidir que nodo aplicar el split. Se escoge entre gini y entropy
- *max_depth*: Altura de máxima de los árboles de decisión. Se escogen valores enteros pequeños en $[1, 3]$
- *min_samples_split*: Mínimo número de instancias en un nodo para aplicar la operación de split. Se escogen valores reales en $[1e-2, 0{,}5]$

AdaBoost

El usar árboles de decisión entrenados de forma secuencial para corregir los errores generados por modelos previos mientras se mejora el poder predictivo, permite generar bordes de decisión no lineales y más finos, es decir, que se adecuen para generar una mayor separabilidad entre clases sin que se caiga en un problema de varianza (esto justamente lo permite el ratio de aprendizaje). De los hiperparámetros que se encuentran en [6], se ajustan los siguientes:

- *estimator*: Estimador débil. Se hace uso de árboles de decisión con los parámetros defaults de [7] y salvo *max_depth* que se escoge valores enteros en $[1, 4]$
- *n_estimators*: Cantidad de estimadores. Se escogen valores enteros en $[1, 100]$
- *learning_rate*: Peso que se aplica a cada modelo de manera secuencial. Se escogen valores reales en $[1e-3, 2]$

Logistic Regression

El decidir la clase de una instancia por medio de un borde decisión lineal podría favorece en la comparativa contra modelos que generan bordes de decisión más complejos, además de se explorar otra forma de generar interacciones por medio de combinaciones lineales de atributos. Esto último permite que este modelo sea ampliamente usado en problemas donde las clases están linealmente separadas. De los hiperparámetros que se encuentran en [8], se ajustan los siguientes:

- C: Parámetro de regularización. Se escogen valores reales en $[1e-10, 5]$
- *penalty*: Tipo de penalización que se aplica a los parámetros. Se escoge entre $l1$ y $l2$
- *solver*: Algoritmo usado para optimizar los parámetros. Se escoge entre liblinear y saga (que admiten ambas penalizaciones)

K-Nearest Neighbors (KNN)

El usar la distancia o similitud entre instancias permite determinar bordes de decisión no lineales entre las clases, debido a que usa operaciones no lineales para medir esta noción de cercanía entre los diferentes puntos, permitiendo así la generación de la separación entre clases. Esto último hace a KNN un modelo para modelar la predicción de forma sencilla y con bajo coste computacional. De los hiperparámetros en [9], se ajustan los siguientes:

- *n_neighbors*: Cantidad de vecinos cercanos que se consideran. Se escogen valores enteros en $[3, 50]$
- *weights*: La función de peso que se le aplica a cada vecino, importancia que tiene. Se escoge entre uniform y distance
- p: Potencia de métrica de Minkowski (por default, y solo se ajusta para esta métrica). Se escogen valores enteros en $[1, 3]$
- *metric*: Métrica que se usa para medir distancia y similitud. Se escoge entre cosine y correlation

5. Experimentos y Discusión de Resultados

5.1. Optimización de Hiperparámetros

Para realizar el fine-tunning de los hiperparámetros en cada uno de los modelos propuestos en 4 Metodología del Proyecto, se hace uso Scikit Optimize, en específico de BayesSearchCV [10] para determinar los mejores hiperparámetros por medio de métodos bayesianos (véase que funciona como Optuna [11] pero sin usar métodos sofisticados como algoritmos genéticos).

5.2. Descripción de Experimentos

Para el fine-tunning de hiperparámetros se hizo uso de la métrica *accuracy*, debido a que como las instancias del conjunto de datos se encuentran balanceadas 3 Análisis Exploratorio de Datos y, bajo este escenario de balance, la métrica *accuracy* se comporta como *f1*, y así permitiendo reducir el impacto de los falsos positivos (FP) y falsos negativos (FN) en las predicciones, que esto se vuelve equivalentemente a reducir la sobrestimación y subestimación, respectivamente, del crecimiento Significativo que una planta alcanza. Y como BayesSearchCV [10] se apoya en cross-validation, también se tiene que definir la cantidad de pliegues o folds para realizar la validación de los hiperparámetros candidatos a ser los mejores para un modelo, por ello se hace uso de 6 pliegues debido a que deja suficientes instancias para validar los hiperparámetros como de entrenamiento.

5.3. Evaluación de Resultados

En base a los mejores modelos (hiperparámetros) encontrados por medio BayesSearchCV [10], se evalúan tanto en el conjunto de entrenamiento como de prueba, se obtienen los siguientes resultados tanto tabulados como gráficados:

	SVM	Forest	AdaBoost	Logistic	KNN
Accuracy	0.638889	0.708333	0.694444	0.652778	1.000000
Precision	0.619565	0.666667	0.659574	0.657895	1.000000
Recall	0.770270	0.864865	0.837838	0.675676	1.000000
F1	0.686747	0.752941	0.738095	0.666667	1.000000

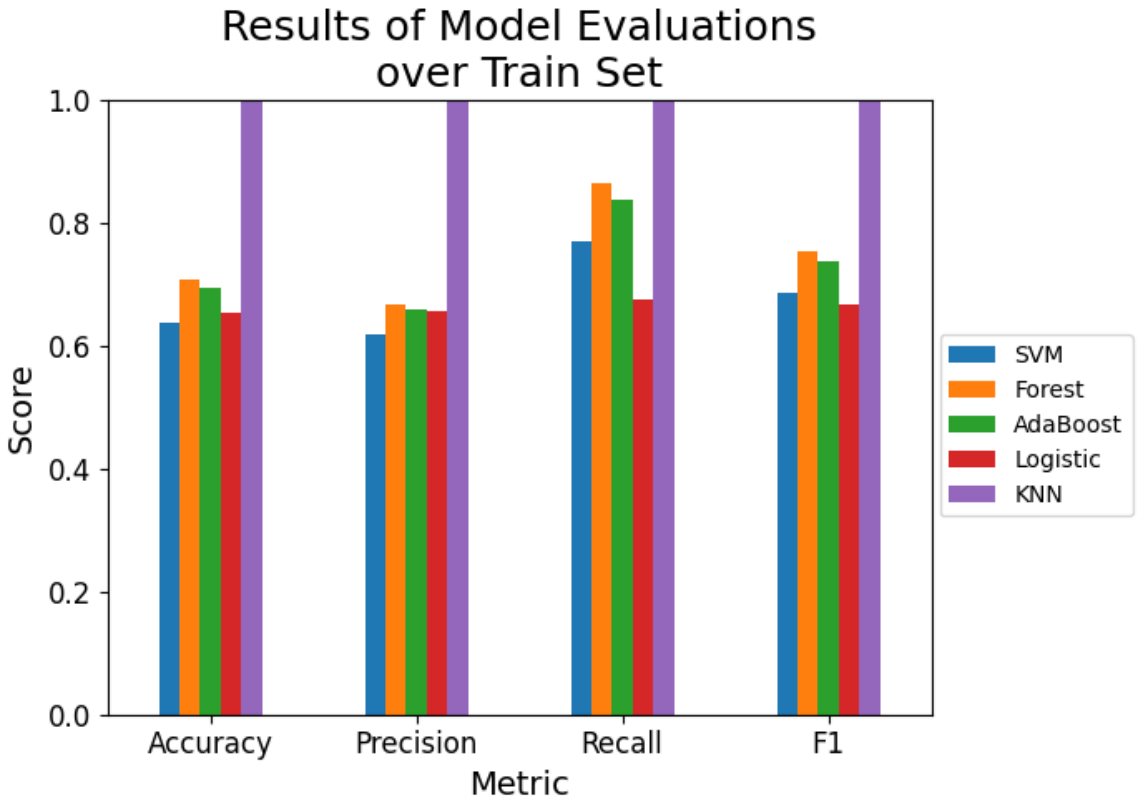

Figura 3: Puntajes de diferentes métricas de los Mejores Modelos en el Conjunto de Entrenamiento

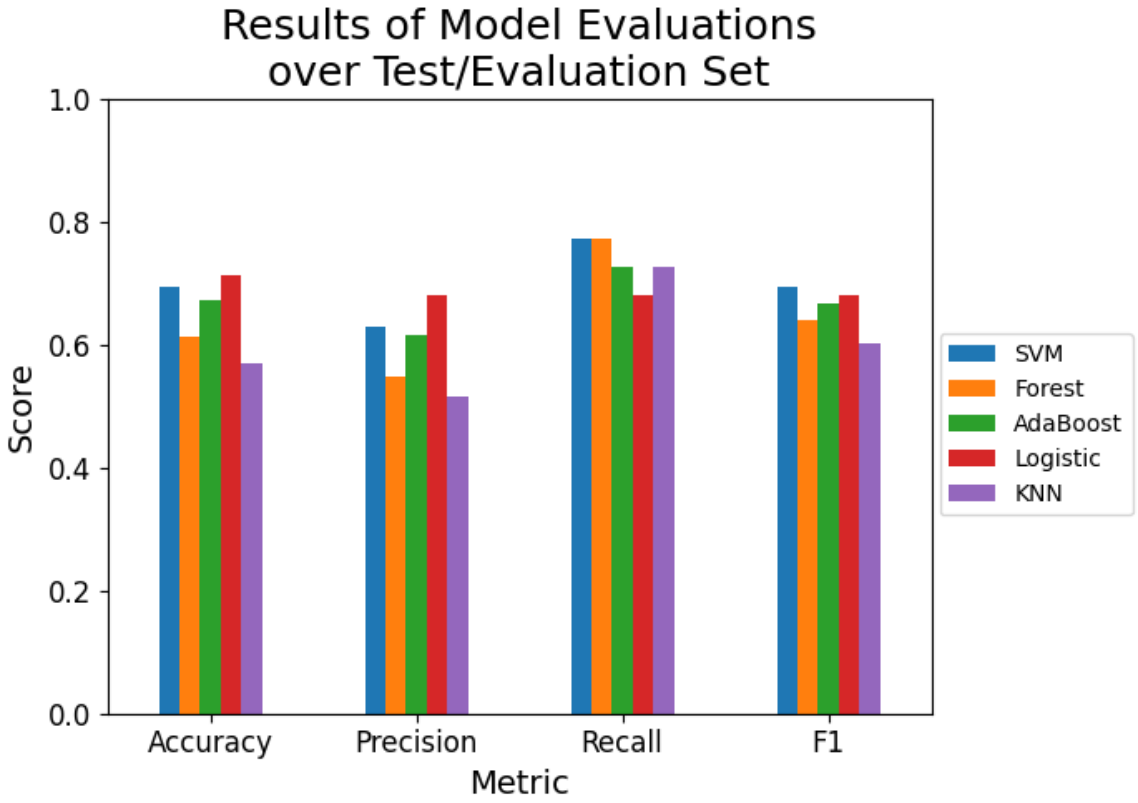

Figura 4: Puntajes de diferentes métricas de los Mejores Modelos en el Conjunto de Prueba

	SVM	Forest	AdaBoost	Logistic	KNN
Accuracy	0.693878	0.612245	0.673469	0.714286	0.571429
Precision	0.629630	0.548387	0.615385	0.681818	0.516129
Recall	0.772727	0.772727	0.727273	0.681818	0.727273
F1	0.693878	0.641509	0.666667	0.681818	0.603774

Haciendo de las principales métricas empleadas en clasificación se puede observar el como se comportan los modelos en ambos conjuntos. Destacando como en Random Forest, AdaBoost y KNN alcanzan puntajes más bajos en el conjunto de prueba que en el conjunto de entrenamiento, destacando como KNN se encuentra sobreajustado de manera extrema.

En cambio, en los modelos como SVM y Logistic Regression, tienen un mejor comportamiento en el conjunto de prueba, haciendo que estos modelos sean lo más adecuados para el problema presente. Debido a que ambos modelos emplean la separabilidad de clases como principio de su funcionamiento, por lo que se tiene que los atributos generados en la etapa de preprocesamiento fueron fructíferos para que el modelo sea capaz de aprender los patrones para poder distinguir entre ambas clases del target.

Para el caso de SVM, tiene un kernel polinómico de grado 1 como sus mejores hiperparámetros; y para el caso de Logistic Regression, su tipo de penalización es $l1$ es su mejor hiperparámetro. Estas configuraciones muestran un indicio fuerte de que las clases son linealmente separables y, de aquí, estos modelos obtengan los mejores resultados y que puedan ser robustos para la predicción en nuevas instancias.

6. Análisis de los Resultados

7. Conclusiones

Referencias Bibliográficas

[1] gorororororo23, *Plant Growth Data Classification*, https://www.kaggle.com/datasets/gorororororo23/plant-growth-data-classification/data, 2024.

[2] scikit-learn developers, *mutual_info_classif*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html.

[3] scikit-learn developers, *chi2*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2.

[4] scikit-learn developers, *SVC*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.

[5] scikit-learn developers, *RandomForestClassifier*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.

[6] scikit-learn developers, *AdaBoostClassifier*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier.

[7] scikit-learn developers, *DecisionTreeClassifier*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[8] scikit-learn developers, *LogisticRegression*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.

[9] scikit-learn developers, *KNeighborsClassifier*, 2025. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[10] scikit-optimize contributors, *BayesSearchCV*, 2025. dirección: https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html#skopt.BayesSearchCV.

[11] T. Akiba, S. Sano, T. Yanase, T. Ohta y M. Koyama, «Optuna: A next-generation hyperparameter optimization framework,» en *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, págs. 2623-2631. DOI: 10.1145/3292500.3330701.