Curso: Técnicas de Aprendizaje Automático – Machine Learning

Tema 4: Arboles y Bosques Aleatorios

Sonia I. Mariño, Rafael Perez, Leonardo Gomez Chavez

FaCENA - UNNE - 2023

Tipología de Arboles de Decisión

Algoritmos basados en árboles para el aprendizaje automático

- Árboles de Decisiones (Decision Trees)
- Bosques Aleatorios (Random Forest)
- Aumento de Gradiente (Gradient Boosting)
- Bagging (Bootstrap Aggregation),
    - referencia al empleo del muestreo repetido con reposición o bootstrapping, para reducir la varianza de algunos modelos de aprendizaje estadístico, entre ellos los basados en árboles.

Definiciones

Árbol de Decisión (AD) - Algoritmo cuya finalidad es reconocer la existencia de relaciones en un determinado conjunto de datos por medio de procesos que imitan el funcionamiento del cerebro humano [1].

Random Forest (Breiman, 2001). Técnica de aprendizaje supervisado que genera múltiples árboles de decisión sobre un conjunto de datos de entrenamiento. Los resultados obtenidos se combinan a fin de obtener un modelo único más robusto en comparación con los resultados de cada árbol por separado (Lizares, 2017). Cada árbol se obtiene mediante un proceso de dos etapas [2]

[1] mencionado en Angélica Villón L., 2021 Aplicación de técnicas de minería de datos para predecir el desempeño académico de los estudiantes de la escuela, https://incyt.upse.edu.ec/ciencia/revistas/index.php/rctu/article/view/637/530

[2] mencionado en Espinosa-Zúñiga, Javier Jesús. Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. Ingeniería, investigación y tecnología, 21(3), 00002. Epub 02 de diciembre de 2020.https://doi.org/10.22201/fi.25940732e.2020.21.3.022

Bosques Aleatorios (Random Forest)

- BA en ML e IA, introducción
- Técnica de AA / ML supervisado
- Mejora resultados al experimentar con varios AD
- Aplicable en problemas de:
  - Clasificación
    - Variable dependiente es categórica
    - El valor en el nodo terminal, asume la *moda* de las observaciones del conjunto de entrenamiento que corresponde en esa región
  - Regresión
    - Variable dependiente es continua
    - El valor en el nodo terminal, asume la *media* de las observaciones en esa región

Tipología de Arboles de Decisión

Algoritmos basados en árboles para el aprendizaje automático

- Árboles de Decisiones (Decision Trees)
- Bosques Aleatorios (Random Forest)
- Aumento de Gradiente (Gradient Boosting)
- Bagging (Agregación Bootstrap "Bootstrap Aggregation")

Métodos de Ensamble

Evolución de los AD

Modelos de aprendizaje supervisado

Métodos de Ensamble

- Su objetivo es combinar las predicciones de varios estimadores base construidos con un algoritmo de aprendizaje dado, para mejorar la generalización / robustez sobre un solo estimador.

Métodos de Ensamble

Generalmente, se distinguen dos familias de métodos de ensamble:

- Métodos de promedio
  - Se construyen varios estimadores de forma independiente y se promedian sus predicciones. En promedio, el estimador combinado suele ser mejor que cualquiera de los estimadores de base única porque su varianza se reduce.
  - Ejemplos: Bagging methods (métodos de embolsado), Forests of randomized trees (bosques de árboles aleatorios)

Métodos de impulso o Boosting

- Los estimadores base se construyen secuencialmente, y se trata de reducir el sesgo del estimador combinado. La motivación es combinar varios modelos débiles para producir un mejor conjunto
- Ejemplos: AdaBoost, Gradient Tree Boosting,

Algoritmo de Bosque aleatorio o BA - RF

- Algoritmo de aprendizaje supervisado basados en AD.
- Los BA surgen ante un problema de los AD, el sobre ajuste / overfiting. Es decir,
  - un *bias* bajo y alta varianza
  - BA puede mitigar el sobre ajuste, promedia los resultados de predicción de los AD. Brinda mayor precisión predictiva que un AD.
- BA introduce dos variantes en los AD: aleatoriedad y agregación
  - Se entrenan varios AD, cada uno con un distinto sub-set de datos –***aleatorio-*** del conjunto original. [sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.]
  - El BA entrenado, permite realizar la predicción para un dato nuevo, que implica ***agregar*** los resultados de cada AD. Es decir, la predicción final resulta de promediar las predicciones de cada AD individual. En la votación para cada resultado previsto, se opta por:
    - la moda, en problema de clasificación,
    - la media, en problema de regresión
- Algoritmo flexible y fácil de usar.

Algoritmo. Aplicaciones

BA aplica en problemas de clasificación y regresión

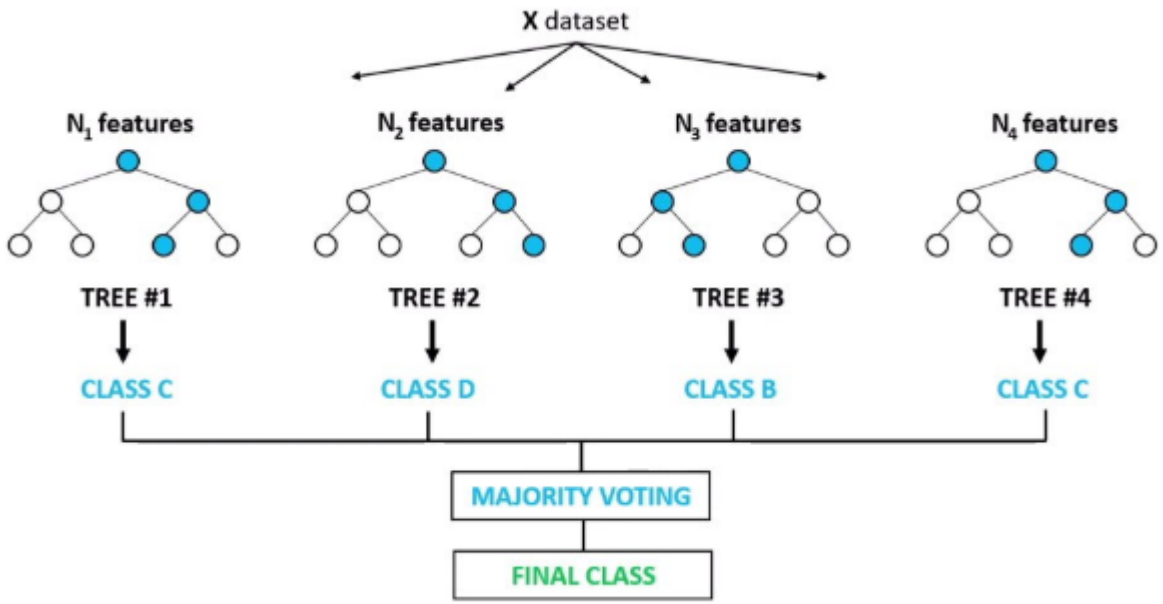
BA, proponer soluciones ante distintas problemáticas:

- localizar características importantes en un conjunto de datos. Se aplica el algoritmo de Boruta.
- proporcionar recomendaciones de diferentes productos a los clientes en el comercio electrónico.
- identificación / determinación de:
    - enfermedades analizando el historial médico.
    - tipología de cliente, fraudulento o legítimo.

Algoritmo

- Paso 1: selección de muestras en forma aleatoria de la base de datos proporcionada.
    - Puede incluirse mayor aleatoriedad si se eligen las variables aleatorias.
- Paso 2: creación de un AD para cada muestra seleccionada. Se obtiene un resultado de predicción de cada árbol creado.
- Paso 3: votación para cada resultado previsto. En problema de *clasificación:* moda, y en problema de *regresión:* media.
- Paso 4: selección del resultado de predicción más votado como predicción final.

Algoritmo


X dataset
N1 features
N2 features
N3 features
N4 features
TREE #1
TREE #2
TREE #3
TREE #4
CLASS C
CLASS D
CLASS B
CLASS C
MAJORITY VOTING
FINAL CLASS

Phyton. BA

biblioteca **scikit-learn**

class sklearn.ensemble.RandomForestClassifier (n_estimators=100, *, criterion{“**gini**”, “entropy”, “log_loss”}, max_depth = int, default=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)[source]

Los hiperparámetros permiten ajustar el modelo para aumentar la precisión del entrenamiento.

Tamaño de la submuestra: max_samples si bootstrap=True (predeterminado), o se utiliza todo el conjunto de datos para construir cada árbol.

Ejemplo. BA Clasificación binaria

Objetivo:

- Predecir la aparición de la diabetes en un plazo de 5 años según los detalles médicos proporcionados. Desde el abordaje de un problema de clasificación binaria.

Actividades:

- Analizar el problema
- Crear un modelo sobre aquel conjunto de datos para predecir si un paciente en particular tiene riesgo de desarrollar diabetes, dado otros factores independientes
- Validar la precisión del modelo
- Realizar predicciones con el modelo entrenado

Ejemplo. BA Clasificación binaria

Importación de paquetes para cargar el conjunto de datos y crear un clasificador de BA

biblioteca **scikit-learn** para cargar y usar el algoritmo de BA.

```
# importar paquetes
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas_profiling
from matplotlib import rcParams
import warnings
warnings.filterwarnings("ignore")
# tamaño de la figura en pulgadas
rcParams["figure.figsize"] = 10, 6
np.random.seed(42)
```

Ejemplo. BA Clasificación binaria

```
# Cargar conjunto de datos
data = pd.read_csv("../data/pima_indians_diabetes.csv")
# mostrar muestra del conjunto de datos
data.sample(5)
```

	time_pregnant_no	plasma_concentration	diastolic_blood_pressure	triceps_skinfold_thickness	serum_insulin	bmi	diabetes_pedigree	age	class
668	6	98	58	33	190	34.0	0.430	43	0
324	2	112	75	32	0	35.7	0.148	21	0
624	2	108	64	0	0	30.8	0.158	21	0
690	8	107	80	0	0	24.6	0.856	34	0
473	7	136	90	0	0	29.9	0.210	50	0

```
# visualizar las columnas o lista de características [variables] en el conjunto de datos.
data.columns
```

```
Index(['time_pregnant_no', 'plasma_concentration', 'diastolic_blood_pressure',
       'triceps_skinfold_thickness', 'serum_insulin', 'bmi',
       'diabetes_pedigree', 'age', 'class'],
      dtype='object')
```

Ejemplo. BA Clasificación binaria

```
# la variable de salida puede asumir los valores 0 o 1. Se debe dividir el conjunto de datos en características independientes y objetivo o meta

# drop: elimina una determinada variable del conjunto de datos de entrada

X = data.drop("class", axis=1)

y = data["class"]


# = train_test_Split: divide aleatoriamente en conjunto de entrenamiento (train) y  conjunto de prueba (test)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, stratify=y, test_size=0.10, random_state=42)
```

Ejemplo. BA Clasificación binaria

```
# crear un clasificador BA
classifier = RandomForestClassifier(n_estimators=100)

# Entrenar el modelo usando el conjunto de entrenamiento
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Ejemplo. BA Clasificación binaria

```
# realizar predicciones y analizar el comportamiento utilizando la métrica precisión

y_pred = classifier.predict(X_test)

# Calcular la precisión del modelo, comparación de valores obtenidos y los predichos

print("Precisión:", accuracy_score(y_test, y_pred))
```

Ejemplo. BA Clasificación binaria y preprocesamiento

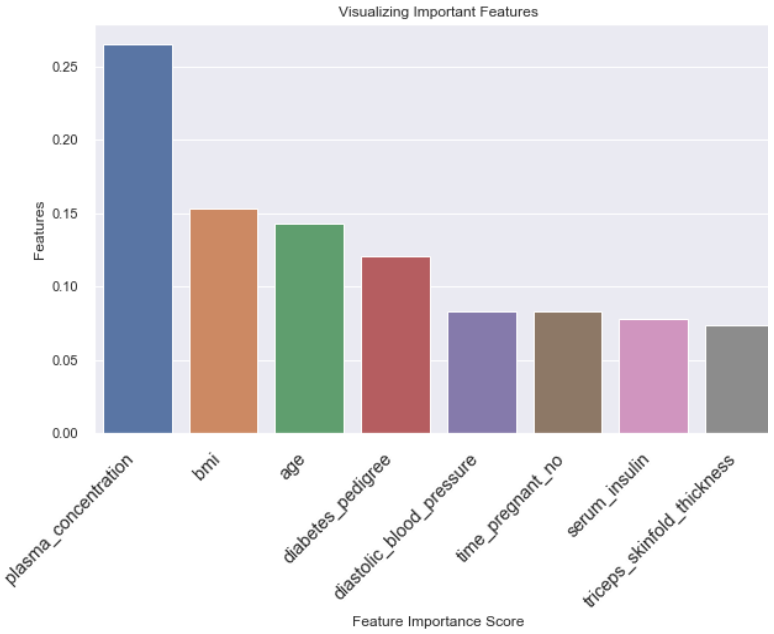
```
# identificar las variables menos relevantes y NO contemplar en la construcción del modelo BA#
feature_importances_df = pd.DataFrame(
    {"feature": list(X.columns), "importance": classifier.feature_importances_}
).sort_values("importance", ascending=False)

# listar las variables y la importancia en el conjunto de datos
feature_importances_df

# representación grafica de las variables y su importancia
sns.barplot(x=feature_importances_df.feature, y=feature_importances_df.importance)

# agregar estiquetas
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Visualizing Important Features")
plt.xticks(
    rotation=45, horizontalalignment="right", fontweight="light", fontsize="x-large"
)
plt.show()
```

	feature	importance
1	plasma_concentration	0.265153
5	bmi	0.152950
7	age	0.142551
6	diabetes_pedigree	0.120932
2	diastolic_blood_pressure	0.083460
0	time_pregnant_no	0.082878
4	serum_insulin	0.078441
3	triceps_skinfold_thickness	0.073634



Ejemplo. BA Clasificación binaria, 2do modelo

```
# construir un modelo BA con variables de mayor importancia


# se elimina la variable objetivo y la de menor incidencia
X = data.drop(["class", "triceps_skinfold_thickness"], axis=1)
y = data["class"]


# estandarizar el conjunto de datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# dividir en conjunto de entrenamiento y de prueba
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, stratify=y, test_size=0.10, random_state=42
)
```

Ejemplo. BA Clasificación binaria. 2do modelo

```
# crear un clasificador BA
modelo2 = RandomForestClassifier(n_estimators=100)

# Entrenar el modelo usando el conjunto de entranamiento
modelo2.fit(X_train, y_train)

# predicción en el conjunto de prueba
y_pred = modelo2.predict(X_test)

# Calcular la precisión del modelo,
print("Precisión:", accuracy_score(y_test, y_pred))
```

Ejemplo. BA Clasificación binaria.

¿Qué modelo se selecciona ?

Justificar la decisión

Phyton. BA, modelo de regression

biblioteca **scikit-learn**

Un bosque aleatorio es un metaestimador que se ajusta a una serie de árboles de decisión de clasificación en varias submuestras del conjunto de datos, y utiliza el promedio para mejorar la precisión predictiva y controlar el sobreajuste.

sklearn.ensemble.RandomForestRegressor — scikit-learn 1.3.0 documentation

Phyton. BA, modelo de regression

biblioteca **scikit-learn**

class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *, **criterion{*“squared_error”, “***absolute_error”, “friedman_mse”, “poisson”}, max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=1.0, max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None)[source]

sklearn.ensemble.RandomForestRegressor — scikit-learn 1.3.0 documentation

Actividad Práctica

Tareas

- Para optar por el modelo adecuado, se deben entrenar con diferentes configuraciones.
- Identificar los parámetros de RF para la mejora del modelo
- Establecer, al menos 2 configuraciones, entrenar y predecir, resumir los resultados
- Analizar la mejor configuración a partir de una métrica (ej. accuracy)

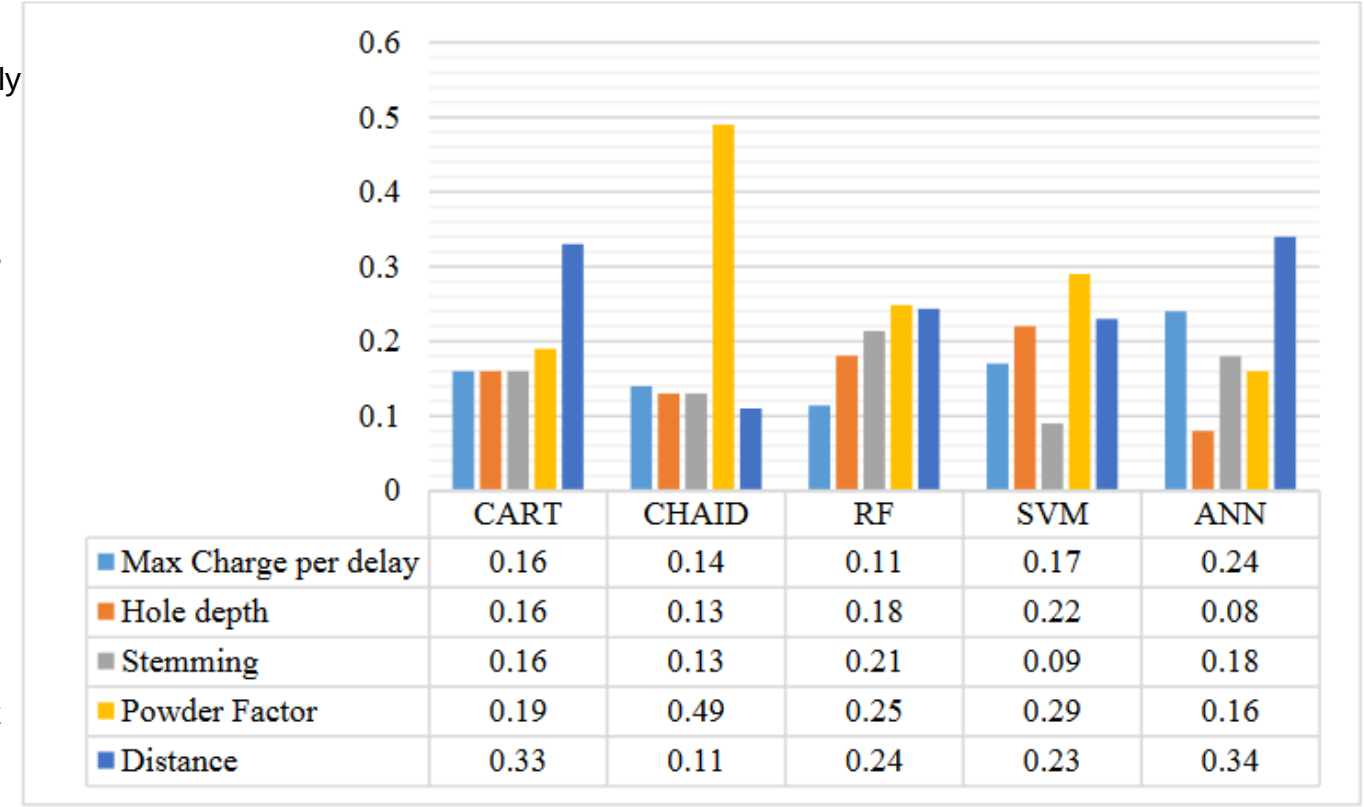
Aplicaciones (1)

A Combination of Feature Selection and Random Forest Techniques to Solve a Problem Related to Blast-Induced Ground Vibration

Abstract: In mining and civil engineering applications, a reliable and proper analysis of ground vibration due to quarry blasting is an extremely important task. While advances in machine learning led to numerous powerful regression models, the usefulness of these models for modeling the ***peak particle velocity (PPV***) remains largely unexplored. Using an extensive database comprising quarry site datasets enriched with vibration variables, this article compares **the predictive performance of five selected machine learning classifiers, including classification and regression trees (CART), chi-squared automatic interaction detection (CHAID), random forest (RF), artificial neural network (ANN), and support vector machine (SVM) for PPV analys**is. Before conducting these model developments, feature selection was applied in order to select the most important input parameters for PPV. The results of this study show that RF performed substantially better than any of the other investigated regression models, including the frequently used SVM and ANN models. The results and process analysis of this study can be utilized by other researchers/designers in similar fields.

Keywords: ground vibration; prediction; machine learning; feature selection; environmental issue of blasting

https://www.mdpi.com/2076-3417/10/3/869



	CART	CHAID	RF	SVM	ANN
Max Charge per delay	0.16	0.14	0.11	0.17	0.24
Hole depth	0.16	0.13	0.18	0.22	0.08
Stemming	0.16	0.13	0.21	0.09	0.18
Powder Factor	0.19	0.49	0.25	0.29	0.16
Distance	0.33	0.11	0.24	0.23	0.34

Figure 4. Variable importance through the developed models.

Aplicaciones (2)

Application of random forest regression and comparison of its performance to multiple linear regression in modeling groundwater nitrate concentration at the African continent scale

Abstract

Groundwater management decisions require robust methods that allow accurate predictive modeling of pollutant occurrences. In this study, random forest regression (RFR) was used for modeling groundwater nitrate contamination at the African continent scale. When compared to more conventional techniques, **key advantages of RFR include its nonparametric nature, its high predictive accuracy, and its capability to determine variable importance**. The latter can be used to better understand the individual role and the combined effect of explanatory variables in a predictive model. In the absence of a systematic groundwater monitoring program at the African continent scale, the study used the groundwater nitrate contamination database for the continent obtained from a meta-analysis to test the modeling approach; 250 groundwater nitrate pollution studies from the African continent were compiled using the literature data. A geographic information system database of 13 spatial attributes was collected, related to land use, soil type, hydrogeology, topography, climatology, type of region, and nitrogen fertilizer application rate, and these were assigned as predictors. The **RFR performance was evaluat**ed in comparison to the **multiple linear regression (MLR)** methods. By using RFR, it was possible to establish which explanatory variables influence the occurrence of nitrate pollution in groundwater (population density, rainfall, recharge, etc.). Both the RFR and MLR techniques identified population density as the most important variable explaining reported nitrate contamination. However, RFR has a much higher predictive power (R2 = 0.97) than a traditional linear regression model (R2 = 0.64). RFR is therefore considered a very promising technique for large-scale modeling of groundwater nitrate pollution

Model	Training data		Testing data	
	MSE	% variation explained	MSE	% variation explained
Random forest	0.04	0.97	0.01	0.98
Linear regression	0.95	0.64	0.75	0.54

MSE mean squared error

https://dial.uclouvain.be/pr/boreal/object/boreal%3A208690/datastream/PDF_01/view