DESARROLLO ACTIVIDAD LABORATORIO: ARBOLES DE DECISION, REGLAS Y ENSEMBLE LEARNING

1. Descripción Inicial:

Librerías importantes usadas: pandas, matplotlib, seaborn, scikit-learn.

Entorno de python: 3.10

El dataset es acerca de datos correspondientes a evaluación de aspectos de carros con respecto a su nivel de seguridad, las instancias son 1750 en total. Como variables de entrada existen Buying, manintenance, Doors, person, lug_boot, class y la variable objetivo es safety para este caso en particular. Safety posee 3 valores para las instancias las cuales son: Low(bajo), med(Medio) y High(Alto).

Variables de entrada:

buying: vhigh, high, med, low.

maint: vhigh, high, med, low.

doors: 2, 3, 4, 5more.

persons: 2, 4, more.

Class: unacc, acc, good, vgood

lug_boot: small, med, big.

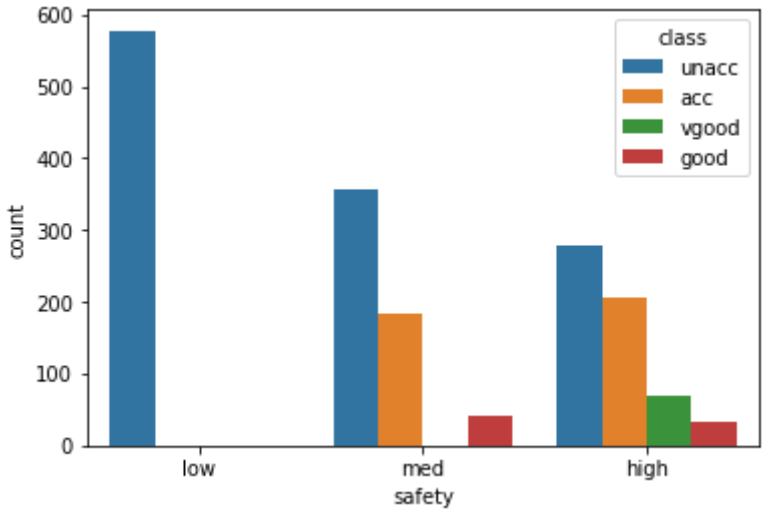
Variable Objetivo:

safety: low, med, high.

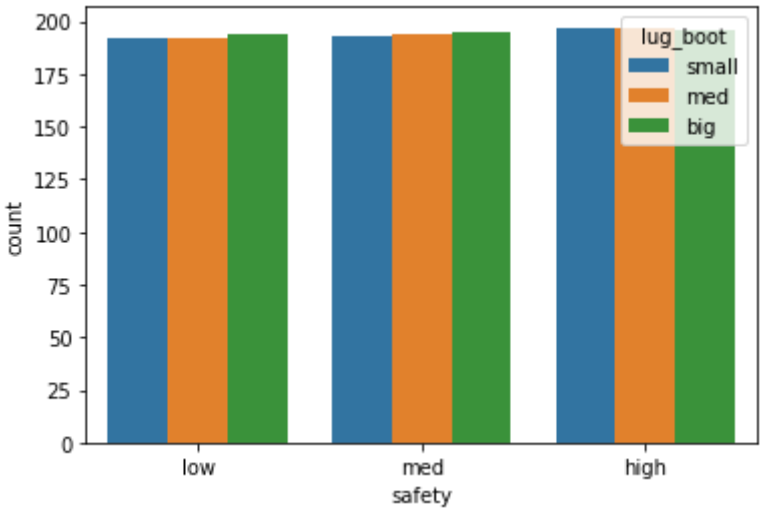
El problema es determinar ¿cuál es la clase con mejor probabilidad de clasificación para las tres clases usando arboles de decisión y random forest? con esto se podría clasificar la seguridad de autos nuevos con base en los diferentes algoritmos.

Se evidencian tres clases de la variable objetivo safety, las cuales determinan el nivel de seguridad del auto según la evaluación low (Bajo), Med(Medio), High(Alto). 1750 instancias en total por 7 columnas, todas las variables son de tipo Object por tal motivo se debe proceder a un encoding para trabajar el algoritmo de mejor manera.

Como se evidencia en la figura La mayor cantidad de autos categorizados como no accesibles llega casi a las 600 observaciones y se categoriza como seguridad baja. A medida que aumenta el nivel de seguridad así mismo disminuye la relación de no accesible. la distribución de autos muy bueno o buenos en cuanto a seguridad es demasiado bajo. Se debe observar al final de realizar el algoritmo si es más considerable balancear el dataset o si es suficiente con la variable objetivo safety que presenta buena distribución.



En cuanto a distribución de la variable safety vs el tamaño del maletero se presenta uniformidad, es determinante la variable class, no tanto las demás para construir las reglas del arbol para tener atención en la creación del modelo.

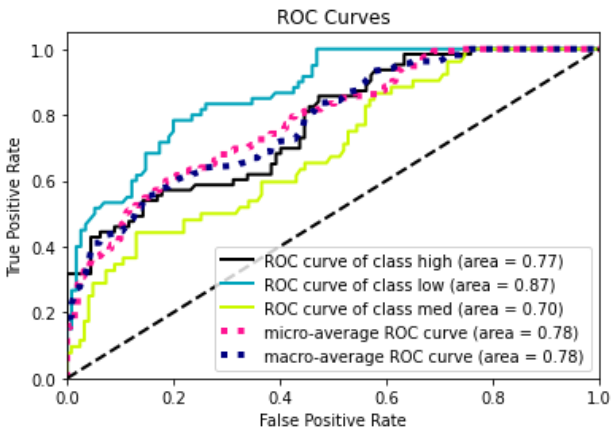


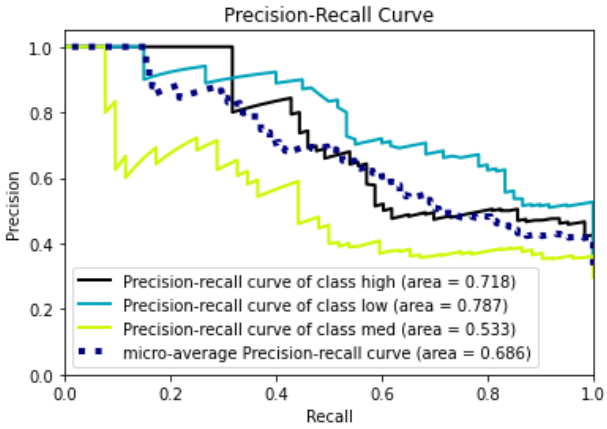
Se evidencia una relación clave entre safety y class, normalmente el criterio de variable objetivo debería ser class con sus niveles unacc, acc, vgood y good pero para este caso en particular se desea clasificar por medio del nivel de safety low,med y high. Se debe revisar como afecta el balanceo del dataset de manera positiva o negativa la clasificación y en caso de ser necesario nivelar la data. En cuanto a las demas variables presentan una distribución uniforme y se debe utilizar un enconding con el fin de mejorar el performance se sugiere variables dummies y/o one hot encoding.

2. Encoding, Partición y Modelado

Se realizó la partición de la data en variable X = Entrada. Por otro lado, y= Variable objetivo, para posteriormente aplicar variables dummies u one hot con pandas. se realiza una partición inicial de 90% datos de entrenamiento y 10% de datos de testeo y un random state de 100. El random state consiste en la forma o iteraciones que realiza con los datos de X y Y para seleccionar los valores a usar en el modelo según entrenamiento y testeo. Se realizan tres modelos (Cart Gini, Cart Entropy y Random Forest) evidenciando mejor eficiencia en el

algoritmo de random forest con un 0.57 de accuracy pero mejor rendimiento en la curva de ROC – AUC. De igual manera se utilizó GridSearchCV para realizar una validación de los mejores criterios para realizar la clasificación donde la profundidad del arbol en 4, estimadores en 200 y el criterio de entropia es el mejor como se evidencia en la gráfica.





Se determino para la construcción del modelo de arbol de decisión dos momentos uno con el coeficiente de gini y el otro con entropy y una máxima profundidad de 3. Los datos se repartieron en 90% train y 10% test ya que con dichas configuraciones presentaba la mayor eficiencia de 0.56 si bien no es un modelo muy bueno se evidencia una alta eficiencia de clasificación de instancias correctas para la clase low con 137 de 175 y para las demas high y med con un valor de 117. En cuanto a la sensibilidad y especificidad la mejor clase para clasificar es Low con unos valores de 0.78 en ambas métricas. Tanto para las clases High como Med el True Negative posee el mejor valor, es decir el modelo es muy bueno para clasificar cuando el carro no es seguro. Esto puede ser por el tema del balanceo del dataset en la columna class. Finalmente, la eficiencia del modelo en general es de 0.56 es decir para instancias futuras puede clasificar de manera correcta de 100 datos 56.

	precision	recall	f1-score	support
high	0.551020	0.428571	0.482143	63.00
low	0.652778	0.783333	0.712121	60.00
med	0.444444	0.461538	0.452830	52.00
accuracy	0.560000	0.560000	0.560000	0.56
macro avg	0.549414	0.557814	0.549031	175.00
weighted avg	0.554240	0.560000	0.552283	175.00

Para la construcción del algoritmo random forest se ejecutó la validación GridSEarchCV para encontrar los mejores parámetros de ejecucción del modelo. Los datos se repartieron en 90% train y 10% test ya que con dichas configuraciones presentaba la mayor eficiencia de 0.57 si bien no es un modelo muy bueno se evidencia una alta eficiencia de clasificación de instancias correctas para la clase low con 119 de 175 y para las demas high y med con un valor de 128 y 126 respectivamente. En cuanto a la sensibilidad y especificidad la mejor clase

para clasificar es Low con unos valores de 1.0. Tanto para las clases High como Med el True Negative posee el mejor valor, es decir el modelo es muy bueno para clasificar cuando el carro no es seguro.

	precision	recall	f1-score	support
high	0.681818	0.476190	0.560748	63.000000
low	0.517241	1.000000	0.681818	60.000000
med	0.600000	0.173077	0.268657	52.000000
accuracy	0.565714	0.565714	0.565714	0.565714
macro avg	0.599687	0.549756	0.503741	175.000000
weighted avg	0.601080	0.565714	0.515465	175.000000

3. Conclusiones

Como conclusiones finales se encuentra que ambos modelos son muy sensibles a sus configuraciones con los parámetros de testeo y entrenamiento al iterar la mejor recomendación es la relación 90% entrenamiento y 10% testeo.

De igual forma se utilizó el criterio de gini y entropy para verificar la eficiencia siendo idénticas. Al no afectar se revisó el overfitting y underfitting del modelo obteniendo en ambos casos una diferencia de 0.04 puntos. Lo cual indica un modelo no sesgado, lo cual fue clave la codificación usando las variables dummies que provee pandas.

En cuanto a las variables importantes se evidencia que Person_2, class_unacc, class_vgood, son los atributos importantes que determinan la clasificación, la profundidad máxima de los árboles fue de 3 para obtener la mejor eficiencia siendo importante la librería scikit learn con poda.

Importante el uso de la técnica de hiper parametrización gridsearchCV para identificar los mejores criterios en el algoritmo random forest, siendo entropy y la profundidad de 4 árboles la mejor. Finalizando con una mejora de la clase low y med para clasificar y High siendo la menos fácil de clasificar. Ambos modelos no son lo suficiente buenos, pero se puede quitar atributos para mejorar la eficiencia o revisar correlaciones después del encoding para quitar valores que no aportan.

Finalmente, se recomienda seguir mejorando el algoritmo de randomo forest mejorando la calidad de la data para acercarse más a una eficiencia coherente y sinergía entre sensibilidad y especificidad para poder estar equilibrado en las tres clases y ejecutar el algoritmo balanceando el atributo class.