1. Lab. Árboles decisión, reglas y ensemble learning

En el presente informe se presenta un resumen e interpretación de la implementación de algunos de los algoritmos, que usan la aplicación técnica de aprendizaje supervisado (clasificación), para el dataset **Laboratorio_dataset_car.csv,** entre los cuales están:

- **CART**: Árbol de decisión (Índice Gini).
- **ID3**: Árbol de decisión (Ganancia de información - Entropía).
- **Random Forest** (Ensemble learning).
- **Prism**: Reglas de clasificación.

El ejercicio fue implementado mediante Jupyter Nootebooks Versión 6.4.2 (https://jupyter.org/), Python versión 3.0 (https://www.python.org/) y algunas librerías para la manipulación de datos y machine learning, en el siguiente repositorio de GitHub se encuentran los recursos o artefactos utilizados.

El proceso de EDA (Análisis Exploratorio de datos), se fundamento en el siguiente paso a paso (https://datos.gob.es/es/documentacion/guia-practica-de-introduccion-al-analisis-exploratorio-de-datos), en el cual se validaron los siguintes aspectos:

1. Definir formato de codificación del archivo UTF-8 (Contempla todos los caracteres especiales).
2. Análisis y descripción de los datos (pandas.df.info(), pandas.df.limit(4), pandas.df.shape, pandas.df.describe(), imprimir_estadisticas(mean, median, mode, std, var, percentile)).
3. Validación de duplicados (df_pd.drop_duplicates(), count_uniques_val_cols(), dataset.nunique()).
4. Detección y tratamiento de datos ausentes (dataset.isnull().sum()).
5. Ajuste de tipos de variables (Remplace de valores atípicos, conversión de valores categóricos a números, trimp(), upper()).
6. Redefinir esquema (conversión tipos de datos a enteros sin incluir la clase).
7. Gráficos y análisis de correlación de variables (comparación clase y atributos).

Los resultados obtenidos se describen a continuación:

Tabla 1. Muestra tipos de datos originales y transformados a partir del Dataset.

Atributo	Tipo original	Tipo conversión	Descripción	Tipo dato original	Tipo dato conversión	Valores originales	Valores remplazados
Buying	cualitativa (ordinal)	cuantitativa (discreta)	Precio de compra	object (str)	int	'vhigh', 'high', 'med', 'low'	3, 2, 1, 0
Maintenance	cualitativa (ordinal)	cuantitativa (discreta)	Precio de mantenimiento	object (str)	int	'vhigh', 'high', 'med', 'low'	3, 2, 1, 0
Doors	cuantitativa (discreta)	cuantitativa (discreta)	Numero de puertas	object (str)	int	2, 3, 4, 5 más	2, 3, 4, 5
Person	cuantitativa (discreta)	cuantitativa (discreta)	Capacidad personas	object (str)	int	2, 4, más	2, 5, 5
Lug_boot	cualitativa (ordinal)	cuantitativa (discreta)	Capacidad del maletero	object (str)	int	'small','med','big'	0, 1, 2
Safety	cualitativa (ordinal)	cuantitativa (discreta)	Seguridad estimada	object (str)	int	'low','med','high'	0, 1, 2
Class	cualitativa (ordinal)	cualitativa (ordinal)	Aceptabilidad automovil	object (str)	category (str)	acc, good, unacc, vgood	acc, good, unacc, vgood

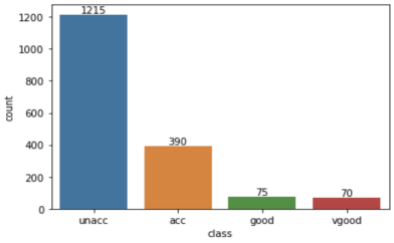
Número de instancias en total: 1750.

Número de atributos: 7 incluyendo la clase.

Tabla 1. Resumen distribución de los datos por cada atributo.

Buying		Maintenance		Doors		Person		lug_boot		safety		class	
high	432	high	432	2	444	2	578	big	585	high	590	acc	390
low	437	low	447	3	435	4	587	med	583	low	578	good	75
med	438	med	434	4	434	more	585	small	582	med	582	unacc	1215
vhigh	443	vhigh	437	5more	437							vgood	70

Imagen 1. Cantidad de instancias por cada clase.



Los tipos de datos de la variable clase son cualitativos-categorico: De acuerdo con los a tributos de la instancia, indican los cuales indican la aceptabiliad o evealuación de un auto.

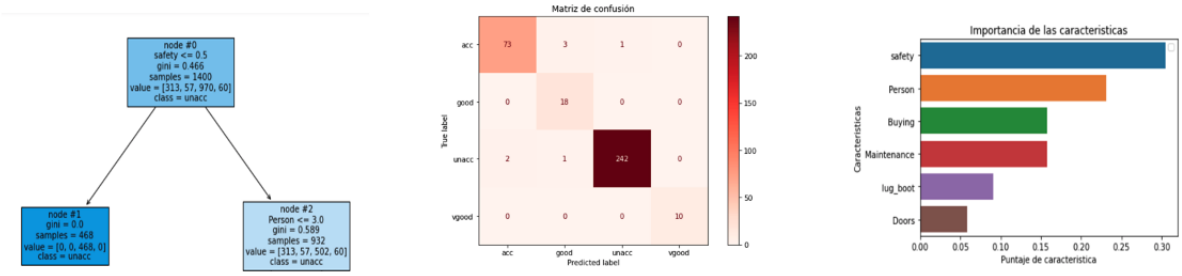
- Acceptable (acc).
- Good (good).
- Unacceptable (unacc).
- Very Good (vgood).

El laboratorio nos proporciona el dataset **Laboratorio_dataset_car.csv**, el cual plantea el problema sobre la aceptabilidad o evaluación de un automóvil según los atributos diferentes a la columna **class** descritos en el dataset, para llevar a cabo la clasificación de las instancias o ejemplos vamos a describir la implementación para cada algoritmo:

2. Árboles decisión, algoritmo CART:

El Dataset con el que trabajaremos tiene un numero considerable de registros, por lo cual para llevar a cabo la implementación del algoritmo utilizamos un modelo entrenado con un 20% de los datos y un 80% para entrenarlo, se utilizo la librería de python “from sklearn.tree import DecisionTreeClassifier”, a la cual le especificamos como parámetros el índice Gini utilizado por el algoritmo para crear puntos de división en el árbol y una muestra no aleatoria, como resultado se obtuvo el siguiente resultado:

Imagen 2. Matriz de confusión y árbol resultante para CART.



De la **Imagen 2**, podemos deducir que el atributo seleccionado con mayor importancia para evaluar un automóvil es la seguridad. Con respecto a la matriz de confusión podemos apreciar que para cada atributo mayoría de evaluaciones correctas **TP**, se encuentran en la diagonal, hay pocos valores evaluados como **FP** y **FN,** lo cual nos da un buen indicio de que el algoritmo tiene un buen rendimiento para clasificar.

Tabla 2. FP y TP Rates, clasificación de instancias algoritmo CART.

Algoritmo	Instancias clasificadas correctamente	Instancias clasificadas incorrectamente	TP Rate	FP Rate
CART	98% de 350 aprox **(343)**	2% de 350 **(17)**	acc: 0,94 good: 1 unacc: 0,98 vgood: 1	acc: 0,007 good: 0,012 unacc: 0,009 vgood: 0

De la **Tabla 2**, podemos deducir que el algoritmo tiene un alto índice de instancias clasificadas correctamente, lo cual se puede notar en el indicador **TP Rate**, donde todos los valores son cercanos a uno y por el contrario el indicador **FP Rate** los valores son cercanos a cero para cada categoría.

Tabla 3. Métricas de rendimiento del algoritmo CART

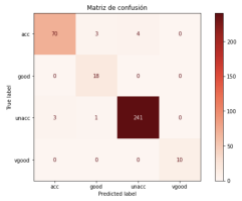
	precision	recall	f1-score	support
acc	0.9733	0.9481	0.9605	77
good	0.8182	1.0000	0.9000	18
unacc	0.9959	0.9878	0.9918	245
vgood	1.0000	1.0000	1.0000	10
accuracy			0.9800	350
macro avg	0.9468	0.9840	0.9631	350
weighted avg	0.9819	0.9800	0.9804	350

De la **Tabla 3**, vemos que la precisión y el **Recall** o sensibilidad tienen valores muy cercanos a 1, esto nos indica que tenemos un modelo confiable para clasificar, los valores del indicador **F1** representa la relación entre precisión y sensibilidad. El indicador de **exactitud (accuracy)** del modelo es del 98% lo cual se establece a partir de las predicciones correctas y el número total de predicciones.

3. Árboles decisión, algoritmo ID3:

Para llevar a cabo la implementación del algoritmo utilizamos un modelo entrenado con un 20% de los datos y un 80% para entrenarlo, se utilizo la librería de python “from sklearn.tree import DecisionTreeClassifier a diferencia que en este utilizamos el parámetro **criterion='entropy'.**

Imagen 3. Matriz de confusión y métricas de rendimiento para ID3**.**

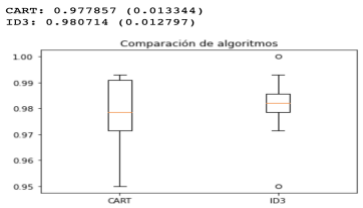


	precision	recall	f1-score	support
acc	0.9589	0.9091	0.9333	77
good	0.8182	1.0000	0.9000	18
unacc	0.9837	0.9837	0.9837	245
vgood	1.0000	1.0000	1.0000	10
accuracy			0.9686	350
macro avg	0.9402	0.9732	0.9543	350
weighted avg	0.9702	0.9686	0.9688	350

Tabla 4. FP y TP Rates, clasificación de instancias algoritmo CART vs ID3.

Algoritmo	Instancias clasificadas correctamente	Instancias clasificadas incorrectamente	TP Rate	FP Rate
CART	98% de 350 aprox **(343)**	2% de 350 aprox **(17)**	acc: 0,94 good: 1 unacc: 0,98 vgood: 1	acc: 0,007 good: 0,012 unacc: 0,009 vgood: 0
ID3	97% de 350 aprox **(341)**	2% de 350 aprox **(11)**	acc: 0,90 good: 1 unacc: 0,98 vgood: 1	acc: 0,010 good: 0,012 unacc: 0,038 vgood: 0

Imagen 4. Validación cruzada estratificada 10 veces (k-fold)



La gráfica comparativa de la **Imagen 4**, compara la precisión de los modelos CART e ID3, los cuales tienen una precisión del 97% y 98%. Para extremos superiores de los bigotes están por encima del 99% Para CART y para ID3 se marca un valor de 100%, lo cual es normal ya que algunas pruebas o resultados de las predicciones de clasificación la precisión es del 100%. El cuartil inferior permanece por encima del 97% mientras que el de CART por encima del 95%.

4. Ranfom Forest, Ensemble Learning

Para llevar a cabo la implementación del algoritmo utilizamos un modelo entrenado con un 20% de los datos y un 80% para entrenarlo, se realizaron 2 pruebas en las cuales se entreno el modelo con un total de 100 árboles y otro con 2000 árboles. Para implementarlo se utilizo la librería de python “from sklearn.ensemble import RandomForestClassifier”, los parámetros asignados son:

- n_estimators = 100 y 2000
- criterion = 'gini'

Como resultado de ambos modelos tenemos que la precisión no aumento considerablemente (Modelo 1: 98%, modelo 2: 94%) a pesar de que eliminamos las variables menos importantes como el número de puertas "**Doors**", lo que nos indica que probablemente no había datos engañosos o que generaran ruido al modelo. Por otro lado, el tiempo de entrenamiento debe ser menor ya que eliminamos una variable del modelo. También en el segundo modelo menos importante es lug_boot fue eliminado en las pruebas.

pero se obtuvo como resultado que precisión era 85%, lo cual afecta a la precisión del modelo por lo cual tampoco es viable quitarla del modelo.

Se implemento una prueba adicional, en la cual no se elimina la variable "**Door**", pero incrementando el parametro de número de árboles a evaluar en el modelo n_estimators=2000, en la segunda instancia creada, se obtuvo una precisión mayor a la del primer modelo, los valores son los siguientes:

- Modelo 1, arboles 100, accuracy 94%
- Modelo 2, arboles 2000, accuracy 97%

En el segundo modelo se sacrifica un poco el tiempo de generación y predicción del modelo, pero obtenemos un resultado de precisión mejor al momento de clasificar las instancias.

Tabla 5. Medidas de rendimiento Random Forest.

	precision	recall	f1-score	support
(scroll output; double click to hide)	0.99	0.96	0.97	77
good	0.89	0.94	0.92	18
unacc	1.00	1.00	1.00	245
vgood	0.83	1.00	0.91	10
accuracy			0.99	350
macro avg	0.93	0.98	0.95	350
weighted avg	0.99	0.99	0.99	350

En la Tabla 5 se obtiene como resultado una, medida de precisión del modelo de 99%, este escenario fue resultado de evaluar el modelo sin eliminar ninguna de las variables, hasta el momento la precisión de este algoritmo es la mas eficiente con respecto a los ya evaluados.

5. Reglas de clasificación, algoritmo Prism:

Para llevar a cabo la implementación del algoritmo utilizamos un modelo entrenado con un 20% de los datos y un 80% para entrenarlo, se utilizo la librería de python “weka.classifiers.rules.Prism”, a la cual le especificamos como parámetros el índice Gini utilizado por el algoritmo para crear puntos de división en el árbol y una muestra no aleatoria.

6. Comparativa:

A continuación, se muestran algunos gráficos indicadores tabulares que nos permiten hacer una comparativa entre los algoritmos:

Imagen 5. Comparativo rendimiento CART, ID3 y RAND FOREST.

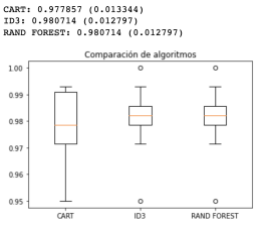


Imagen 6. Curva ROC.

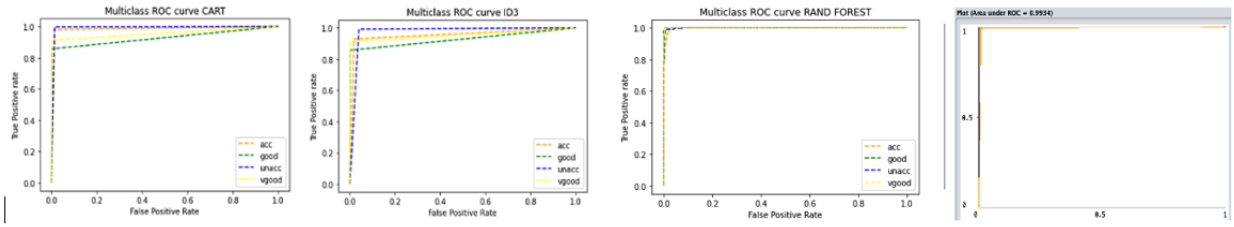
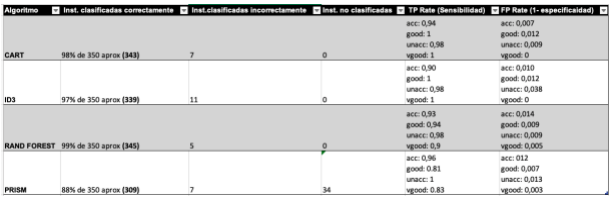


Tabla 7. Comparativa clasificación instancias.



Algoritmo	Inst. clasificadas correctamente	Inst.clasificadas incorrectamente	Inst. no clasificadas	TP Rate (Sensibilidad)	FP Rate (1- especificidad)
CART	98% de 350 aprox (343)	7	0	acc: 0,94 good: 1 unacc: 0,98 vgood: 1	acc: 0,007 good: 0,012 unacc: 0,009 vgood: 0
ID3	97% de 350 aprox (339)	11	0	acc: 0,90 good: 1 unacc: 0,98 vgood: 1	acc: 0,010 good: 0,012 unacc: 0,038 vgood: 0
RAND FOREST	99% de 350 aprox (345)	5	0	acc: 0,93 good: 0,94 unacc: 0,98 vgood: 0,9	acc: 0,014 good: 0,009 unacc: 0,009 vgood: 0,005
PRISM	88% de 350 aprox (309)	7	34	acc: 0,96 good: 0,81 unacc: 1 vgood: 0,83	acc: 012 good: 0,007 unacc: 0,013 vgood: 0,003

Tabla 8. Medidas de rendimiento.

Algoritmo	ROC Curv Auc	Avg. Precision	Avg. Sensibilidad (Recal	Avg. F-mesure	Avg. Exactitud (Accuracy	Avg. FP Rate (Fall ou	Avg. TP Rate (Sensibilidad	Avg. Especificidad TN	Soporte
RAND FOREST	0.99	0.98	0.98	0.98	0.98	0.008	0.94	0.99	350
CART	0.96	0.98	0.98	0.98	0.99	0.0072	0.016	0.98	350
ID3	0.95	0.97	0.96	0.96	0.96	0.015	0.97	0.98	350
PRISM	0.94	0.97	0.97	0.97	0.88	0.012	0.97	-	350

7. Conclusiones

- Las variables seguridad y capacidad de pasajeros son factores clave a la hora de evaluar la compra de un auto móvil.
- El número de puertas y la capacidad del maletero son los dos factores menos importantes a la hora de evaluar un automóvil, sin embargo, la información contenida en dichas variables aporto valor a la predicción de los algoritmos.
- El algoritmo que clasifico las instancias con mayor precisión fue el Random Forest utilizando como índice Gini, ya que utiliza un promedio de la clasificación de instancias a partir de 100 y 2000 árboles.
- La matriz de confusión y las métricas de rendimiento tales como (Avg. Precisión (**1 es el estado ideal**), Avg. Sensibilidad (Recall) **1 es el estado ideal**, Avg. F-mesure **1 es el estado ideal**, Avg. Exactitud (Accuracy) **1 es el estado ideal**, Avg. FP Rate (Fall out) **mejor, 0 es el estado ideal**, Avg. TP Rate (Sensibilidad) **mejor, 1 es el estado ideal**, Avg. Especificidad, TNR Soporte), evidencian valores de rendimiento muy confiables para cada modelo.
- El bosque aleatorio ayuda a evitar el sobreajuste, que es uno de los problemas clave con el clasificador de árboles de decisión, la cantidad de arboles utilizados en este algoritmo afecta el rendimiento de tiempo para entrenar y predecir, pero por otro lado es clave para obtener una mayor precisión en los resultados de clasificación.
- Los índices de la curva de ROC-AUC comprenden valores entre 0.5 y 1, siendo 1 un diagnostico perfecto, nuestros resultados obtenidos son muy buenos, ya que todos están por encima del 90%, destacando el algoritmo Random Forest el cual es de 99%.
- Con la matriz de confusión podemos ver que hay muy pocos FP y FN en cada variable, por cada modelo evaluado.

8. Referencias bibliográficas

datos.gob.es. (22 de 9 de 2021). *datos.gob.es.* Obtenido de Guía Práctica de Introducción al Análisis Exploratorio de Datos: https://datos.gob.es/es/documentacion/guia-practica-de-introduccion-al-analisis-exploratorio-de-datos

Rocio Chavez Ciencia de Datos. (2020). Histogramas en Python [video]. YouTube.

AIEngineering. (2020). Data Cleaning and Analysis using Apache Spark [video]. YouTube.

Un Analista de Datos Peruano. (2019). Para qué me sirve la Desviación Estandar? [video]. YouTube.

orvizar TV. (2020). Pandas desde cero #3 |Limpiar Datos | Valores nulos | Eliminar datos | tutorial español [video]. YouTube.

Dattos.org. (2019). Gráficos en Python con Matplotlib, Seaborn y Ploty. YouTube.

Tomas_IA. (2020). Limpieza de datos con Python [video]. YouTube.

Escuela de datos. (28 de 6 de 2017). *Escuela de datos*. Obtenido de Introducción a Pandas y Jupyter Notebook de Python: http://es.schoolofdata.org/2017/06/28/introduccion-a-pandas-y-jupyter-notebook-de-python/

Invarato, R. (20 de 4 de 2021). *jarroba.com*. Obtenido de Instalar Apache Spark en cualquier sistema operativo y aprender a programarlo con Python, Scala o Java: https://jarroba.com/instalar-apache-spark-en-cualquier-sistema-operativo-y-aprender-a-programarlo-con-python-scala-o-java/

Universidad Internacional de La Rioja (UNIR). (2021). *unir.net.* Obtenido de cms.unir.net: https://cms.unir.net/ezpdf/generate/topic/295302

Universidad Internacional de La Rioja (UNIR). (2021). *Tema 3. Árboles de decisión.* Obtenido de cms.unir.net: https://cms.unir.net/ezpdf/generate/topic/296148

V, C. (18 de 10 de 2015). *Customer acceptability evaluation on Cars using Classification.* Obtenido de rstudio-pubs-static.s3.amazonaws.com: https://rstudio-pubs-static.s3.amazonaws.com/118220_5a7997d6b0aa493c878d661968fc1f08.html

PAHWA, H. (01 de 2022). *Classification of car's acceptability.* Obtenido de kaggle: https://www.kaggle.com/retroflake/classification-of-car-s-acceptability

Resolver, P. (4 de 8 de 2021). Arbol de Decision (Decision Tree Classifier) en Python. Salamanca, Salamanca, España.

Ünlü, T. (24 de 6 de 2020). *Classification of Car Evaluation Data Set by Decision Tree Algorithm (RStudio).* Obtenido de medium: https://medium.com/data-science-

practices/classification-of-car-evaluation-data-set-by-decision-tree-algorithm-rstudio-3a1c2fbe02c6

BANERJEE, P. (2020). *Decision-Tree Classifier Tutorial.* Obtenido de kaggle: https://www.kaggle.com/prashant111/decision-tree-classifier-tutorial

Gonzalez, A. c. (14 de 6 de 2019). CONJUNTO DE DATOS DESBALANCEADO | #36 Curso Machine Learning con Python.

Kumar, A. (21 de 7 de 2020). *vitalflux.* Obtenido de Random Forest Classifier Python Code Example: https://vitalflux.com/random-forest-classifier-python-code-example/

A CLOUD GURU. (2021). *Creating a scikit-learn Random Forest Classifier in AWS SageMaker*. Obtenido de learn.acloud.guru/: https://learn.acloud.guru/search?query=Creating%20a%20scikit-learn%20Random%20Forest%20Classifier%20in%20AWS%20SageMaker&page=1

BANERJEE, P. (2020). *Random Forest Classifier Tutorial.* Obtenido de kaggle: https://www.kaggle.com/prashant111/random-forest-classifier-tutorial

Reutemann, P. ". (2019). *PyWeka.* Obtenido de fracpete.github.io: https://fracpete.github.io/python-weka-wrapper/examples.html

futurelearn. (s.f.). *Adding PRISM to Weka.* Obtenido de futurelearn: https://www.futurelearn.com/info/courses/more-data-mining-with-weka/0/steps/29122

Wikipedia La enciclopedia libre. (8 de 8 de 2021). *Curva ROC.* Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Curva_ROC

programador clic. (2020). *Curva ROC de varias clases.* Obtenido de programmerclick: https://programmerclick.com/article/77801955644/