A picture containing icon

Description automatically generated

UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

UNITEC

ASIGNATURA: SISTEMAS INTELIGENTES

SECCIÓN: 1296

Tarea #3 – Arboles y Bosques

PRESENTADO POR:

DANIEL AUGUSTO MORALES ALVARADO CTA: 11941247

JUDÁ AARON PONCE VILLALTA CTA: 11841248

CATEDRÁTICO: Dr. KENNY M. DÁVILA

CAMPUS TEGUCIGALPA 20 DE NOVIEMBRE DE 2022

índice

[Introducción 1](#_Toc119678324)

[Implementación 2](#_Toc119678325)

[Resultados 3](#_Toc119678326)

[Análisis 6](#_Toc119678327)

[Dificultades Encontradas 6](#_Toc119678328)

[Planteamiento 6](#_Toc119678329)

[Conclusiones 7](#_Toc119678330)

Introducción

El presente reporte evidenciará la aplicación de árboles de decisión y random forest, cada uno con sus propios atributos. Se analizará el comportamiento de entrenar un solo árbol como un bosque con n cantidad de árboles. Como sabemos un árbol decisiones se asemeja mucho a la vida diaria ya que depende de las decesiones que uno tome este traerá consecuencias buenas o malas. En este caso se entrenarán arboles con 4 tamaños distintos de detataset, así como también su criterio y profundidad. Al finalizar el entrenamiento con los diferentes parámetros se tendrán 32 combinaciones en total y esto datos serán escritos en un archivo CSV, por otro lado, el random forest se entrenará con los mismos 4 datasets, en cambio se reemplaza criterio por n cantidad de árboles para la evaluación de los resultados, mismos que serán escritos en un archivo CSV también y se tendrán 48 combinaciones en total.

Cuando hablamos de entrenamiento de estas estructuras no debemos de descartar el rendimiento de cada una de estas, es por ello que se calcularán diferentes tipos de métricas tomando en cuenta los atributos y el dataset con el que se va a entrenar, scikit-learn provee las funciones para hacer estos cálculos, todos estos serán guardados en un archivo CSV indicando el dataset, los atributos utilizados y promedios de entrenamiento y validación, también el tiempo que tardará en entrenar como en validar. Todos estos resultados serán analizados en este reporte.

En definitiva, este tipo de trabajos o experimentos son necesarios para comprender el proceso que lleva entrenar este tipo de estructuras considerando que entre mas datos haya disponibles para entrenar, los resultados serán mejores y más fáciles de analizar, con pocos datos es muy difícil poder sacar conclusiones claras y las futuras decisiones que tomaran posterior a esto.

Implementación

Para llevar a cabo esta tarea se realizaron 5 script distintos, 3 de estos solicitados previamente y 2 extra que tienen la misma funcionalidad que el script 1 y 2 con diferencia que realizan el entrenamiento de forma automática con las diferentes combinaciones.

Se utilizaron 5 librerías en total que son las siguientes:

* sys
* pandas
* time
* pickle
* scikit-learn

Para poder ejecutar el script 1 relacionado con decision tree, basta con colocar el siguiente comando: py.\Tipo entrenamiento .\dataset con el que se quiere entrenar criterio profundidad, por ejemplo: py.\decision\_tree.py .\training\_data\_small.csv gini 2.

Hay que tomar en cuanta que tanto el dataset, criterio y profundidad son variables, es decir que por cada vez que se quiera entrenar esos parámetros pueden cambiar, para incluir la opción “None” basta con obviar la profundidad, por ejemplo: py.\decision\_tree.py .\training\_data\_small.csv gini. Para el caso de random forest solo se cambia el tipo de entrenamiento y criterio por n árboles, por ejemplo: py.\random\_forest.py .\training\_data\_small.csv 10 2.

Para el script 3 se coloca el siguiente comando: py \model\_accuracy\_validator.py .\Modelos\RandomForest\RandomForest-very-100-4.pkl .\validation\_data.csv, aquí se debe de acceder a la carpeta modelos que contiene tanto los arboles como los random forest ya entrenados.

Los nombres de los archivos entrenados tienen la siguiente nomenclatura: RandomForest-datasetSize-n\_estimators-depth.pkl, y DecisionTree-datasetSize-criterio-profundidad.pkl. Para los 2 script extra solo es necesario el siguiente comando: py .\scriptAutomatizadoDecision.py o py .\scriptAutomatizadoRandom.py, que como se menciona al inicio, entrena de manera automática todas las combinaciones solicitadas.

Resultados

Para guardar los resultados tanto de decision tree y random forest, se crearon archivos CSV dentro del script 3 que permite la creación y escritura o solo escritura para los modelos que se vayan validando y que se guarden el archivo correcto y por orden, es decir por size, criterio-depth para decision tree y size nTree-depth para random forest, y cada calculo delimitado por un coma (,).

Los resultados que se obtuvieron fueron los siguientes:

Text

Description automatically generated

Ilustración Resultados para decision tree (32 combinaciones)

Text

Description automatically generated

Ilustración Resultados random forest (48 combinaciones)

Análisis

Dificultades Encontradas

Al inicio fue confuso la realización de esta tarea, ya que estábamos validando datos donde no era. Por otro lado, para generar la tabla no podíamos acceder a todos los cálculos realizados en los otros scripts, por lo que fue necesario guardar en un objeto de tipo pickle para poder obtener y realizar las métricas solicitadas y que la mayoría de estas debían ser calculadas en el 3er script. Otra dificultad que en una parte de la documentación de scikit-learn una de las funciones para calcular las métricas estaba obsoletas y esto generaba resultados extraños. También para validar con modelos con dataset muy grandes salía un error de la división 0 ya que los valores eran continuos o infinitos por lo tanto se agregó un argumento mas que lo recomiendo scikit-learn denominado “zero division”.

Planteamiento

Conclusiones

* Se entrenaron de la manera correcta y deseada arboles y bosques con los atributos solicitados y el total de combinaciones esperadas. La librería scikit-learn es demasiado útil, su documentación es clara y muy precisa y ayudó para la finalización de los entrenamientos.
* Se almacenaron los modelos entrenados en formato .pickle, nos ayudó bastante y permitió guardar el modelo completo y con sus atributos para que estos fueran fáciles de extraer en otros scripts para calcular las métricas de dicho modelo.
* Se comprendió el funcionamiento y el proceso que lleva el entrenamiento, también se analizaron los resultados de cada modelo mismo que ayudaran en gran manera a predecir en este caso el gusto particular de los restaurantes.