Clasificación Ordinal y Métodos Basados en ML

Armando Salinas Lorenzana, Yulissa Guadalupe Crockwell Palomares.
Centro de Investigación en Matemáticas. Unidad Monterrey
Email: armando.salinas@cimat.mx, yulissa.crockwell@cimat.mx

***Resumen*—En este trabajo aplicamos distintas metodologías para la implementación de datos con clases ordinales en los métodos de clasificación vistos en clase, se estudia un método simple para clasificación aplicado a Árboles de decisión y Random Forest, Boosting de división fija, Regresión Logística Ordinal y Redes neuronales para datos ordinales. Se muestra la metodología con el cual se considera el carácter ordinal de los datos de etiqueta lo métodos implementado y sobre esto se mide su eficiencia con las métricas Mean Absolute Error y Quadratically Weighted Kappa. Posteriormente se comparan los resultados de la clasificación obtenidas de los modelos ordinales con los modelos en su forma estándar (no ordinales). Por último se comparan todos los modelos ordinales para determinar cuál fue el mejor modelo para el conjunto de datos entrenado.**

I. Introducción

Los problemas de clasificación ordinal son aquellos problemas de aprendizaje automático donde el objetivo es clasificar patrones usando una escala categórica que muestra un orden o jerarquía entre las etiquetas. Comúnmente los métodos de aprendizaje automático para problemas de clasificación asumen que los valores de las clases no están ordenados. Sin embargo en muchas aplicaciones prácticas, los valores de las clases sí exhiben un orden natural, por ejemplo, al evaluar la calidad de un servicio, comúnmente respondemos encuestas de satisfacción donde las categorías muestran un orden (Satisfactorio $>$ Aceptable $>$ Malo $>$ Muy malo), así como este ejemplo hay muchos problemas que presentan una clase ordinal, por lo que el estudio y la investigación en este tema ha sido de interés y ha incrementado con el tiempo habiendo distintas propuestas de algoritmos y metodologías para tratar el problema de los datos ordinales al considerar el orden o jerarquía entre las clases [1], [2].

I-A. Propupesta de taxonomía para regresión ordinal

En [3] podemos ver una propuesta de taxonomía para los distintos métodos de regresión ordinal, el cual se clasifican en tres grandes grupos: enfoques ingenuos, descomposiciones binarias ordinales y modelos de umbrales.

I-A1. Enfoques ingenuos: Los problemas de regresión ordinal pueden simplificarse fácilmente en otros problemas estándar haciendo algunas suposiciones. Estos métodos pueden ser muy competitivos dado que, aunque estas suposiciones pueden no cumplirse, heredan el rendimiento de modelos muy bien afinados. Estos a su vez se dividen en Regresión, Clasificación Nominal y Clasificación de costos sensibles.

I-A2. Descomposiciones binarias ordinales: Este grupo incluye todos aquellos métodos que se basan en descomponer la variable objetivo ordinal en varias binarias, las cuales son luego estimadas por uno o múltiples modelos (OneVsAll y OneVsOne). Este método considera la descomposición binaria del problema, por lo que al final se entrenan más de un modelo con la descomposición binaria hecha. Además de definir cómo descomponer el problema, también se define una regla para predecir nuevos patrones una vez que se obtienen los valores de decisión. Estas a su vez se dividen en Modelos múltiples y Modelo único con múltiples salidas.

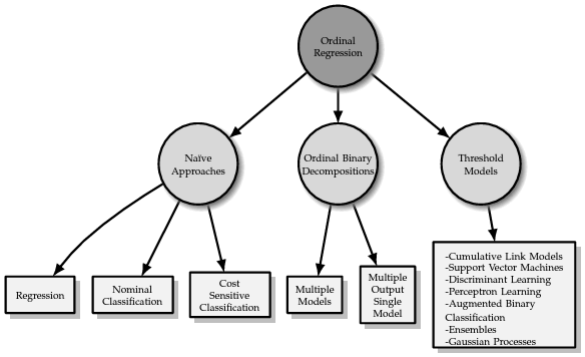

Figura 1. Propuesta de taxonomía para regresión ordinal (figura tomada de [3].

I-A3. Modelos de umbrales: En este grupo de métodos, se supone una variable latente que subyace a la respuesta ordinal de los datos, estos métodos se basan en esa suposición y se conocen como modelos de umbral. Estos son los enfoques más populares para problemas ordinales y de clasificación. Los modelos de umbral pueden verse como una extensión de los modelos de regresión ingenuos. La principal diferencia entre estos dos enfoques es que las distancias entre las diferentes clases no están definidas a priori para los modelos de umbral, sino que se estiman durante el proceso de aprendizaje. Aunque también están relacionados con los enfoques de descomposición binaria ordinal (de un solo modelo), la principal diferencia es que los modelos de umbral se basan en un único vector de mapeo con múltiples umbrales, uno para cada clase.

II. Objetivos

II-A. Objetivo general

Implementar una metodología para cada que modelo de clasificación que considere el carácter ordinal de los datos en la variable objetivo.

II-B. Objetivos particulares

- Encontrar una base de datos con carácter ordinal.

- Encontrar las metodologías que se adecuen a los modelos.
- Implementar la metodología.
- Comparar resultados con los modelos no ordinales.
- Comparar los modelos ordinales entre sí para el conjunto de datos entrenado.

III. Métricas de evaluación del desempeño

Para evaluar el rendimiento de nuestros modelos de datos ordinarios y modelos estándar, realizaremos la evaluación de las siguientes medidas de desempeño, el error absoluto medio (MAE, por sus siglas en ingles) y el kappa ponderado cuadrático, todo esto para determinar la eficiencia y utilidad de cada modelo.

III-A. Error medio absoluto

El Error medio absoluto, definido por la suma de los valores absolutos de las diferencias entre los valores reales y valores de predicción del modelo entre el número total de datos en nuestro conjunto de datos [4]. Además, esta medida se caracteriza por darle menor peso a los datos atípicos:

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

Donde y_i representa los valores reales, $\hat{y}_i$ representa los valores predichos por el modelo, n es el número total de datos y N es el número total de datos en el conjunto de datos.

III-B. Kappa ponderado cuadrático

El kappa ponderado cuadrático, es una métrica de desempeño para modelos de clasificación que considera la la distribución de las clases en el conjunto de datos. Esta definida como [5],

$$\kappa_w = 1 - \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}O_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}E_{ij}}$$

Donde:

- O es la matriz de frecuencias observadas
- E es la matriz de frecuencias esperadas, calculada a partir de las distribuciones marginales de las clasificaciones.
- w_{ij} es el peso asignado a la discrepancia entre la categoría i y la categoría j.

Definiendo los pesos w_{ij} como:

$$w_{ij} = \frac{(i-j)^2}{(k-1)^2}$$

donde:

- i y j son las categorías.
- k número total de categorías.

Esta métrica varía entre 0 y 1, donde 0 nos indica una mayor precisión entre los valores reales y los valores de predicción de nuestro modelos. Por otro lado el 1 nos hace mención a una discrepancia entre nuestros valores.

IV. Metodología

En esta sección presentamos la metodología para cada uno de los modelos usados para clasificar. Dado que existen distintos grupos de métodos, hay distintas implementaciones para cada clasificador, sin embargo a menudo que vayamos avanzando en este reporte se irán mencionando los métodos usados y en base en qué trabajos reportados en la literatura, nos guiamos.

IV-A. Regresión Logística Ordinal

Se comentan dos modelos en particular [6], el de probabilidades proporcionales y el de riesgos proporcionales, ambas son probablemente los más útiles en la práctica debido a la simplicidad de su interpretación. Se ha demostrado que estos modelos lineales son extensiones multivariadas de los modelos lineales generalizados.

IV-A1. Modelo de probabilidades proporcionales: Una variable de respuesta puede tomar valores en categorías ordenadas (por ejemplo, bajo, medio, alto). Las covariables o factores explicativos son otras variables que se cree influyen en la variable de respuesta. Supongamos que tenemos k categorías de la variable de respuesta, supongamos que las categorías ordenadas tienen las siguientes probabilidades $\pi_1(x), \pi_2(x), ..., \pi_k(x)$, donde x son las covariables. Sea Y la variable de respuesta que toma valores de $1, ..., k$, con las probabilidades mencionadas anteriormente, y sea $\kappa_j(x)$ la probabilidad de que $Y \leq j$. Entonces la variable de probabilidades proporcionales nos dice que

$$\kappa_j(x) = \kappa_j exp(-\beta^T x) \quad (1 \leq j < k), \tag{2}$$

donde β es un vector de parámetros desconocidos. La proporción de las probabilidades corresponde

$$\frac{\kappa_j(x_1)}{\kappa_j(x_2)} = \exp(\beta^T(x_2 - x_1)), \quad (1 \leq j < k)$$

podemos notar que no depende de j y que únicamente depende la diferencia de las covariables $x_2 - x_1$. Dado que las probabilidades para que el evento $Y \leq j$ ocurra, es el cociente $\gamma_j(x)/1 - \gamma_j(x)$, donde $\gamma_j(x) = \pi_1(x) + ... + \pi_j(x)$, el modelo de probabilidades proporcionales es idéntico al modelo logístico lineal.

$$log[\gamma_j(x)/(1-\gamma_j(x))] = \theta_j - \beta^T x \quad (1 \leq j < k) \tag{3}$$

donde $\theta_j = logk_j$. Podemos notaer en particular que cuando hay solo dos categorías de respuesta, (3) es equivalente al modelo logístico lineal usual para datos binarios y en este caso particular también es equivalente a un modelo log-lineal. Sin embargo, en general, cuando el número de categorías excede 2, el modelo logístico lineal (3) no corresponde a una estructura log-lineal.

IV-A2. Modelo de riesgos proporcionales: La función de riesgo o función de riesgo instantáneo $\lambda(t;x)$, es de gran importancia en el análisis de datos de supervivencia, se define como la probabilidad instantánea de falla en el tiempo t condicional a la supervivencia hasta el tiempo t. Para un individuo con covariable x, el modelo de riesgos proporcionales es

$$\lambda(t;x) = \lambda_0(t)exp(-\beta^T x) \tag{4}$$

donde $\lambda_0(t)$ es la función de hazard cuando $x = 0$ y β es un vector de parámetros desconocidos. Simplemente observamos que la función de supervivencia $S(t;x)$ es la probabilidad de sobrevivir más allá del tiempo t dado la covariable x, satisface

$$-\log\{S(t;x)\} = \Lambda_0(t)\exp(-\beta^T x), \tag{5}$$

donde $\Lambda_0(t) = \int_0^t \lambda_0(s)ds$. Por tanto si consideramos dos individuos con covariables x_1 y x_2 respectivamente, entonces la función de supervivencia debe satisfacer

$$\log\{S(t;x)\}/\log\{S(t;x)\} = \exp\{\beta^T(x_2 - x_1)\}. \tag{6}$$

En otras palabras, el cociente de las funciones de supervivencia logarítmicas, al igual que el cociente de las funciones de riesgo, depende únicamente de la diferencia entre los valores de las covariables $x_2 - x_1$ y es constante para todos los t. Para datos discretos, el modelo de riesgos proporcionales (6) se convierte en

$$-\log\{1 - \gamma_j(x)\} = \exp(\theta_j - \beta^T x), \tag{7}$$

donde $1 - \gamma_j(x)$ es la probabilidad complementaria o la probabilidad de "supervivencia" más allá de la categoría j dadas los valores de las covariables x. Para obtener la estructura lineal apropiada análoga al modelo logístico lineal, escribimos (7) en la forma más conveniente

$$\log[-\log\{1 - \gamma_j(x)\}] = \theta_j - \beta^T x, \tag{8}$$

la transformación a linealidad se denomina transformación log-log complementaria. En particular cuando solo hay dos grupos, de modo que la covariables toman solo dos valores, x_1 y x_2, la diferencia entre los log-log complementarios correspondientes es la constante $\beta^T(x_2 - x_1)$ y es independiente de la categoría involucrada. En este sentido, las propiedades del modelo log-log complementario son paralelas a las del modelo de probabilidades proporcionales o modelo logit.

Los modelos de proporción de probabilidades y de proporción de riesgos tienen la misma forma general, por lo que podemos expresarlo de la siguiente forma

$$link\{\gamma_j(x)\} = \theta_j - \beta^T x, \tag{9}$$

donde "link" es la función logit o log-log complementaria. Cualquier otra función monótona creciente que mapee el intervalo unitario $(0, 1)$ en $(-\infty, \infty)$ puede ser utilizada como función de enlace. En particular, la función normal inversa $\Phi^{-1}(\gamma)$, la función inversa de Cauchy, $\arctan\{\pi(\gamma - 0.05)\}$, y la función log-log, $\log(-\log(\gamma))$ son posibles candidatas, aunque la interpretación de los parámetros no es generalmente tan directa como con los modelos de proporción de probabilidades o de proporción de riesgos.

Para obtener los parámetros del modelo (9). se calculan las estimaciones de máxima verosimilitud, este calculo se simplifica considerablemente al notar que la estructura sistemática lineal (9) junto con la variación multinomial comprenden un modelo lineal generalizado multivariado. Los modelos lineales generalizados univariados han sido discutidos por Neider y Wedderburn, quienes han demostrado que las estimaciones de los parámetros pueden obtenerse mediante mínimos cuadrados ponderados iterativos.

Por nuestra parte encontrar los parámetros, pudimos implementar una función dentro de la librería *statsmodels*, que se encarga de encontrar estos parámetros, facilitando el uso y la implementación de dicho modelo.

IV-B. Adaboost y boosting de división fija

Para realizar la implementación ordinal de las etiquetas para Adaboost, nos basamos de [7], se distinguen dos métodos para realizar la implementación: el boosting de división fija, en el cual los métodos de agregación se aplican a una dicotomización fija de las categorías de respuesta (se hace la divisón binaria de las etiquetas), y métodos en los que la agregación ordinal se realiza en cada paso de iteración, éste último método considera el carácter ordinal dentro del algoritmo de Adaboost. Para la metodología que implementamos nos basamos en el método de división fija.

IV-B1. Boosting de división fija: El procedimiento de clasificación se divide en dos etapas de la siguiente manera. En la primera etapa, se utilizan técnicas para dividir las categorías de respuesta. En la segunda etapa, se combinan los clasificadores binarios resultantes y dado un criterio se consideran todos los clasificadores para determinar la clase a la que pertenecen los objetos.

Primeramente asumimos que cada objeto pertenece a una de las k clases. Por tanto consideremos un vector x que contiene p características asociadas al objeto. Por tanto un clasificador se entrena con datos del cual haya aprendido, de esta forma denotamos el entrenamiento L como $L = \{(Y_i, x_i), i = 1, ..., n_L\}$ denotado como el entrenamiento de aprendizaje donde $Y_i \in \{1, ..., k\}$ denota la clase al cual $x_i' = (x_{i1}, ..., x_{ip})$ son asociadas a las covariables. Por tanto un clasificador basado en la partición del espacio X de las covariables tiene la forma

$$C(., L) : X \longrightarrow \{1, ..., k\} \tag{10}$$

$$x \longrightarrow C(x, L), \tag{11}$$

Ahora, para la primera etapa la reducción a problemas binarios se realiza considerando divisiones dentro de las categorías de respuesta. Esto funciona definiendo

$$Y^{(r)} = \begin{cases} 1 & Y \in \{1, ..., r\} \\ 2 & Y \in \{r+1, ..., k\} \end{cases}$$

para $r = 1, ..., k-1$. De esta forma hacemos la división binaria de las clases, por lo que tendremos $k-1$ divisiones. Cabe mencionar que esta partición solo tiene sentido si las clases tienen un orden determinado. Por tanto definimos $C^{(r)}(., L)$ como el clasificador binario para las clases definidas por $Y^{(r)}$, para un r tal que $\{1, ..., r\}, \{r+1, ..., k\}$. Por tanto para un r fijo se obtiene la clase predicha para cualquier observación x por

$$C_{agg}^{(r)} = \underset{j}{\operatorname{argmax}} \sum_{m=1}^{M} c_m^{(r)} I(C^{(r)}(x, L_m^{(r)}) = j). \tag{12}$$

El clasificador agregado de primera etapa $C_{agg}^{(r)}$ ha sido diseñado para una división fija en r. Donde cada clasificador $C_{agg}^{(r)}$ nos revuelve un vector binario $(y_i, ..., y_k)$ donde

$$y_j = \begin{cases} 1 & Y = j \\ 0 & \text{otro caso.} \end{cases}$$

Ahora, al explotar el orden de las categorías, la combinación de los clasificadores agregados $C_{agg}^{(1)}, ..., C_{agg}^{(k-1)}$ se basa en la agregación de la segunda etapa. Por tanto la segunda etapa consiste en realizar una transformación, por tanto sea $\hat{y}_1^{(r)}, ..., \hat{y}_k^{(r)}$ el resultado de los clasificadores. Realizamos la transformación de esta secuencia de la siguiente manera

- Para $C_{agg}^{(r)} = 1$ corresponde a $\hat{Y} \in \{1, ..., r\}$ se tiene

$$\hat{y}_1^{(r)}(x) = ... = \hat{y}_r^{(r)}(x) = \frac{1}{r}, \quad \hat{y}_{r+1}^{(r)}(x) = ... = \hat{y}_k^{(r)}(x) = 0.$$

- Para $C_{agg}^{(r)} = 2$ corresponde a $\hat{Y} \in \{r+1, ..., k\}$ se tiene

$$\hat{y}_1^{(r)}(x) = ... = \hat{y}_r^{(r)}(x) = 0, \quad \hat{y}_{r+1}^{(r)}(x) = ... = \hat{y}_k^{(r)}(x) = \frac{1}{k-r}.$$

De esta forma el clasificador $C_{agg}^{(r)}$ binario produce una secuencia $\frac{1}{r}(1, 1, ..., 1, 0, 0, ..., 0)$ o $\frac{1}{(k-r)}(0, 0, ..., 0, 1, 1, ..., 1)$ donde los cambios son de 1 a 0 o de 0 a 1 después del rth componente. Dividimos las secuencias por r o $k-r$, respectivamente, para tener en cuenta el diferente número de categorías dentro de cada dicotomización. Por último el clasificador final viene dado por

$$C_{agg}(x) = \underset{j}{\operatorname{argmax}} \sum_{r=1}^{k-1} \hat{y}_j^{(r)}(x). \tag{13}$$

Este último clasificador, determina la clase de pertenencia de la observación x, considerando a todos los $k-1$ clasificadores. Para esta metodología podemos considerar como clasificador binario cualquier modelo, de esta forma podríamos definir $C_{agg}^{(r)}$ siendo uno de los dos clasificadores. Para nuestro trabajo implementamos Adaboost.

IV-C. Redes neuronales

El enfoque de redes neuronales para datos ordinales se realizó de acuerdo a la metodología empleada en [8]. Supongamos que D representa un conjunto de datos de regresión ordinal que consiste en n puntos de datos (x, y), donde $x \in \mathbb{R}^d$ es un vector de características de entrada y y es su categoría ordinal de un conjunto finito Y. Asumimos que $Y = 1, 2, ..., K$ con "$<$" como relación de orden. Para una red neuronal de clasificación estándar que no considera el orden de las categorías, el objetivo de la red es predecir la probabilidad de que un punto de datos x pertenezca a una categoría k (es decir, $y = k$). La entrada es x y el objetivo de codificar la categoría k es un vector $t = (0, ..., 0, 1, 0, ..., 0)$, donde solo el elemento t_k se establece en 1 y todos los demás en 0. El objetivo es aprender una función para mapear el vector de entrada x a un vector de distribución de probabilidad $o = (o_1, o_2, ..., o_k, ..., o_K)$, donde o_k está más cerca de 1 y los otros elementos están cerca de cero, sujeto a la restricción $\sum_{i=1}^{K} o_i = 1$.

Por el contrario, el enfoque de esta metodología, es que la red neuronal considera el orden de las categorías. Si un punto de datos x pertenece a la categoría k, se clasifica automáticamente en categorías de orden inferior $(1, 2, ..., k-1)$ también. Entonces, el vector objetivo de x es $t = (1, 1, ..., 1, 0, 0, 0)$, donde t_i ($1 \leq i \leq k$) se establece en 1 y los otros elementos son cero. Así, el objetivo es aprender una función para mapear el vector de entrada x a un vector de probabilidad $o = (o_1, o_2, ..., o_k, ..., o_K)$, donde o_i ($i \leq k$) está cerca de 1 y o_i ($i \geq k$) está cerca de 0. $\sum_{i=1}^{K} o_i$ es la estimación del número de categorías (es decir, k) a las que pertenece x, en lugar de 1.

Una vez que tenemos claro el objetivo de la nueva red neuronal podemos utilizar casi exactamente una red neuronal convencional para aprender las relaciones ordinales de D. Creamos una red neuronal con entrada d, es decir, el mismo número de características de x, y consideramos k nodos en la capa de salida, es decir, el número de clases. Pueden considerarse una o más capas ocultas. Al igual que una red neuronal estándar para clasificación, los nodos de entrada están completamente conectados con los nodos ocultos, que a su vez están completamente conectados con los nodos de salida. De esta forma establecemos la estructura de la red neuronal. En cuanto a la función de transferencia de los nodos ocultos puede ser una función lineal, una función sigmoide y una función tangente hiperbólica, para este trabajo se considero la función sigmoide. La única diferencia con respecto a las redes neuronales tradicionales consiste en la función de salida, comúnmente se utiliza la función softmax

$$\frac{e^{-z_i}}{\sum_{i=1}^{k} e^{-z_i}}$$

satisfaciendo la restricción de que la suma de las salidas $\sum_{i=1}^{k} 0_i$ es 1, z_i es la entrada neta al nodo de salida O_i.

Por el contrario, esta nueva red neuronal debe considerar una función de salida sigmoide

$$\frac{1}{1+e^{-z_i}}. \tag{14}$$

El nodo de salida O_i se utiliza para estimar la portabilidad o_i de que un punto de datos pertenezca a la categoría i de forma independiente sin estar sujeto a la normalización como lo hacen las redes neuronales convencionales. Por otra parte, el vector objetivo tiene la forma (1,1,1,0,0,0), como en el caso de Adaboost y Bagging, así, los primero k elementos son 1 y los demás elementos son 0. De esta forma le damos el carácter ordinal y el valor objetivo de los nodos de salida son $O_i(i \leq k)$ en 1 y $O_i(i > k)$ en 0. El vector objetivo de este estilo enseñan a la red a ajustar sus pesos considerando el orden ordinal de los datos.
Para el entrenamiento de la red neuronal para datos ordinales se procede de manera similar a las redes neuronales convencionales, la función de costo puede ser al entropía relativa o el error cuadrático entre el vector objetivo y el vector de salida. para la entropía relativa, la función de costo es

$$f_c = \sum_{i=1}^{k}(t_i \log o_i + (1-t_i)\log(1-o_i)). \tag{15}$$

Para la función de error cuadrático la función de costo de entropia relativa es

$$f_c = \sum_{i=1}^{k}(t_i - o_i)^2. \tag{16}$$

Ambas funciones de costos dan resultados similares, en este trabajo utilizamos el error cuadrático como función de costos.

Por otra parte si analizamos la propagación del error hacia atrás, podemos observar que la derivada de f_t del nodo de salida O_i es

$$\frac{\partial f_t}{\partial z_i} = \frac{e^{-z_i}}{(1+e^{-z_i})} = \frac{1}{1+e^{-z_i}}\left(1 - \frac{1}{1+e^{-z_i}}\right) = o_i(1-o_i). \tag{17}$$

Por tanto el error neto propagado al nodo de salida O_i para la función de consto de entropía relativa es

$$\frac{\partial f_c}{\partial f_t}\frac{\partial o_i}{\partial z_i} = (t_i - o_i)o_i(1-o_i). \tag{18}$$

Para la función de costos de error cuadrático es

$$\frac{\partial f_c}{\partial o_i}\frac{\partial f_t}{\partial z_i} = -2(t_i - o_i)o_i(1-o_i) = -2o_i(t_i - o_i)(1-o_i). \tag{19}$$

Los errores netos se propagan a través de las redes neuronales para ajustar los pesos utilizando el descenso de gradiente, como lo hacen las redes neuronales tradicionales.

Para realizar las predicciones fijamos un umbral T, en este caso $T = 0.5$, dado que en la salida tenemos un vector de dimensión igual al número de características. Se recorre el arreglo de salida y se detiene cuando la salida de un nodo es menor que el umbral predefinido T, entonces el índice k del último nodo O_k cuya salida es mayor que T es la categoría predicha del punto de datos.

IV-D. Random Forest y Árboles de decisión

El algoritmo de clasificación estándar del random forest, trata las clases ordinales como normales, teniendo una perdida de información de los datos valiosa para aportar mayor precisión al problema. Es por esto que para abordar el random forest con datos ordinarios nos basamos en la metodología empleada en [9]. Este problema transforma los datos ordinales en datos binarios.Supongamos que tenemos un conjuntos de datos iniciales (X, Y) donde X es un conjunto de instancias y y un vector de clases ordinales, donde $y_i \in \{1, 2, \ldots, k\}$ para $i = 1, 2, \ldots, n$. Se transforma cada atributo ordinal y en $k-1$ atributos binarios $y_1, y_2, \ldots, y_{k-1}$. Cada atributo binario y_j representa la prueba de si el valor de la instancia y_i es mayor que un valor umbral específico t_j. Lo definimos como :

$$y_{ij} = \begin{cases} 1 & \text{si } y_i > t_j \\ 0 & \text{en caso contrario} \end{cases}$$

donde $t_j = j$ para $j = 1, 2, \ldots, k-1$.
De esto, para crear el conjutno de datos binarios, se derivan $k-1$ nuevos conjuntos de datos binarios (X_j, Y_j) para $j = 1, 2, \ldots, k-1$. Cada conjunto de datos binario X_j contiene las mismas instancias que el conjunto de datos original X, pero las etiquetas de clase se transforman en los atributos binarios y_{ij} correspondientes.
De esto, para entrenar el modelo, aplicamos random forest a los datos ya transformados, es decir, a cada uno de los $k-1$ conjuntos de datos binarios (X_j, y_j).
Para predicir un x_{new} se procesa a través de cada uno de los $k-1$ modelos de random forest entrenados V_k.De esto, se predice la probabilidad $P(V_i)$ de que la instancia pertenezca a la clase "mayor que el umbral.[en] su conjunto de datos binario correspondiente. Posteriormente se estiman las probabilidades para las clases ordinales V_i utilizando las siguientes probabilidades:

- Primera y última clases:

$$P(V_1) = 1 - P(Class > V_1),$$
$$P(V_k) = P(Class > V_{k-1}).$$

- Clases intermedias:

$$P(V_i) = P(Class \geq V_{i-1}) \times (1 - P(Class > V_i)), \quad 1 < i < k.$$

Por lo que cada clase predicha se asigna a la instancia x_{new} según la probabilidad estimada $P(V_i)$ más alta. Esta misma metodología se implementó para los árboles de desición. Aprovechando que podemos obtener las probabilidades de cada clase también.

V. Dataset

Para realizar las pruebas de los 5 modelos utilizamos la base de datos *Car Evaluation*. Dentro del aprendizaje automático, este conjunto de datos fue utilizado para la evaluación de HINT (Herramienta de Inducción de Jerarquías), que demostró ser capaz de reconstruir completamente el modelo jerárquico original. La Base de Datos de Evaluación de Automóviles se derivó de un simple modelo jerárquico de decisión desarrollado originalmente para la demostración de DEX (M. Bohanec, V. Rajkovic: Sistema experto para la toma de decisiones). El modelo evalúa automóviles de acuerdo con la siguiente estructura de conceptos:

Concepto	Descripción
CAR	Aceptabilidad del automóvil
PRICE	Precio general
buying	Precio de compra
maint	Precio del mantenimiento
TECH	Características técnicas
COMFORT	Comodidad
doors	Número de puertas
persons	Capacidad en términos de personas a transportar
lug_boot	Tamaño del maletero
safety	Seguridad estimada del automóvil

Cuadro I
Estructura de conceptos para la evaluación de automóviles.

Consta de 1728 instancias y 6 atributos. De los cuales los 6 atributos toman los valores de acuerdo al cuadro II.

Atributo	Valores posibles
buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

Cuadro II
Atributos y sus valores posibles para la evaluación de automóviles.

Esta base de datos no cuenta con valores faltantes, por lo que fue de gran ventaja para su análisis. En el cuadro III podemos ver los valores que toma la clase objetivo y en la tabla IV podemos observar cómo se distribuye, gráficamente podemos verlo en la figura 2.

Clase Objetivo	Significado
unacc	Inaceptable
acc	Aceptable
good	Bueno
v-godd	Muy bueno

Cuadro III
Variable objetivo.

Por tanto todos los métodos de clasificación se realizaron en base a estos datos, podemos notar que las clases se encuentran muy desvalanceadas y esto puede ser un problema para los modelos de clasificación.

VI. Resultados

En esta sección mostraremos los resultados de las metodologías implementadas para cada modelo de clasificación.

Clase	N	N[%]
unacc	1210	(70.023 %)
acc	384	(22.222 %)
good	69	(3.993 %)
v-good	65	(3.762 %)

Cuadro IV
Distribución de clases (número de instancias por clase).

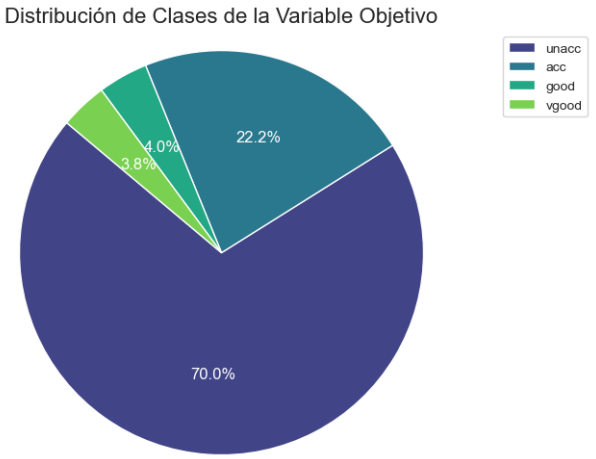

Figura 2. Distribución de las clases.

Cabe aclarar que dado el gran desvalance entre las clases, se implementó la función *SMOTE*. Para cada muestra en la clase minoritaria, *SMOTE* busca sus vecinos más cercanos en el espacio de características. Luego, selecciona uno o varios de estos vecinos y crea nuevas muestras sintéticas entre la muestra original y sus vecinos. Estas nuevas muestras se añaden al conjunto de datos, aumentando así el número de muestras en la clase minoritaria. Al generar estas muestras sintéticas, *SMOTE* ayuda a equilibrar el número de muestras en todas las clases, lo que puede mejorar el rendimiento de los modelos de aprendizaje automático al reducir el sesgo hacia las clases mayoritarias. Se hizo la partición de los datos en un 20 % para test y 80 % para entrenamiento (no olvidemos que al tratar el desvalance de las clases, el número de muestras de entrenamiento incrementó para compensar las clases minoritarias, los datos de test quedaron igual en cantidad). También cabe aclarar que para realizar la comparación entre el modelo ordinal y el modelo estándar, se pudieron haber elegido distintos parámetros, pudiendo unos u otro favorecer al modelo al clasificar, es por esta razón que para cada modelo estándar se realizó un 3 fold-CV, con los parámetros más importantes de cada modelo y de esta forma determinar un modelo más eficiente para este conjunto de datos. En base a los parámetros del modelo estándar con mejores resultados, se usaron los mismo para los modelos ordinales, de esta forma se pensó así para realizar una mejor comparación entre modelos ordinales y no ordinales.

VI-A. *Regresión logística*

VI-A1. Regresión logística estándar: Para la regresión logística el grid se hizo considerando los siguientes parámetros

```
param_grid = {
    'max_iter': [1000, 2000, 3000],
    'solver': ['lbfgs', 'newton-cg', 'sag'],
    'multi_class' : ['ovr', 'multinomial']
}
```

De los cuales los mejores parámetros resultaron ser max_iter: 1000, multi_class: 'multinomial', solver: 'sag'. De esta forma la matriz de confusión para el modelo de regresión logística con estos parámetros puede verse en al figura 3.

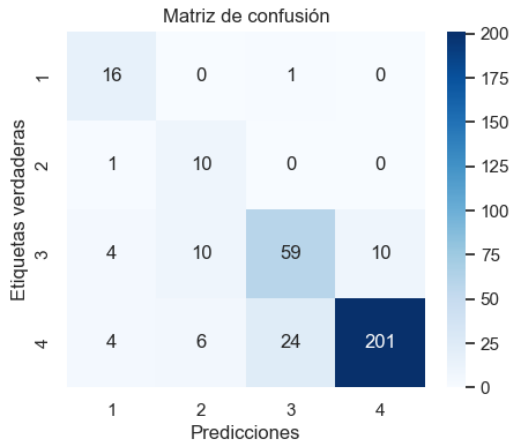

Figura 3. Matriz de confusión de la Regresión logística estándar. Para evitar la escritura en las figuras se cambiaron el nombre de las clases por números, 4:Inaceptable, 3:Aceptable, 2:Bueno, 1:Muy bueno.

Podemos notar que a pesar que no es el modelo ordinal, se equivocó 60 veces de 346, en general para las clases 1 y 2 tuvo buena precisión, equivocándose una única vez en ambas clases, también recordemos que estás son las clases con menos elementos, para la clase 3, tuvo 24 errores de 83, se equivocó principalmente con la clase 2 y 4, el valor con el que más errores fue con el 4, que corresponde al valor de inaceptable, y se equivocó con el valor 3, de aceptable, por lo que la penalización por estos errores no fue la más alta, también los valores recordemos que esta clase es la que tiene mayor número de muestras. Finalmente este modelo obtuvo un $MAE = 0.2293$ y un $QWK = 0.7513$

VI-A2. Regresión ordinal - Modelo de probabilidades proporcionales: La matriz de confusión para este modelo se muestra a continuación

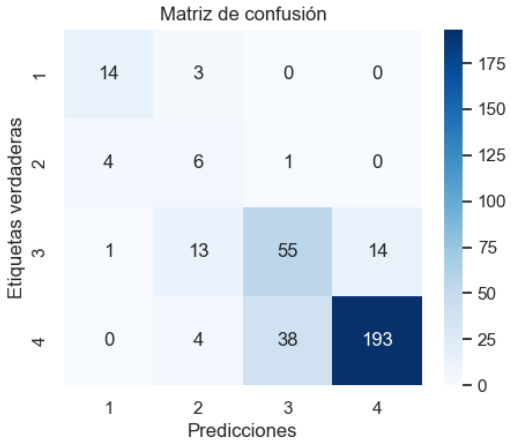

Figura 4. Matriz de confusión de la Regresión logística ordinal usando el modelo de probabilidades proporcionales.

Algo interesante que podemos observar es que se equivoca más que el modelo de regresión logística estándar, pero a pesar de los errores que comete, lo hace con menor penalización, por ejemplo con la clase 1, tuvo 3 errores pero los tres errores fueron con clase que le procede, la clase 2, en cambio con el modelo estándar, tuvo 1 error pero fue con la clase 3, un error con mayor penalización, lo mismo sucede con las demás clases. A pesar de que es este modelo considera la ordinalidad de los datos, obtuvo un $MAE = 0.2398$, un poco menor que el modelo estándar, sin embargo el valor de QWK fue $QWK = 0.8004$, siendo mayor que el estándar, lo cual indica que obtuvo una mejora.

VI-A3. Regresión Ordinal - Modelo de riesgos proporcionales: La matriz de confusión es la siguiente

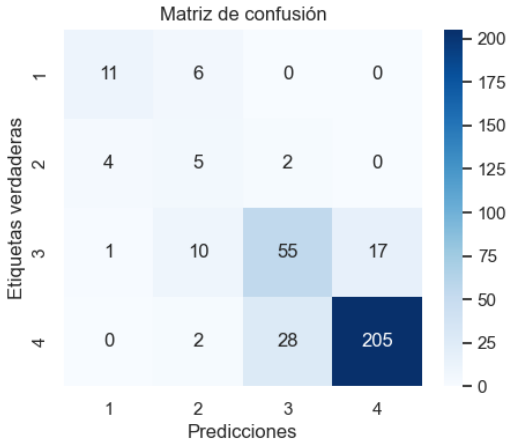

Figura 5. Matriz de confusión de la Regresión logística ordinal usando el modelo de riesgos probabilidades proporcionales.

Podemos observa la matriz de confusión y notar que se equivoca más en las clases 4 y 3 y en las clases 1 y 2 se equivoca un poco más que el modelo de probabilidades proporcionales. Sin embargo a pesar de esto, la penalización de los error son menores puesto que no comete errores "graves"

como es el caso del modelo estándar, por ejemplo, para la clase 4 de datos verdaderos tiene 2 errores menos para la clase 2 y 10 errores menos para el 3. Por tanto, con esto obtuvo un valor de $MAE = 0.2109$ y un valor de $QWK = 0.8202$, es decir tuvo una mejor puntuación en ambas métricas, aunque en realidad no por mucho. Con esto podemos corroborar que a pesar de que se cometan más errores, el tipo de error influye en las métricas y lo pudimos ver reflejado en este ejemplo, y podemos corroborar que el modelo capto la ordinalidad de los datos. Sin embargo, dado que este último modelo tuvo una mejor puntuación, entonces vamos a considerarlo como el modelo que representará al modelo de regresión logística ordinal.

VI-B. Adaboost y Boosting de división fija

VI-B1. Adaboost estándar: Para el modelo de Adaboost únicamente se probó cambios con un parámetro, el cual se muestra continuación.

```
param_grid = {
    'n_estimators': [50, 100, 150, 200,
    300, 400, 500, 1000]
}
```

Para estos modelos resulto que el mejor modelo era para n_estimators: 150. De esta forma la matriz de confusión para el modelo de Adaboost puede verse en la figura 6.

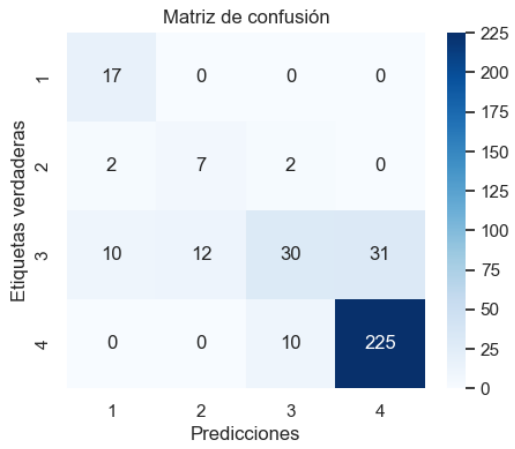

Figura 6. Matriz de confusión de Adaboost estándar.

Para el modelo de Adaboost estándar, podemos observar que se equivoca menos en la clase 4 y 1, pero y en la clase 3 comete muchos más errores en comparación del modelo de regresión logístico estándar, prácticamente para la clase 1 no hay errores, para la clase 2 comete 4 errores, y la clase 3 comete 53 errores, sin embargo para la clase 4 comete únicamente 10 errores y con la clase que le prosigue, la clase 3. A pesar de esto tiene la siguiente puntuación $MAE = 0.2225$ y $QWK = 0.8092$, esto nos indica que el modelo de adaboost estándar resultó mejor que el modelo de regresión logística estándar.

VI-B2. Boosting de división fija: La matriz de confusión para el modelo boosting de división fija se pude observar en la figura 7.

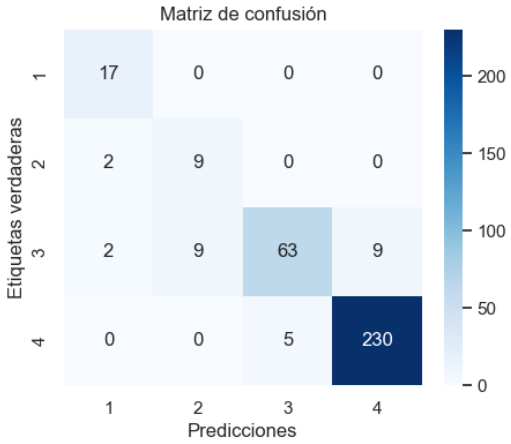

Figura 7. Matriz de confusión de Boosting de división fija.

Al observar la matriz de confusión podemos ver que este modelo resultó ser mucho más preciso que los modelos anteriores, se equivocó 27 veces de 346 pero no solo eso, si no que los errores cometidos no reciben mucha penalización, por ejemplo para la clase 4, se equivocó únicamente 5 veces para con la clase 3, y cero con las clases 1 y 2. Para la clase 1 no tuvo errores, para la clase 2, tuvo 2 errores confundiéndolo con la clase 1. Por estás buenas características obtuvo una puntuación de $MAE = 0.0838$ y $QWK = 0.9282$, una diferencia grande en comparación con el adaboost ordinal y los demás modelos.

VI-C. Árboles de decisión

VI-C1. Árboles de decisión estándar: Para los árboles de decisión se probaron los siguientes parámetros

```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [10, 20, 40, 80],
}
```

Por tanto el mejor modelo obtuvo los siguientes parámetros, criterion: 'entropy', max_depth: 20. La matriz de confusión para Árboles de decisión lo podemos ver en la figura 8.

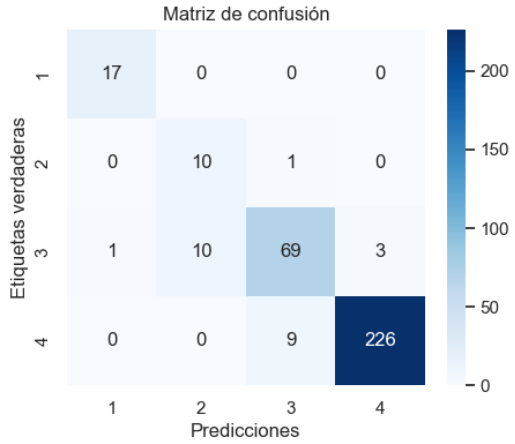

Figura 8. Matriz de confusión para árboles de decisión.

Podemos observar que el modelo se equivoca 24 veces de 346, siendo este el modelo con menos errores hasta el momento, por tanto, para la clase 1 no cometió errores, para la clase 2 cometió un único error, para la clase 3 cometió 14 errores, de los cuales 1 fue con 1, por lo que eso lo penalizó más. Para la clase 4 cometió 9 errores pero con la clase 3 que le prosigue. Por tanto las métricas son $MAE = 0.0722$ y $QWK = 0.9393$. Superando al boosting de división fija.

VI-C2. Árboles de decisión ordinal: Paralos de decisión ordinal, la matriz de confusión se puede observar en la figura 9.

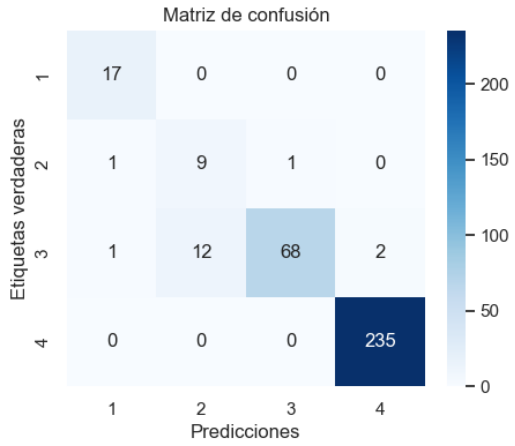

Figura 9. Matriz de confusión para árboles de decisión ordinal.

Al analizar la matriz de confusión podemos notar que esta vez comete 17 errores, disminuyendo la cantidad de errores con respecto al árbol de decisión estándar. Para las clases 1 y 4 no comete errores, para la clase 2 comete 2 errores y para la clase 3 comete 15 errores, siendo esta la clase con más errores. Por tanto las métricas son $MAE = 0.0520$ y $QWK = 0.9558$. De esta forma supera al árbol de decisión estándar. Teniendo la mejor puntuación hasta el momento.

VI-D. Redes neuronales

VI-D1. Redes neuronales estándar: Para las redes neuronales se probó un grid con los siguientes parámetros.

```
param_grid = {
    'hidden_layer_sizes': [(5,5,5,5,5),
    (10,10,10), (10,10,10,10,10)],
    'activation': ['relu', 'tanh', 'logistic'],
}
```

Tras la prueba con los distintos parámetros se determinó que el mejor modelo debía tener los siguientes parámetros: activation: 'relu', hidden_layer_sizes: (10, 10, 10, 10, 10). Por tanto la matriz de confusión para la red neuronal con 5 capas y 10 nodos con los parámetros mostrados se observa en la figura 10.

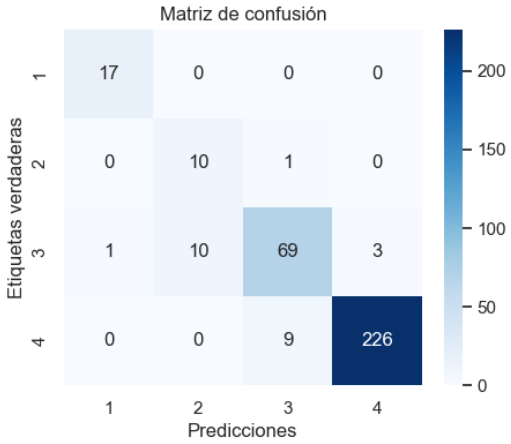

Figura 10. Matriz de confusión de la Red Neuronal estándar.

Podemos observar que el modelo comete 24 errores, el mismo número de errores del árbol de decisión estándar, la clase 1 no tiene errores, la clase 2 tiene un error en 3, la clase 3 tiene 14 errores, y uno es en la clase 1 por lo que se penaliza fuertemente, por último la clase 4 comete 9 errores con la clase 3, que le prosigue. Con esto la red neuronal estándar tiene $MAE = 0.0722$ y $QWK = 0.9393$. Casualmente estás métricas fueron muy similares al árbol de decisión estándar.

VI-D2. Red Neuronal Ordinal: La matriz de confusión para la red neuronal ordinal se observa en la figura 11.

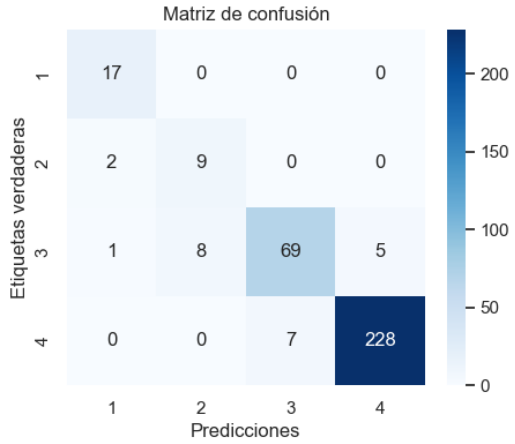

Figura 11. Matriz de confusión de la Red Neuronal ordinal.

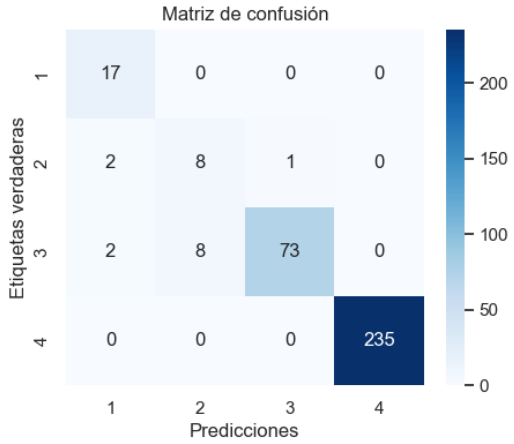

Figura 12. Matriz de confusión para Random Forest estándar.

Podemos notar que comete 23 errores, disminuyendo un error con respecto al modelo estándar, a pesar de que el modelo de red neuronal estándar ya era bueno, mejoró aún más. Las métricas que obtuvo son las siguientes: $MAE = 0.0693$ y $QWK = 0.9423$. Aunque no existe mucha diferencia entre ambas redes neuronales, este ha proporcionado la mejor puntuación.

VI-E. Random Forest

VI-E1. Random Forest estándar: Para random forest los parámetros que se utilizaron para el grid fueron los siguientes.

```
    param_grid = {
    'n_estimators': [50, 100, 150, 200,
    500, 1000],
    'max_depth': [10, 20, 30],
    'max_features': ['sqrt', 'log2']
}
```

Después de realizar las pruebas se determinó que el mejor modelo debía tener los parámetros siguientes: max_depth: 30, max_features: 'sqrt', n_estimators: 150. Por tanto para un modelo random forest estándar con los parámetros indicados se tiene la matriz de confusión como se muestra a continuación.

Podemos observar que éste modelo ha cometido 13 errores, ha sido el modelo con menos errores de todos los analizados hasta ahora, las únicas clases en las cuales se ven reflejados estos errores son en la clase 2 y la clase 3. Las métricas obtenidas con este modelo son las siguientes: $MAE = 0.0433$ y $QWK = 0.9582$, por lo que lo convierte el modelo con mejor puntuación, siendo muy bueno, aún superando a las redes neuronales.

VI-E2. Random forest ordinal: Para este modelo la matriz de confusión se observa en la figura 13

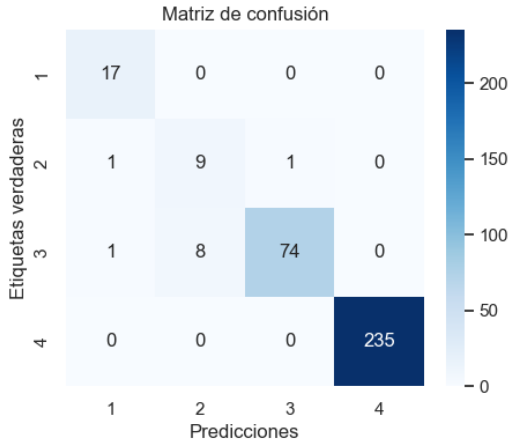

Figura 13. Matriz de confusión para el modelo Random Forest Ordinal.

Podemos observar en la matriz de confusión que únicamente comente 11 errores, superando a su versión estándar, por lo que este modelo es muy preciso, su métricas son las siguiente: $MAE = 0.0346$ y $QWK = 0.9684$, siendo este el mejor modelo, con la mejor puntuación.

Una de las ventajas de Random Forest, es la interpretabilidad, por lo que podemos aprovechar a obtener los pesos reflejados en cada una de las variables (ver figura 14).

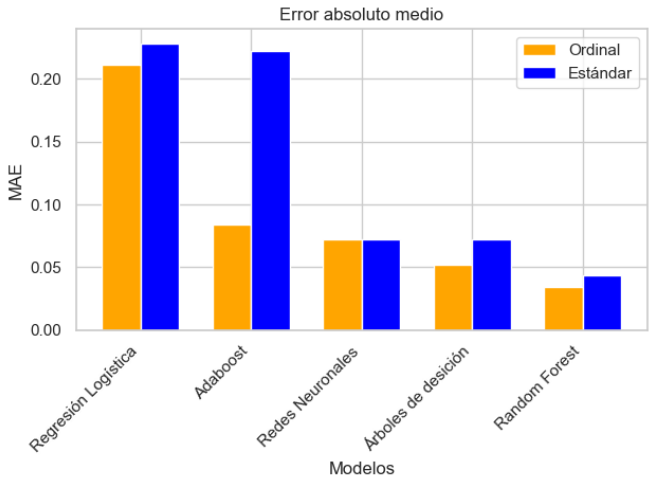

Figura 15. Mean Absolute Error de todos los modelos.

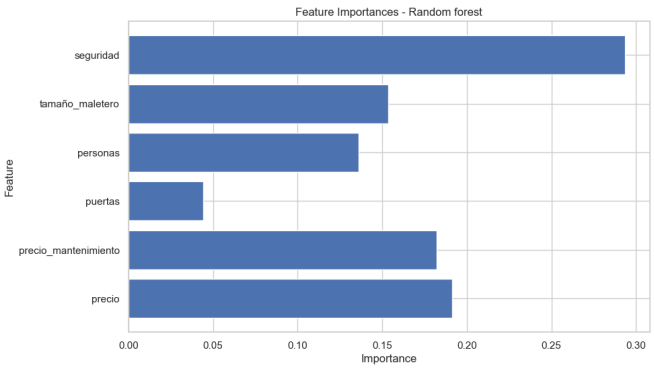

Figura 14. Importancia de las variables.

De aquí podemos observar la importancia de cada una de las variables consideradas de la base de datos, podemos notar que la seguridad en los automóviles son los que toman mayor importancia, lo cual hace sentido cuando se evalúa el auto, una buena apariencia en este refleja lo seguro que es. La segunda variable característica más importante es el precio de este, posteriormente el precio de mantenimiento, luego el tamaño del maletero, posteriormente el número de personas y por último el número de puertas. El orden de importancia de las variables hace sentido al momento de evaluar un automóvil, por lo que es bastante acertada.

VII. Resultados de Métricas

Ya se ha analizado cada uno de los modelos en su forma estándar y ordinal. Por lo que todos los resultados obtenidos podemos resumirlos en las figuras 15, 16.

Recordemos que la métrica MAE nos refleja un valor entre 0 y 1, por lo que entre más cerca del cero se encuentra, mejor modelo es. Podemos distinguir de la figura 15 el valor de la métrica para cada modelo, para los modelos estándar se encuentran en barras azules y para los ordinales en naranja, de esta forma podemos distinguir primeramente que el mejor modelo con la puntuación más baja fue el Random Forest ordinal, posteriormente le siguen los árboles de desición ordinal, posteriormente las redes neuronales ordinales, posteriormente bagging ordinal y por último la regresión logística ordinal. De manera general podemos observar que todos los modelos ordinales presentaron una mejor puntuación respecto a su modelo estándar. Aunque como se mostró con la regresión logística, el modelo ordinal presentaba más errores, pero los errores eran menos penalizados que el modelo estándar, por lo que al final su puntuación terminó siendo mejor, por tanto para estos datos podemos decir que considerar métodos para datos ordinales demostró ser más eficiente que los modelos estándar, y para este conjunto de datos el mejor modelo fue random forest, y el peor la regresión logística.

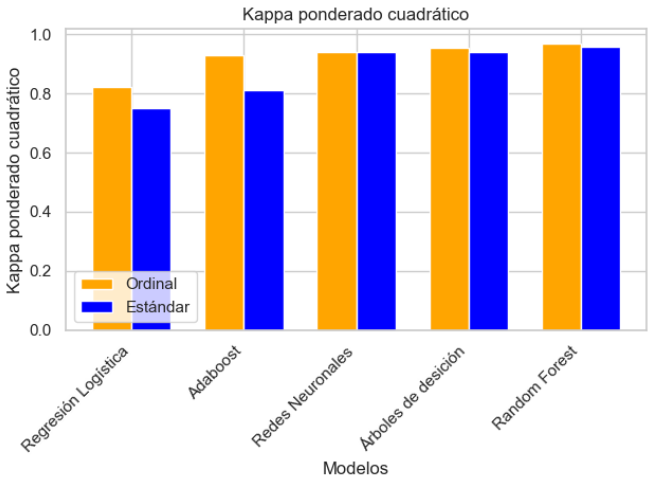

Figura 16. Quadratically Weighted Kappa para todos los modelos.

De la figura 16 podemos observar la puntuación con la métrica Kappa ponderado cuadrático, con está métrica podemos saber la eficiencia del modelo al notar que las métrica se encuentran cercanas a 1. Por tanto, podemos notar que nuevamente radom forest ordinal muestra el mejor desempeño, seguido de su modelo estándar, posteriormente árboles de decisión ordinal, posteriormente redes neuronales ordinal, posteriormente boosting de divisón fija, y por último la regresión logística ordinal, es decir se ha mantenido el mismo orden que con la métrica del error absoluto medio, por tanto podemos decir que el mejor modelo para estos datos de forma general es el random forest, seguido de las redes neuronales, el peor modelo fue la regresión logística. Un punto a tomar en cuenta es que al notar que no hay tanta distancia entre las redes neuronales y random forest, es posible que con más capas ocultas el rendimiento de las redes neuronales mejoren la puntuación, aunque esto implicaría un mayor tiempo computacional.

Podemos notar que tanto redes neuronales, árboles de decisión y random forest en su forma estándar hacen un muy buen trabajo, a pesar de no considerar la ordinalidad, y cuando la consideran el aumento en las métricas no es mucho. Aunque

para estos datos el mejor modelo fue Random forest, se han hecho estudios donde se han comparado distintas metodologías con distintas bases de datos ordinales [3], y se ha llegado al resultado que no existe un mejor método definido para cualquier conjunto de datos, en general depende mucho de los datos, y un método u otro puede dar mejores resultados.

VIII. Conclusiones

- Los modelos ordinales dieron un mejor resultado respecto a sus modelos estándar.
- El mejor modelo para los datos ordinales depende de los datos y no hay un solo método definido.
- Los modelos ordinales pueden presentar más errores pero con menor penalización respecto a sus modelos estándar.
- Considerar el carácter ordinal de los datos en general mejora la eficiencia en la predicción de nuevos datos.

Referencias

[1] W. Chu, Z. Ghahramani, and C. K. Williams, "Gaussian processes for ordinal regression.," *Journal of machine learning research*, vol. 6, no. 7, 2005.

[2] R. Herbrich, T. Graepel, K. Obermayer, *et al.*, *Regression models for ordinal data: A machine learning approach*. Citeseer, 1999.

[3] P. A. Gutiérrez, M. Perez-Ortiz, J. Sanchez-Monedero, F. Fernandez-Navarro, and C. Hervas-Martinez, "Ordinal regression methods: survey and experimental study," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 127–146, 2015.

[4] P. K. R. Kumar and Y. Kumar, "Time series data prediction using iot and machine learning technique," in *Procedia Computer Science*, p. 373–381, ECML, 2020.

[5] U. Jakobsson and A. Westergren, "Statistical methods for assessing agreement for ordinal data," in *Scand J Caring Sci*, pp. 6–7, 2005.

[6] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 109–127, 1980.

[7] G. Tutz and K. Hechenbichler, "Aggregating classifiers with ordinal response structure," *Journal of Statistical Computation and Simulation*, vol. 75, no. 5, pp. 391–408, 2005.

[8] J. Cheng, Z. Wang, and G. Pollastri, "A neural network approach to ordinal regression," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1279–1284, IEEE, 2008.

[9] E. Frank and M. Hall, "A simple approach to ordinal classificaction," in *PEND*, pp. 146 – 156, ECML, 2001.

IX. Contribuciones

Armando Salinas Lorenzana
- Regresión Logística Ordinal
- Redes Neuronales para datos ordinales.

Yulissa Crockwell Palomares
- Una método aplicado a Adaboost
- Un enfoque simple para la clasificación ordinal aplicado a Random Forest.

Ambas partes hicimos la revisión bibliográfica, se revisó en conjunto la metodología, cada parte se encargó de realizar el código pero fue supervisado por la otra parte. Ambos contribuimos en la escritura del reporte y la creación de la presentación.