UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Informe Ejecutivo: Análisis de clasificación de Asteroides Peligrosos

TICS411: Minería de Datos - Sección 3

Autores:

Alumno 1: *Condori, Mayra*

Alumno 2: *Chávarry, Sebastián*

Alumno 3: *Lagos, Conzuelo*

Alumno 4: *Palma, Diego*

Alumno 5: *Tapia, Nícolas*

Profesor: Andrés Medina

FACULTAD DE INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

30/11/2021

1. Introducción

El Análisis de clasificación es una tarea de análisis de datos dentro de la Minería de datos, la cual identifica y asigna categorías a una colección de datos para permitir un análisis más preciso. Este tipo de análisis se puede utilizar para cuestionar, tomar una decisión o clasificar clases mediante el uso de algún algoritmo. En ese sentido, en el presente informe se busca realizar la clasificación de un conjunto de datos de asteroides recolectados por la NASA, y determinar si algún asteroide podría llegar a ser peligroso o no para la tierra, teniendo en cuenta que recae en peligroso si su diámetro es sobre 2 km. Para realizar dicha clasificación se proporciona la información de algunas características importantes de los asteroides, tales como su periodo orbital, su periodo de rotación, etc.

Para realizar el análisis de clasificación utilizaremos los algoritmos de K-Nearest Neighborhood (KNN), Support Vector Machine (SVM), Redes Neuronales, Regresión Logística y Árboles de decisión. Previo al análisis de cada algoritmo se realiza una limpieza de los datos y una exploración descriptiva de los mismos. Luego, se evidencian resultados de cada modelo desarrollado por el algoritmo y el entrenamiento de los mismos. Finalmente, se selecciona el mejor modelo entre todos los algoritmos y se realizan conclusiones generales de la clasificación que realiza el modelo acerca de posibles asteroides peligrosos.

2. Limpieza de Datos

La exploración y limpieza de los datos es de gran importancia en el análisis de de clasificación porque nos permite determinar las variables relevantes para el estudio y los datos adecuados para el mismo. Para ello, debemos tener en cuenta que este tipo de análisis solo es factible para variables cuantitativas, por lo que es necesario codificar las variables categóricas. Asimismo, se deben eliminar los valores atípicos de los datos, puesto que la mayoría de los algoritmos de clasificación, como Support Vector Machine (SVM), son muy susceptibles a estos.

En primer lugar, se eliminaron las variables que no son relevantes para el estudio o que no aportan información valiosa al análisis. En este caso solo se elimina la variable del nombre del asteroide ('full_name'). Luego, se realizó una inspección en el contenido de las variables, específicamente se determinó qué variables poseían valores nulos. Dentro de estas variables, se eliminaron aquellas que poseían más del 91 % de valores nulos, puesto que agregar valores a una gran cantidad de datos faltantes afectaría la distribución de la misma. Las variables eliminadas fueron las que están etiquetadas como 'G', 'extent', 'albedo', 'rot-per', 'GM', 'BV', 'UB', 'IR', 'spec_B' y 'specT'. Además de ello, las variables de la extensión del arco ('data_arc') y la magnitud absoluta ('H') también presentan valores nulos, pero en un porcentaje mucho menor (1 %). Para estos dos casos se decidió eliminar los valores faltontes, pues tenemos un gran conjunto de datos y esto no compromete en gran medida al tamaño de nuestra data de entrenamiento.

A continuación, se realizó una breve inspección de las características más importantes de las variables, tanto cuantitativas como cualitativas. Dentro de las variables cuantitativas tenemos:

- **Cuantitativas Discretas:** Extensión de arco ('data_arc'), número de observaciones ('n_obs_used').
- **Cuantitativas Continuas:** Eje semi-mayor ('a'), exentricidad ('e'), Inclinación con respecto al plano elíptico x-y ('i'), Longitud del nodo ascendente ('om'), periheluo ('w'), distancia al perihelio ('q'), distancia al afelio ('ad'), periodo orbital ('per_y'), magnitud absoluta ('H'), albedo geométrico ('albedo') y la distancia de intersección a la órbita mínima de la Tierra ('moid').
- **Categóricas ordinales:** Código de condición de la órbita ('condition_code'). Se entrega como número ordinal.
- **Categóricas nominales:** Objeto cercano a la Tierra ('neo') y Posiblemente peligroso ('pha'). Solo 2 clases en cada variable.

Como se mencionó anteriormente, el análisis de clasificación solo es factible para variables cuantitativas, por lo que las variables categóricas nominales se codificaron de tal manera que fueran '0' o '1'. Asimismo, se realizó un análisis de correlación entre las variables con el fin de determinar solo las variables que aporten diferente variabilidad al estudio. Bajo esa premisa, se eliminaron las variables etiquetadas como 'a', 'per_y' y 'moid', puesto que presentaban una correlación mayor a 0.95 con otras variables como 'ad' y 'q'.

Posteriormente, se escalaron los datos para poder tener un rango estandarizado entre todas las variables. Esto nos permite tener un mejor rendimiento en los algoritmos de clasificación. Una vez se tuvo los datos escalados, se graficó un diagrama de caja para visualizar los valores atípicos. A partir de ello se utilizó el rango intercuartil para eliminar dichos valores, correspondiente a los cuartiles del 1 % y el 99 %.

En base a la exploración y limpieza de datos realizada, se obtiene finalmente un conjunto de datos con 13 variables limpias y 98762 datos. A continuación, utilizaremos este conjunto de datos para realizar el análisis de clasificaicón.

3. Análisis de clasificación

3.1. *K-Nearest Neighbors*

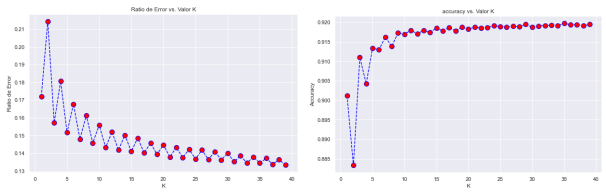
En términos sencillos K-Nearest-Neighbor es un algoritmo de Machine Learning supervisado, eso quiere decir que ya sabe las etiquetas que tiene que predecir, por lo que a partir de las etiquetas que establezca el usuario y las características de un conjunto de datos que lo rodea, este podrá predecir la etiqueta correspondiente, en otras palabras podrá clasificar la data. En este caso K representa la cantidad de puntos vecinos que vamos a considerar para poder clasificar los datos según N cantidad de grupos, que ya se conoce de antemano. Los pasos a seguir son los siguientes: 1) Calcular la distancia entre el dato a clasificar y el resto del dataset. 2) Seleccionar los 'k' elementos más cercanos, es decir, los que tengan la menor distancia de acuerdo a la función utilizada. 3) Seleccionar la etiqueta que corresponde, según la etiqueta que predomine entre los k puntos. Es como una votación, en la que se selecciona la etiqueta que mayor votos tenga. (No obstante, también se considera la probabilidad simple de pertenecer a esa clase y la distancia entre los puntos.

Es un algoritmo objetivamente sencillo de aprender y aplicar. Así que en poco tiempo la NASA podría implementarlo para evitar futuras catástrofes. Además nos permite clasificar resultados de manera efectiva. En este caso indicaría si un asteroide es peligroso o no a partir de las características de los K datos más cercanos. Sin embargo, utiliza cada fila del dataset para entrenar cada punto, por lo que podría implicar un mayor costo computacional y aumentaría el tiempo de ejecución. Por lo que, entre mayor sea el K y el dataset más se va a demorar. En este caso dado que contamos con muchas filas, probablemente se va a demorar mucho en ejecutar. KNN no es muy efectivo para procesar datos con muchas variables. En este caso las 13 variables tras la limpieza podría generarnos problemas

Para el proceso de selección de variables se utilizó 3 métodos: El Mapa de calor de Correlación de las variables, PCA y el método ExtraTreesClassifier. Su explicación a profundidad se encuentra en el *notebook* correspondiente, pero tras haberlos aplicado se decidió trabajar solo con las variables comunes entre estos métodos. Por lo tanto, se seleccionó: 'q', 'ad', 'n_obs_used', 'data_arc', 'i' y 'w'.

Después, los datos se dividieron en un 70 % para entrenamiento y un 30 % para pruebas. En este caso se decidió experimentar modificando únicamente el hiperparámetro de `n_neighbors`, puesto que era el más manipulable. Por ello, se probó calcular la precisión de modelos con distintos K, en ese sentido inicialmente se probó entre 1 a 315. Pero, como se evidenció en los resultados no había mucha diferencia conforme se aumentaba el K, así que se decidió probar con en un rango de 1 a 50. Para ello, se experimentó usando el método GridSearch, el cual nos permite hallar el valor óptimo de los hiperparámetros. En esa línea, solo se utilizó como el rango de 1 a 50 de `n_neighbors`. Así se obtuvo que el valor K óptimo es de 32.

Por otro lado, también se buscó visualizar el ratio de errores y precisión de otros parámetros. Así se obtuvieron los siguientes gráficos:



(a) Ratio de error vs KNN (b) Accuracy vs KNN

Figura 1: Ratio de error and Accuracy.s

Sin embargo, el resultado obtenido del AUC demostraba que el modelo no era tan eficiente como se esperaba. Así que para darle otro giro a la experimentación se volvió a calcular el GridSearch, pero esta vez utilizando las variables originales obtenidas en la limpieza de datos inicial. Así se obtuvo que el valor óptimo debería ser k = 14 y se volvió a calcular la curva de ROC:

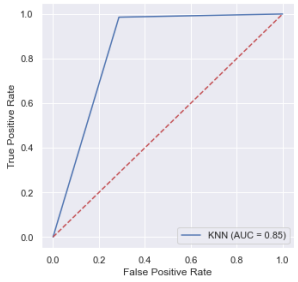


Figura 2: Curva ROC con el modelo KNN.

Así se obtuvo un AUC mayor demostrando que este modelo obtenido es mucho más eficiente y preciso para aplicar en otros datasets. Esta experimentación si bien teóricamente hablando contradice el principio que KNN trabaja mejor con menos variables, esto demuestra que la cantidad de variables utilizadas en este caso no excede los límites del algoritmo, aunque sí implican un mayor coste computacional a largo plazo.

En ese sentido, el modelo obtenido implica que cada asteroide en un conjunto de datos va a ser comparado con los 12 asteroides más cercanos y el modelo en un 85 % va a predecir correctamente si es peligroso o no. Uno debe entender que si bien es una probabilidad alta y un buen modelo, eso no quiere decir que el 100 % de las veces va a predecir correctamente. Por lo que, habría que tomar con precaución los resultados, puesto que 17 de cada 20 veces el resultado será preciso.

3.2. *Support Vector Machine*

El modelo de *Support Vector Machine* o SVM es uno de las técnicas de *Machine learning* más versátiles y poderosas, pues es capaz de realizar clasificaciones binarias (Verdadero o Falso) con alta precisión si se utiliza de forma adecuada. En general, este modelo se enfoca en encontrar la mejor frontera de decisión definida por vectores de soporte (líneas sólidas en Fig. 3). Estos vectores se contruyen en base a las instancias más alejadas de cada grupo. Como se muestra en la figura 3, la clasificación que genera SVM puede ser de forma lineal o no lineal (líneas punteadas), es decir, que la frontera que separa las clases binarias, puede ser una recta o una curva no lineal.

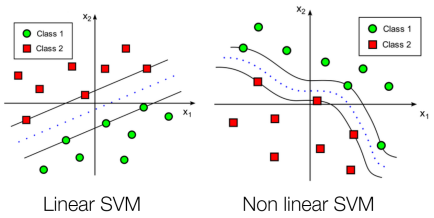


Figura 3: Clasificcación utilizando SVM lineal y no lineal.

Se decidió utilizar este modelo, pues presenta numerosas ventajas en el análisis de clasificación. Dentro de ellas, resaltamos las más relevantes en este estudio:

- El entrenamiento es sencillo y el módulo `scikit-learn` de Python nos lo facilita aún más.
- SVM no sufre de la maldición de la dimensionalidad. Sin embargo, es importante tener solo las variables más relevantes, de tal modo de encontrar la solución más simple (Principio de Parsimonia).
- SVM es menos propenso al sobreentrenamiento.

A pesar de que este modelo parece ser muy prometedor, también presenta algunas desventajas que debemos tener en cuenta, tales como:

- El entrenamiento y testeo es relativamente lento (costo cuadrático). Esto es perjudicial debido a la gran cantidad de datos que poseemos.
- Los *outliers* afectan considerablemente el rendimiento del modelo. Es por ello que estos se removieron en la limpieza de datos.
- Mayor problema: Seleccionar adecuadamente el kernel.

Como se menciona, uno de los mayores problemas de SVM es seleccionar el *kernel* adecuado. En este caso, se decide sintonizar el modelo con tres tipos de kernel distintos: *Lineal*, *RBF* y *Polinomial*. Para cada uno de los *kernels* se decidió sintonizar su hiperparámetro más importante. En el caso del kernel *lineal*, se sintonizó 'C', pues este le indica al modelo 'cuánto' debe evitar la clasificación errónea de cada clase; para el kernel *rbf*, se sintonizó el parámetro 'gamma' (γ), el cual es utilizado como medida de similaridad entre 2 puntos; en el caso de *Polinomial*, se sintoniza el parámetro 'degree' que indica el grado del polinomio.

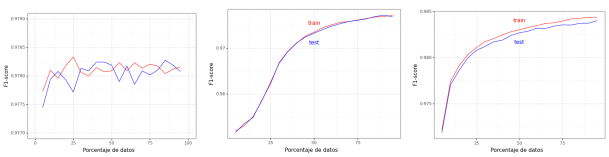
Debido a que la base de datos que poseemos presenta un sesgo con los valores de clase 'True', se decide utilizar la métrica 'f1-score' para evaluar cada modelo y escoger el mejor de ellos. Asimismo, cabe mencionar que utilizaremos el 70 % de los datos para entrenamiento y el resto para evaluación. En caso este método no sea suficiente para determinar el valor óptimo del hiperparámetro, se utilizará el método de K-fold cross validation (KCV). Con respecto a la selección de variables, se hicieron pruebas rápidas seleccionando algunas variables que se consideraron importantes, pero se obtuvieron resultados muy pobres por lo que se descartó. En ese sentido, se decidió utilizar todas las variables debido a que SVM nos lo permite y de acuerdo con Finio, de Harvard, todas las variables influyen en la predicción de si un asteroide puede ser peligroso o no.

Luego del entrenamiento, se determinó que los parámetros adecuados para cada kernel son $C = 0.2$, $\gamma = 0.001$ y grado $= 3$, respectivamente. En la figura 4 se observa la gráficas de entrenamiento con el 'f1-score' obtenido por cada modelo. Cabe mencionar que para el caso de KCV se utilizaron 10 *folds* y se graficó el valor de la media con la desviación estándar.

(a) 'C' con KCV (b) 'γ' con Train-test (c) Grado con KCV

Figura 4: Sintonización del hiperparámetro del modelo SVM con kernel lineal (a), rbf (b) y polinomial (c).

Posteriormente, se verificó que la curva de entrenamiento sea la adecuada para ambos modelos, y verificar que no exista *underfitting* u *overfitting*. En la Figura 5 se puede observar que el modelo mejor entrenado es el `SVM Polinomial`, pues la diferencia entre la curva de entrenamiento y testeo llega a un buen ajuste, por lo que se decide trabajar con este modelo para la clasificaión.

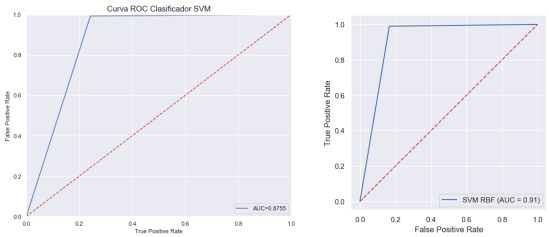


(a) SVM Lineal (b) SVM RBF (c) SVM Polinomial

Figura 5: Curva de entrenamiento y testeo.

Luego de decidir el modelo más adecuado para esta aplicación, procedemos a graficar la cruva ROC para analizar qué tan bien puede clasificar nuestro modelo. En la figura 6a se puede ver que el modelo presenta un AUC de 0.88, esto nos indica que el modelo tiene alta capacidad predicir correctamente verdaderos positivos y negativos, es decir, identifica correctamente si el asteroide es peligroso o no. Además de ello, la forma de la curva nos indica que predice muy pocos falsos negativos, es decir, que raramente precide que un asteroide no es peligroso cuando en realidad sí lo es, lo cual sería fatal para esta aplicación.

Si bien se ha entrenado de forma correcta distintos modelos, el modelo de SVM cuenta con más hiperparámetros que también pueden ser sintonizados y para lo cual se requeriría una sintonización más profunda. Afortunadamente `scikit-learn` nos proporciona funciones que facilitan esa búsqueda, tales como `GridSearchCV`. A esta función se le debe indicar los parámetros y sus respectivos valores a evaluar, de tal modo que la función entrena todos los modelos posibles y devuelve el modelo con mejor rendimiento. Bajo esa premisa, se entrenan diversos modelos con todos los tipos de kernel y se obtiene que el mejor modelo posee los parámetros $\gamma = 0.01$, $C = 1000$ y kernel `'rbf'`. Al igual que el modelo anterior se grafica la curva ROC (figura 6b) para evaluar al modelo y se encuentra que este posee una mayor capacidad que el modelo anterior para clasificar a los asteroides. Asimismo, la forma de la curva también nos indica que este modelo predice muy pocos falsos negativos, lo cual es adecuado para este modelo.



(a) Polinomial: grado$= 3$ (b) RBF: $\gamma = 0.01$ y $C = 1000$

Figura 6: Curvas ROC del modelo SVM polinomial y RBF.

En conlusión, el mejor modelo SVM para esta aplicación es el modelo SVM RBF con los siguientes hiperparámetros: $\gamma = 0.01$ y $C = 1000$.

3.3. Regresión Logística

El modelo de regresión logística es un modelo estadístico que permite predecir a partir de un muestreo de datos etiquetados. Este se adapta a una amplia variedad de situaciones en las que se obtienen una respuesta entre 0 y 1, basado en la probabilidad de cada registro. Además, es una de las técnicas más utilizadas dentro de la minería de datos debido a que permite encontrar relaciones y probabilidades entre datos que provienen de varios factores, considerada como una técnica de aprendizaje automático. En este aplicación nos ayudará a predecir si un asteroide es peligroso para la tierra basándonos en las etiquetas dadas por la variable 'Danger'.

En este caso en particular se tiene como ventaja la optimización de secuencia mínima que presenta una proyección de datos y es un algoritmo que responde a preguntas cerradas, lo cual ayuda en nuestro estudio de predecir el peligro de los asteroides porque solo hay 2 respuestas posibles: si es peligroso o no. Otras de las ventajas que presenta la regresión logística son el bajo consumo de recursos de la computadora, su fácil interpretación que será punto clave para determinar los resultados y se debe resaltar también su gran eficiencia. Lo cual es un beneficio que resalta respecto a otras técnicas.

Entre las desventajas tenemos la imposibilidad de resolver directamente problemas no lineales, es decir que aquellos problemas que tengan más de dos respuestas no serán viables. Otra desventaja es la dependencia que muestra en las características, ya que no es una herramienta útil para identificar las características más adecuadas, siendo necesario identificar estas mediante otros métodos. Estas desventajas en nuestro caso no llegan a afectar por la determinación cerrada de respuestas a la cuestionante.

Para crear el modelo se seleccionaron las variables que más colaboraban. Si bien se basó el análisis en la limpieza de datos, se eliminaron 3 columnas: 'condition_code', debido a que solo existen 3 opciones de respuesta a esta columna y predomina en un 97 % un dato, y en regresión logística es fundamental que se elijan correctamente las variables. Asimismo, 'neo' y 'pha', pues son variables que cuentan con la misma condición de tener un dato unánime en las muestras por lo cual no interviene en el análisis de los datos que deseamos para realizar el modelo. Entonces, para entrenar al modelo se utilizaron las variables: e, i, om, w, q, ad, data_arc, n_obs_used, H, albedo y nuestra variable a predcir es Danger.

Para la selección de las variables entre los modelos, se analizó primeramente las variables de `LogisticRegression()` que son:

- penalty: Penalización en el entrenamiento.
- fit_intercept: Considera en forma automática una columna con valor constante para el modelo
- C: valor correspondiente al INVERSO del parámetro de penalizacion
- random_state: para replicar un experimento.
- solver: método para resolver el modelo
- max_iter: máximo número de iteraciones.

Realizado el análisis se determinó poner en práctica el hiperparámetro *solver*, que son los métodos para resolver la predicción, este variable tiene las siguientes opciones que fueron probadas: *newton-cg*, *lbfgs*, *liblinear*, *sag* y *saga*. Cada método de resolución tiene restricciones con la penalización, por lo cual se fue cambiando la penalización para los modelos correspondientes.

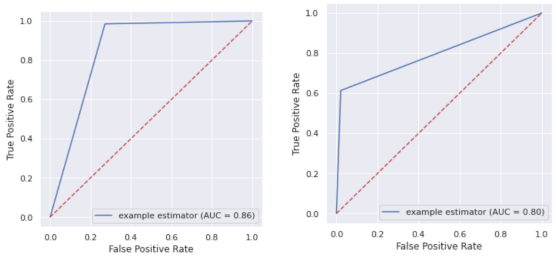
Otra de las variables que tuvo un cambio dentro de los modelos es *fit_intercept*, esta variable se puso como 'True' o 'False' para cada modelo, debido a que si nosotros tenemos una columna con valor constante para el modelo, el análisis cambia por lo cual hay variaciones con respecto al mismo modelo pero con otro *fit_intercept*, esto se ve reflejado en la curva de roc con el AUC. Asimismo, se decide dejar el máximo de iteraciones con 100 y C=1, para que se pueda analizar con una base similar todos los modelos. Es por lo cual la estructura de los modelos consiste en el cambio de *solver*, *penalty* y *fit_intercept*. Se entrenó satisfactoriamente los 10 modelos entre los cuales:

Tabla 1: Sintonización del modelo de Regresión Logísitica

# iteración	1	2	3	4	5	6	7	8	9	10
solver	ibfgs	ibfgs	newton-cg	newton-cg	liblinear	liblinear	sag	sag	saga	saga
penalty	l2	l2	none	none	l2	l2	l2	l2	l1	l1
fit_intercept	True	False	False	True	True	False	False	True	True	False
AUC	0.86	0.8	0.8	0.86	0.86	0.8	0.8	0.86	0.86	0.8

De los cuales podemos definir que en el modelo de regresión logística existen casi nulas diferencias entre los modelos evaluados en la variable solver, pero por otra parte se define que la variable *fit_intercept* es de vital definición al momento del entrenamiento y posterior determinado de acurracy. Si se determina el acurracy por la curva ROC (Figura 9), el mejor modelo tiene un AUC de 0.86, habiendo así 5 modelos con el mismo resultado se define como el mejor modelo entrenado a el primer modelo debido a que la definición de la regresión logística toma este modelo como base para la resolución, si es que nosotros no especificaramos otros hiperparámetros. Además, en el resumen del modelo y el entrenamiento se puede evidenciar una leve diferencia positiva en cuanto a los otros modelos. Pero al momento de evaluar un test acurracy cada modelo este sale igual a 0.66 que seria el promedio de los acurracy de los 10 modelos.

A partir del análisis realizado y los modelos entrenados, este modelo si puede generar información importante, se realizó un ejemplo en cada modelo, en el cual se daba los valores para que el modelo diga si con estas características el asteroide es peligroso o no y se puede probar con datos verdaderos y el resultado de la variable danger que es peligro sale como True (Verdadero) o False (Falso), por lo cual el modelo se puede utilizar y tendría un 86 % de probabilidad que esto ocurra. La explicación paso a paso se encuentra en el notebook de Regresión Logística, tanto como para las pruebas y la construcción del modelo.



(a) Reg. Logística: modelo 1 (b) Reg. Logística: modelo 10

Figura 7: Curvas ROC de los modelos de Regresión Logística.

3.4. Redes Neuronales

Una red neuronal es un modelo simple que intenta imitar el funcionamiento del cerebro, es decir, son redes de neuronas simuladas conectadas entre si. A pesar de que existen varios tipos la que más se suele utilizar es la red basada en capas. Las conexiones entre capas se componen de 3 partes principalmente, la primera es la capa de entrada, la segunda es la capa oculta que pueden ser una o varias y la ultima es la capa de salida, las unidades de cada capa se conectan entre ellas mediante ponderaciones.

La red aprende mediante datos que ya tienen la predicción que se quiere realizar y examina los registros individualmente para hacer ajustes a las ponderaciones. Al principio estas comienzan siendo aleatorias y a medida que el modelo se entrena las conexiones se autoajustan dado el aprendizaje que va obteniendo, este proceso finaliza cuando se logra alcanzar un máximo valor de aciertos, que no aumente con más iteraciones.

Una de las ventajas del modelo es que capta la información de manera óptima y obtiene resultados de alta precisión. Por otro lado, una de sus desventajas es que suele necesitar una gran cantidad de datos para un entrenamiento correcto.

Los pasos realizados para la implementación del código se encuentran detallados en el *notebook* del código. Para poder aplicar este modelo no existió la necesidad de crear variables para el entrenamiento sino más bien se tuvo que realizar un procesamiento de tres de las variables preprocesadas, las cuales eran categóricas y se transformaron mediante el 'onehotencoded', este proceso también se debió realizar posteriormente por las variables al momento de usar el modelo en la nueva data para realizar la predicción.

Finalmente el resultado muestra que el modelo realiza predicciones con un 97 % de asertividad lo que lo hace en si un buen modelo y una ventaja al mismo tiempo al momento de ser utilizado, por otra parte el 3 % restante contiene en parte los falsos negativos que presentan un numero impor-

tante, como se puede ver en la matriz de confusión y que representan una desventaja en este caso.

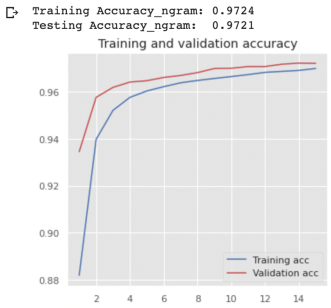


Figura 8: Precisión del entrenamiento y validación.

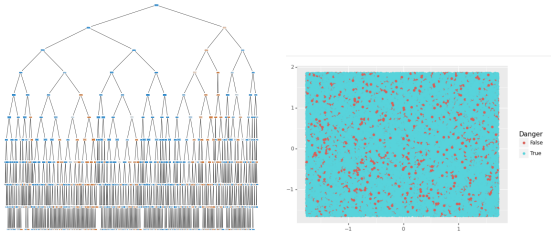
3.5. Árbolses de Decisión

En primer lugar, un árbol de decisión es un mapa de los posibles resultados de una serie de decisiones relacionadas. Permite que un individuo o una organización comparen posibles acciones entre sí según condiciones, obteniendo probabilidades y predicciones. Básicamente se pueden usar para dirigir un intercambio de ideas informal o trazar un algoritmo que anticipe matemáticamente la mejor opción frente a un objetivo particular.

Respecto a la estructura de un árbol de decisión, por lo general, comienza con un único nodo y luego se ramifica en resultados posibles separando bajo algún criterio o condición. Cada uno de esos resultados crea nodos adicionales, que se ramifican en otras posibilidades y es por estos resultados que este algoritmo al visualizarlo, le da una forma similar a la de un árbol. Cabe mencionar que los árboles de decisión también se pueden dibujar con símbolos de diagramas de flujo, que a algunas personas les parecen más fáciles de leer y comprender.

Debido a sus características se decide utilizar un modelo de árboles de decisión. En primer lugar se definieron las variables 'e', 'i', 'om', 'w', 'q', 'ad', 'data_arc', 'n_obs_used', 'H', 'albedo', 'neo', 'pha'y por último 'condicion_code', cuyas deficiones se realizaron en la seccíon 2. Todas estas variables se definieron como las 'predictoras' en base a nuestro atributo en estudio, que es 'Danger', es por esto mismo que 'Danger' fue clasificada como la variable dependiente para predecir si es peligroso o no bajo los atributos mencionados. Posterior a esto, se ejecutó el código con valor de profundidad (Max_depth) igual a 10 (hiperparámetro seleccionado). El resultado del árbol con esta profundidad (figura 9a) aparenta ser adecuado dada la cantidad de nodos.

A modo de complemento, en la figura 9b se realizó un gráfico que clasifica los puntos según a si es o no peligroso asignándole un color. Se puede observar que hay una predominancia evidente por el color azul, lo que nos lleva a intuir de inmediato que el modelo ya apunta a una predicción sesgada hacia que hay mayor probabilidad de encontrar un asteroide peligroso, es decir, con un diámetro sobre 2km que bajo ese valor.



(a) Árbol de Decición (b) Gráfico de clasificación

Figura 9: Arbol de decisión y gráfico de clasificación

Por último a modo de medidor de rendimiento que se profundizará en el siguiente apartado, se obtuvo la curva ROC teniendo un valor de AUC igual a 0.9 (Figura 10, lo que nos dice que el modelo tiene cierta solidez en la predicción. Con respecto al hiperparámetro, se designó la profundidad del modelo como variable para luego determinar cómo medir el rendimiento en la curva ROC el indicador AUC y comparar sus valores bajo el criterio del valor más cercano a 1. Cabe mencionar que previo a esto se decidió estudiar una iteración fuera de las 10 solicitadas para comparar el indicador AUC seleccionando exclusivamente las variables "i", "q", "ad" y "albedo" ya que en mi intuición, podrían ser las variables más relevantes para poder predecir si es o no un asteroide peligroso.

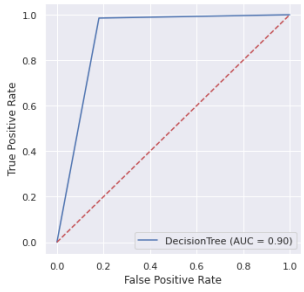


Figura 10: Curva ROC del Árbol de Decisión

Como resultado se obtuvo un valor en el indicador AUC de 0.59 en comparación a los 0.9 obtenidos con todas las variables, por lo que descartamos esa posibilidad y se procede con las 10 iteraciones para definir si la profundidad definida fue la más apropiada o no. En la Tabla 2 se puede apreciar las iteraciones realizadas para sintonizar el hiperparámetro de `max_depth`

Tabla 2: Sintonización del hiperparámetro `max_depth`

# iteracion	1	2	3	4	5	6	7	8	9	10
max_depth	3	4	1	2	15	20	5	25	12	100
AUC	0.66	0.85	0.68	0.66	0.89	0.9	0.82	0.89	0.9	0.89

3.6. Evaluación de modelos

Luego de analizar cada modelo por separado, procedemos a realizar una comparación entre modelos. Para ello, nos apoyamos de la métrica de la curva ROC. Esta métrica nos permite analizar el rendimiento de cada modelo de forma gráfica, además de indicarnos qué tan bien predice los verdaderos positivos y nos indica si el modelo es propenso a predecir falsos negativos. En la Figura 11 se muestra la curva ROC de cada modelo en una sola gráfica

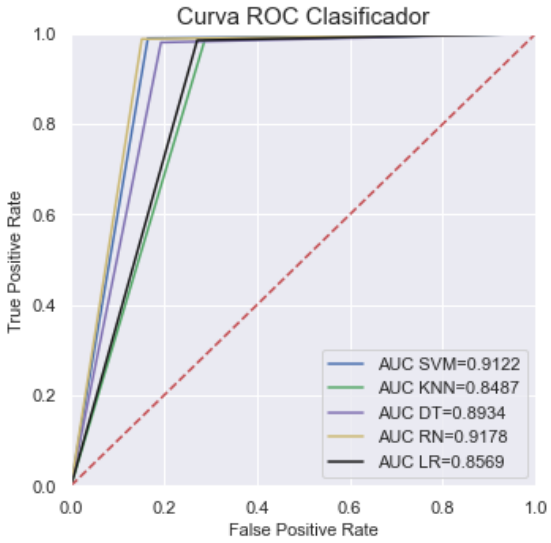


Figura 11: Curvas ROC de todos los modelos de clasificación

A partir de la figura 11 se puede concluir que el modelo con mejor rendimiento es el de Redes Neuronales. Este modelo presenta un área bajo la curva de 0.9178 y nos indica que el modelo tiene alta capacidad predicir correctamente verdaderos positivos y negativos, es decir, identifica correctamente si el asteroide es peligroso o no. Además, la curva nos indica que el modelo predice raraente que un asteroide no es peligroso cuando en realidad sí lo es (falso negativo), lo cual sería fatal para esta aplicación.

4. Conclusiones

En primer lugar, podemos concluir que los modelos de clasificación presentan un mayor rendimientos cuando son entrenados con todas las variables limpias que cuando solo se seleccionan algunas de ellas. Esto tiene sentido con la teoría, pues de acuerdo con Finio cada variable representa una característica importante del asteroide. Sin embargo, cabe mencionar que, de acuerdo con el análisis hecho en la limpieza de datos y 'ExtraTreesClassifier' las variables que más influyen en esta clasificación son 'H', la magnitud absoluta del asteroide; 'albedo', el albedo geométrico; y 'q' y 'ad', la distancia al perihelio y afelio .

Con respecto al rendimiento de los modelos, se utilizó la métrica de la curva ROC para determinar el modelo que realiza mejores predicciones. En base a ello, se obtuvo que el modelo entrenado por Redes Neuronales es el que mejor predice si un asteroide es peligroso o no. Este modelo presente un área bajo la curva de 0.9184 en la curva ROC, lo cual indica que existe un 91.84 % de probabilidad de que la predicción realizada a un asteroide sea más correcta que si se realizara de forma aleatoria. Este porcentaje indica que el modelo realiza predicciones muy buenas, pues se encuentra en un rango entre 91 % y 97 %.