Trabajo Final: Predicción de Lectura de Etiquetado

Alumno: Julio César Velázquez Corona 00356796

Docente: María del Carmen Villar Patiño

Materia: Machine Learning

9 de Mayo del 2025

Índice

1. Entendimiento del Negocio

Se llevará a cabo un proceso de Machine Learning aplicado a datos de salud del INEGI. Los datos provienen de la encuesta ENSANUT 2018, del cuestionario de etiquetado frontal de alimentos. La intención es predecir si un mexicano va a leer y ocupar la información que muestra el etiquetado frontal en alimentos empacados y bebidas embotelladas. En el diccionario de datos se encuentra la descripción de las columnas y contempla:

- Datos sociodemográficos
- Preguntas del cuestionario

1.a. En un párrafo, haz una descripción de la razón e importancia por la cual el Gobierno de México creó el etiquetado frontal de alimentos. Incluir cuándo entró en vigor.

El principal motivo por el que considero que se creó el etiquetado es para reducir las enfermedades provocadas por el consumo de alimentos hipercalóricos o con mucha azucar y también reducir la obesidad en nuestro país. Ya que México ocupa el primer puesto de obesidad infantil en el mundo. La implementación del etiquetado en los alimentos entró en vigor el 1 de Octubre de 2020.

1.b. Crear una tabla con las variables, su tipo, sus posibles valores, unidades (si aplica) y una definición sencilla.

La tabla la hice como un dataframe en R, utilicé strings en lugar de factores para manejar los datos de manera más sencilla:

```
tablaDeVariables <- data.frame(
    Variable = c("p1", "p3", "p4", "p7", "sexo", "ent", "dominio", "estrato", "region"),
    Tipo = c(
        "Categórica ordinal",
        "Categórica nominal",
        "Categórica nominal",
        "Categórica ordinal",
        "Binaria",
        "Categórica nominal",
        "Binaria",
        "Categórica ordinal",
        "Categórica nominal"
    ),
    valoresPosibles = c(
        "1: menos de 500, 2: de 500 a 1000, 3: de 1001 a 1500, 4: de 1501 a 2000, 5: de
        ↪  2001 a 3000, 6: de 3001 a 4000, 7: más de 4000, 9: No sabe / No respondió",
        "1: Sí, 2: No, 9: No sabe / No responde",
        "1: Sí, 2: No, 9: No sabe / No responde",
        "1: Nunca, 2: Casi nunca, 3: A veces, 4: Casi siempre, 5: Siempre, 9: No sabe /
        ↪  No responde",
        "1: Hombre, 2: Mujer",
```

```
        "1: Aguascalientes, 2: Baja California, 3: Baja California Sur, 4: Campeche, 5:
        ↪  Coahuila de Zaragoza, 6: Colima, 7: Chiapas, 8: Chihuahua, 9: Ciudad de
        ↪  México, 10: Durango, 11: Guanajuato, 12: Guerrero, 13: Hidalgo, 14: Jalisco,
        ↪  15: México, 16: Michoacán de Ocampo, 17: Morelos, 18: Nayarit, 19: Nuevo
        ↪  León, 20: Oaxaca, 21: Puebla, 22: Querétaro, 23: Quintana Roo, 24: San Luis
        ↪  Potosí, 25: Sinaloa, 26: Sonora, 27: Tabasco, 28: Tamaulipas, 29: Tlaxcala,
        ↪  30: Veracruz de Ignacio de la Llave, 31: Yucatán, 32: Zacatecas",
        "1: Urbano, 2: Rural",
        "1: Bajo, 2: Medio bajo, 3: Medio alto, 4: Alto",
        "1: Norte, 2: Centro, 3: Ciudad de México, 4: Sur"
    ),
    Unidades = c("Calorías", "No aplica", "No aplica", "No aplica", "No aplica", "No
    ↪  aplica", "No aplica", "No aplica", "No aplica"),
    Definición = c(
        "Calorías que el encuestado considera adecuadas para su edad y sexo.",
        "Si el encuestado sabe que los alimentos tienen información nutricional o no.",
        "Si el encuestado lee o no la información nutricional en alimentos y bebidas.",
        "Frecuencia con la que el encuestado revisa etiquetas al comprar.",
        "Sexo del encuestado",
        "Entidad federativa del encuestado.",
        "Si el encuestado es de zona urbana o rural.",
        "Estrato socioeconómico del encuestado.",
        "Región geográfica del país del encuestado."
    ),
    stringsAsFactors = FALSE
)
```

1.c. Plantear el problema de clasificación inicial (omitir distribución de probabilidad), considerando lo siguiente: predecir si una persona lee la información nutrimental (variable dependiente P4, con los valores de SI y NO), el valor de interés es NO y la variable P7 no se considera.

- Descripción: contamos con información de la encuesta ENSANUT 2018 del INEGI, la cual contiene datos sobre la gente y el etiquetado nutrimental, al tomar en cuenta a una nueva persona o encuestado, predecir si lee o no la información nutrimental de los alimentos.
- Problema de clasificación:
  - Población Ω: Personas encuestadas en la ENSANUT 2018.
  - Clases:
    - Ω_0: El encuestado sí lee la información nutrimental (P4 = 1).
    - Ω_1: El encuestado no lee la información nutrimental (P4 = 2).
    - Nuestro valor de interés es Ω_1.
  - Vector de caracteristicas X: {p1, p3, sexo, entidad, dominio, estrato, region}.

1.d. Leer los datos.

```
> datos <- read.csv("datosTF.csv")
> head(datos)
    P1 P3 P4 P7 EDAD SEXO ENT DOMINIO REGION EST_DIS UPM_DIS ESTRATO
  1  9  1  1  2   73    2   1       1      2       6       1       3
```

```
  2  9  1  2  2   55    1   1       1      2       6       8       3
  3  9  1  1  4   51    2   1       1      2       6       8       3
  4  1  1  1  4   34    2   1       1      2       6       8       3
  5  1  1  2  2   46    1   1       1      2       6       8       3
  6  9  2 NA  4   22    2   1       1      2       5       9       2
```

1.d.i. Crear un subconjuntos de datos correspondiente al problema.

Para que este subconjunto de datos sea correspondiente a nuestro problema, debemos eliminar P7:

```
> subConjDatos <- subset(datos, select = -P7)
> head(subConjDatos)
    P1 P3 P4 EDAD SEXO ENT DOMINIO REGION EST_DIS UPM_DIS ESTRATO
  1  9  1  1   73    2   1       1      2       6       1       3
  2  9  1  2   55    1   1       1      2       6       8       3
  3  9  1  1   51    2   1       1      2       6       8       3
  4  1  1  1   34    2   1       1      2       6       8       3
  5  1  1  2   46    1   1       1      2       6       8       3
  6  9  2 NA   22    2   1       1      2       5       9       2
```

1.d.ii. Eliminar los renglones con NA y de las personas que no saben o no respondieron (código 9).

```
> datosLimpios <- subConjDatos %>% filter(!is.na(P4))
> any(is.na(datosLimpios$P4))
[1] FALSE
> datosMasLimpios <- datosLimpios %>% filter(P4 != 9)
> summary(datosMasLimpios$P4)
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.00    1.00    1.00    1.43    2.00    2.00
```

1.d.iii. Sustituir en la variable dependiente (P4), el valor numérico por su significado (código).

```
> datosMasLimpios2 <- datosMasLimpios %>% mutate(P4 = case_when(P4 == 1 ~ "Si", P4 == 2 ~
↪  "No"))
> head(datosMasLimpios2)
    P1 P3 P4 EDAD SEXO ENT DOMINIO REGION EST_DIS UPM_DIS ESTRATO
1  9  1 Si   73    2   1       1      2       6       1       3
2  9  1 No   55    1   1       1      2       6       8       3
3  9  1 Si   51    2   1       1      2       6       8       3
4  1  1 Si   34    2   1       1      2       6       8       3
5  1  1 No   46    1   1       1      2       6       8       3
6  9  1 Si   43    2   1       1      2       5       9       2
```

1.d.iv. Reportar el número de renglones que quedó en los datos finales.

```
> length(datosMasLimpios2$P4)
[1] 28112
```

2. Entendimiento de los Datos

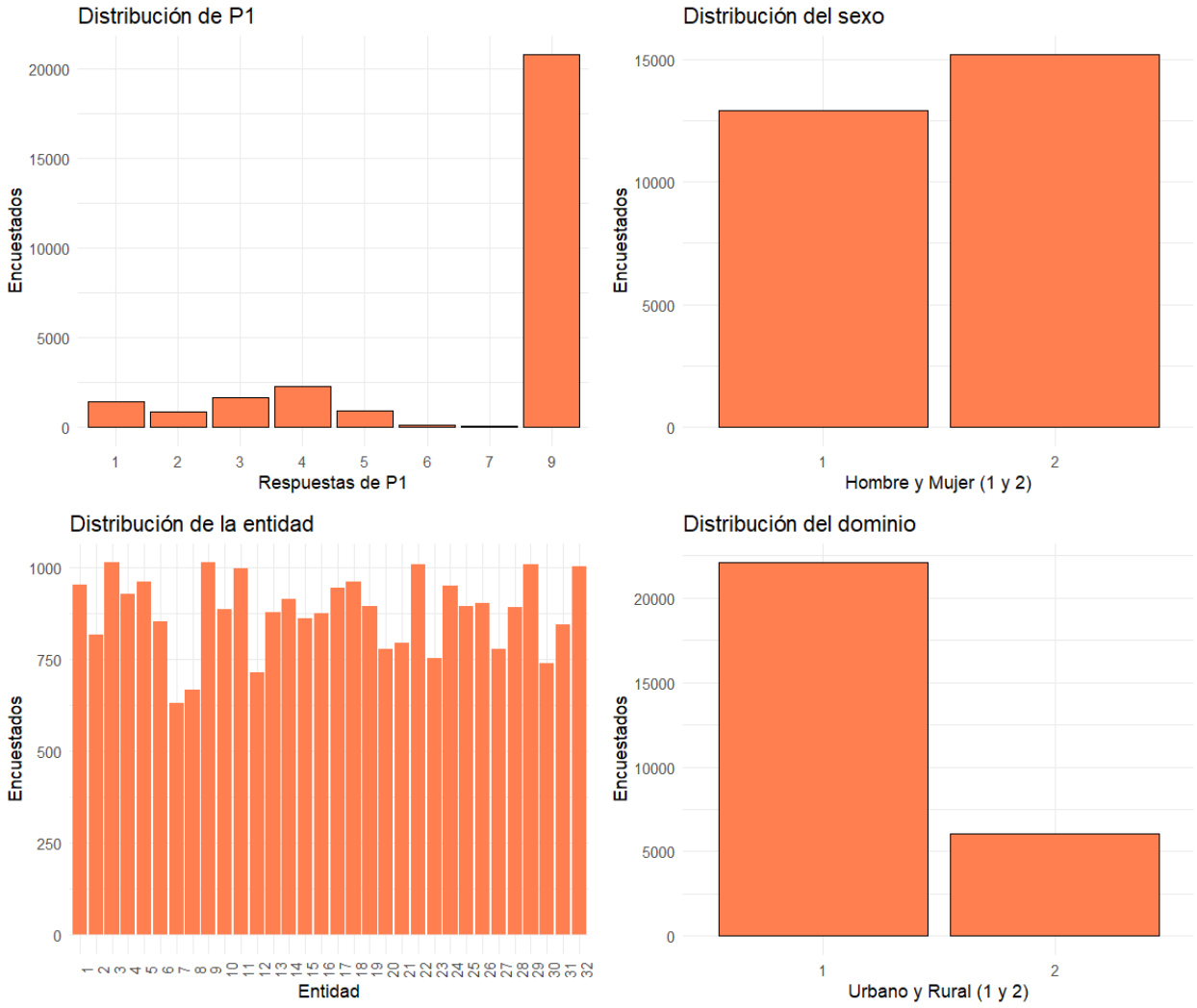
Análisis estadístico de los rasgos del vector de características (apoyarse de gráficos, de tamaño apropiado).

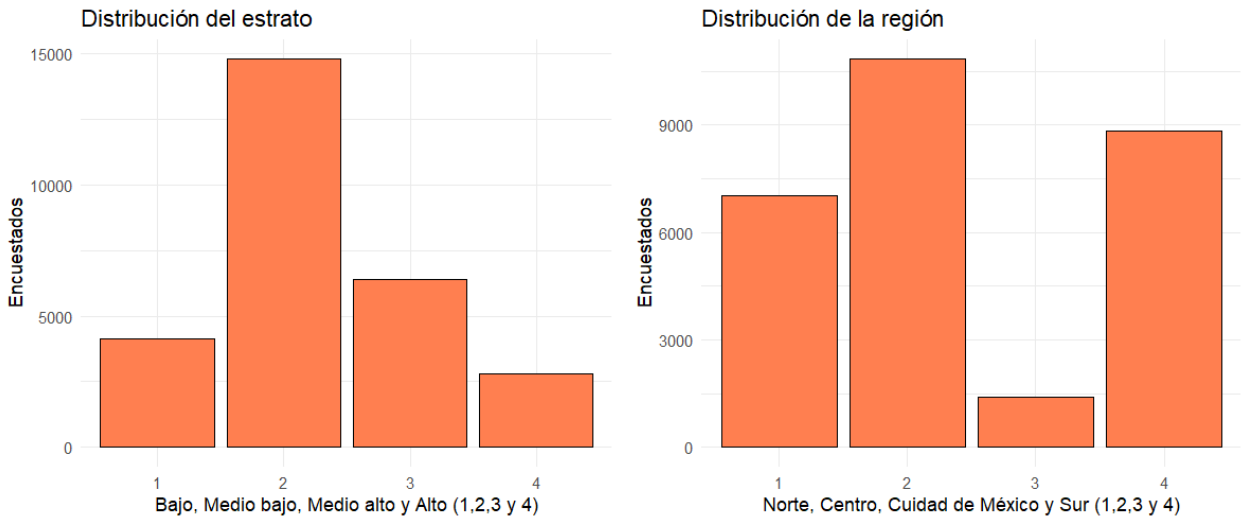
2.a. Número de datos faltantes por columna.

```
> # No hay datos NA
> sapply(datosMasLimpios2, function(x) sum(is.na(x)))
   P1      P3      P4    EDAD    SEXO     ENT DOMINIO  REGION EST_DIS UPM_DIS ESTRATO
    0       0       0       0       0       0       0       0       0       0       0
```

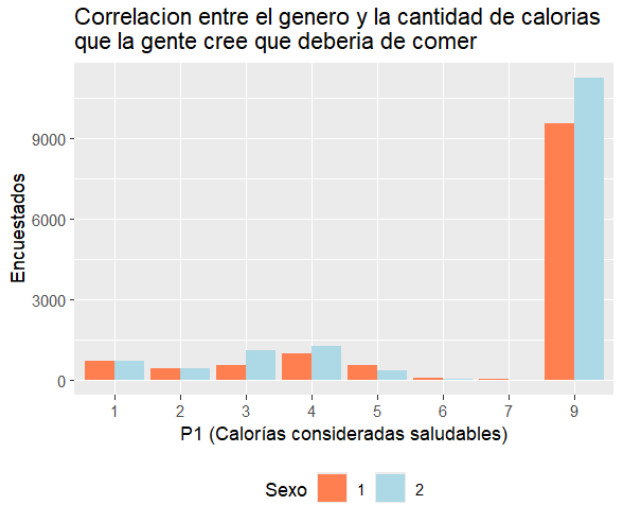
2.b. Análisis univariado.

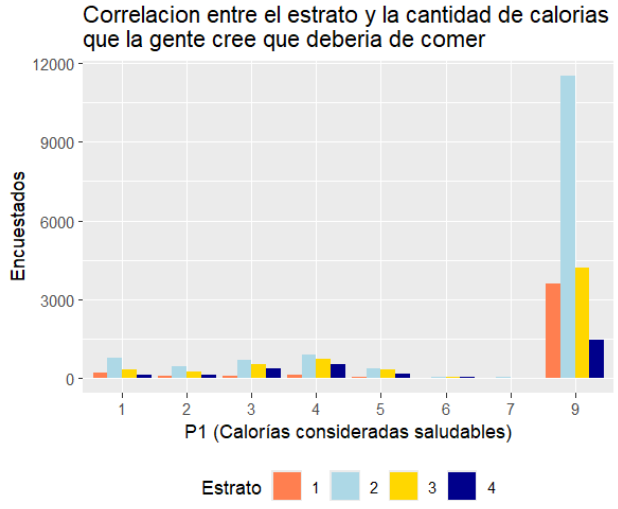
```
# No hay datos distintos al 1
> summary(datosMasLimpios2$P3)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1       1       1       1       1       1
```



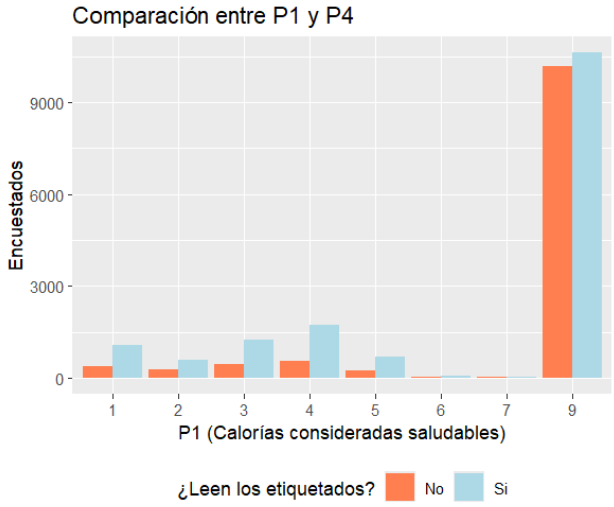


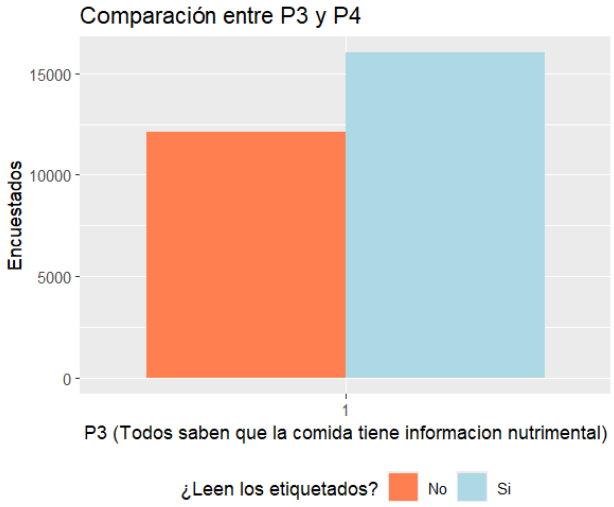
2.c. Análisis de correlación.

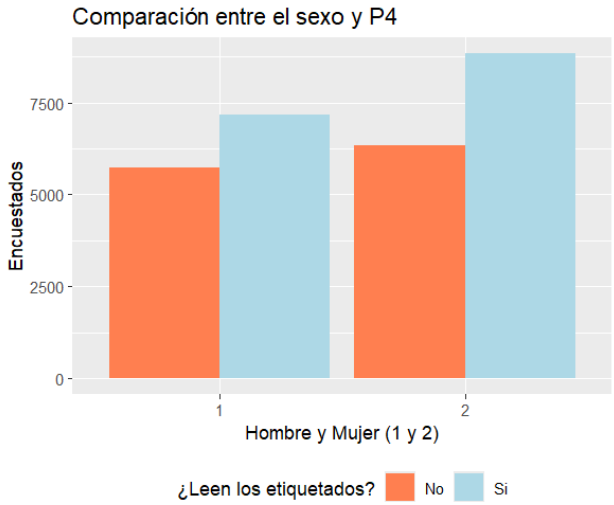


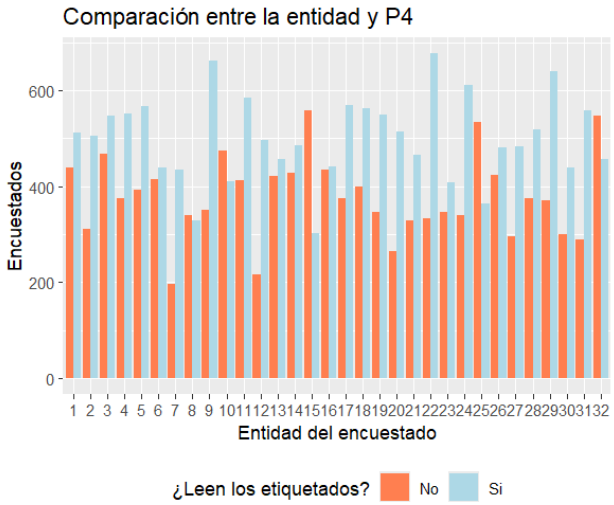


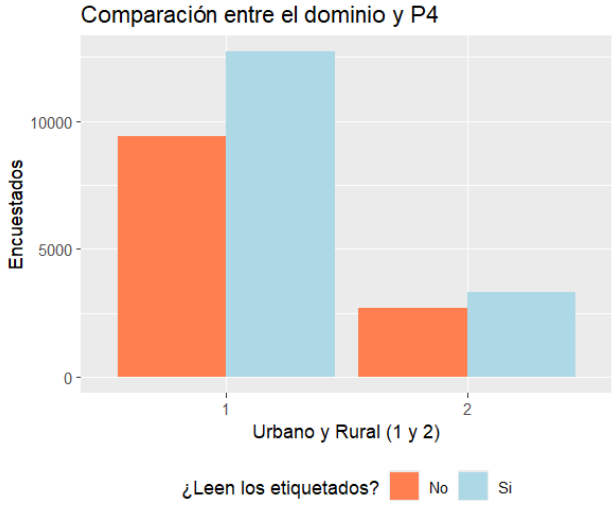
2.d. Realizar una comparación de la distribución de cada una de las variables con respecto a la variable de interés.

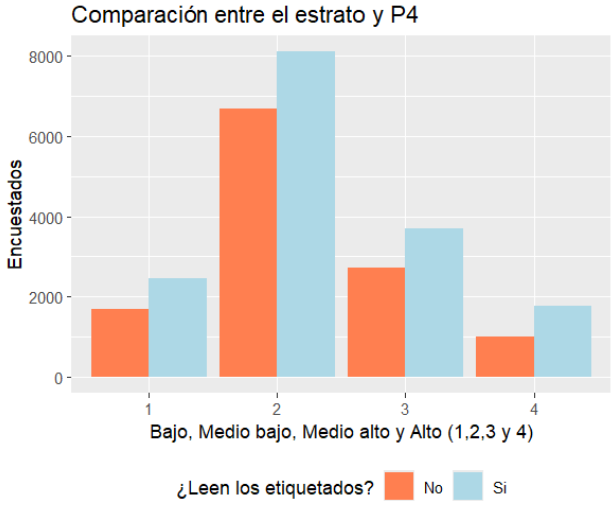


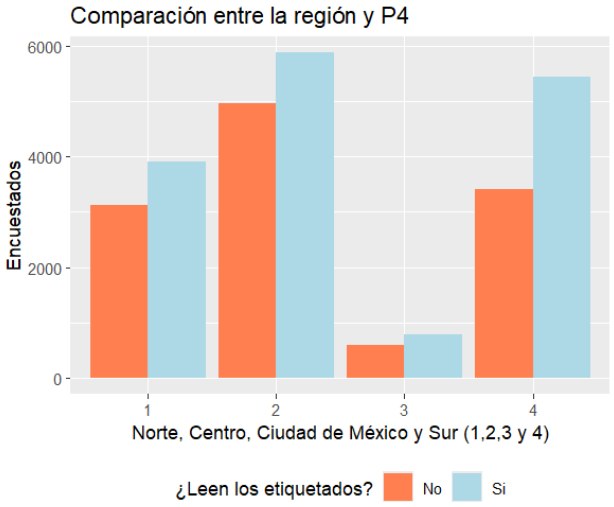




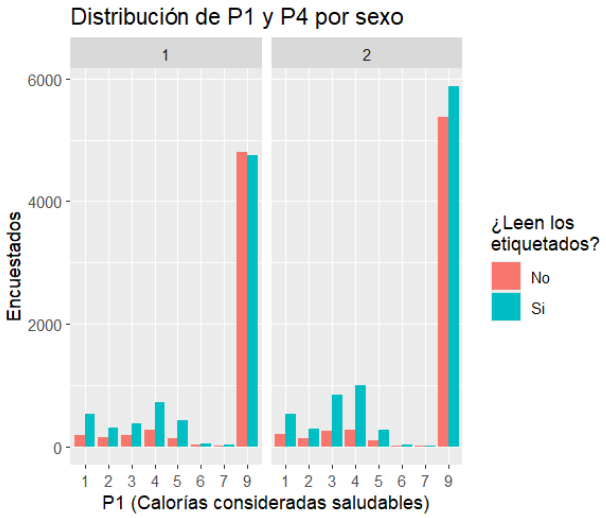








2.e. Otros análisis bivariados y/o multivariados.



2.e. Escribir los hallazgos e información útil de este paso.

- Muchisima gente no supo responder en P1.
- Contestaron más mujeres.
- Contestaron muy pocos de zona rural.
- Hay una gran cantidad de gente de estrato medio bajo.
- Los hombres tienen la idea de que deben de comer más que las mujeres.
- La gente de estrato mayor tiene la idea de que debe de comer más a comparación de los estratos menores.
- Todos saben que la comida tiene información nutrimental.
- Los hombres que no saben cuántas calorías deben de consumir al día leen menos los etiquetados a comparación de las mujeres.

3. Preparación de los Datos

Transformaciones e ingeniería de características, colocar criterio en que se basó cualquiera de las decisiones.

3.a. Omitir alguna característica, cambiar de numérica a texto o crear una nueva a partir de otras.

Tuve la duda de eliminar las columnas EST_DIS y UPM_DIS debido a que se me hacían columnas con números aleatorios y no sabía que significaban, sim embargo, le pregunté a ChatGPT qué significaban estas columnas, este fue el resultado:

- EST_DIS (Estrato de diseño): Identifica el estrato al que pertenece una unidad de observación dentro del diseño muestral. Los estratos se definen agrupando unidades con características similares (como tamaño de localidad, región geográfica o nivel socioeconómico) para mejorar la precisión de las estimaciones y asegurar una representación adecuada de subgrupos de interés.
- UPM_DIS (Unidad Primaria de Muestreo de diseño): Corresponde a la unidad primaria seleccionada en la primera etapa del muestreo. Las UPMs suelen ser áreas geográficas como manzanas, localidades o AGEBs (Áreas Geoestadísticas Básicas). Cada UPM contiene varias unidades secundarias (como hogares o personas) que se seleccionan en etapas posteriores del muestreo.

Después de leer esto, puedo darme cuenta que en efecto, no sirven de nada, las voy a borrar:

```
> datosMasLimpios3 <- datosMasLimpios2 %>% select(-EST_DIS, -UPM_DIS)
> str(datosMasLimpios3)
'data.frame':        28112 obs. of  9 variables:
 $ P1     : int  9 9 9 1 1 9 9 1 9 9 ...
 $ P3     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ P4     : chr  "Si" "No" "Si" "Si" ...
 $ EDAD   : int  73 55 51 34 46 43 64 20 65 60 ...
 $ SEXO   : int  2 1 2 2 1 2 1 1 1 1 ...
 $ ENT    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ DOMINIO: int  1 1 1 1 1 1 1 1 1 1 ...
 $ REGION : int  2 2 2 2 2 2 2 2 2 2 ...
 $ ESTRATO: int  3 3 3 3 3 2 3 3 3 3 ...
```

Ahora, la columna P3 solo contiene numeros 1, antes pensaba que esa columna era importante porque demuestra que todos saben que la comida tiene información nutrimental, sin embargo, he determinado que esa columna no discrimina nada como tal así que será eliminada:

```
> datosMasLimpios4 <- datosMasLimpios3 %>% select(-P3)
> str(datosMasLimpios4)
'data.frame':        28112 obs. of  8 variables:
$ P1     : int  9 9 9 1 1 9 9 1 9 9 ...
$ P4     : chr  "Si" "No" "Si" "Si" ...
$ EDAD   : int  73 55 51 34 46 43 64 20 65 60 ...
$ SEXO   : int  2 1 2 2 1 2 1 1 1 1 ...
$ ENT    : int  1 1 1 1 1 1 1 1 1 1 ...
$ DOMINIO: int  1 1 1 1 1 1 1 1 1 1 ...
$ REGION : int  2 2 2 2 2 2 2 2 2 2 ...
$ ESTRATO: int  3 3 3 3 3 2 3 3 3 3 ...
```

Para no tener problemas más adelante, tomé la decisión de convertor todos los datos a factores a excepción de la edad, también fue recomendación de la maestra:

```
> datosMasLimpios4$P1 <- as.factor(datosMasLimpios4$P1)
> datosMasLimpios4$P4 <- as.factor(datosMasLimpios4$P4)
> datosMasLimpios4$SEXO <- as.factor(datosMasLimpios4$SEXO)
> datosMasLimpios4$ENT <- as.factor(datosMasLimpios4$ENT)
> datosMasLimpios4$DOMINIO <- as.factor(datosMasLimpios4$DOMINIO)
> datosMasLimpios4$REGION <- as.factor(datosMasLimpios4$REGION)
> datosMasLimpios4$ESTRATO <- as.factor(datosMasLimpios4$ESTRATO)
> str(datosMasLimpios4)
'data.frame':        28112 obs. of  8 variables:
$ P1     : Factor w/ 8 levels "1","2","3","4",..: 8 8 8 1 1 8 8 1 8 8 ...
$ P4     : Factor w/ 2 levels "No","Si": 2 1 2 2 1 2 1 2 1 2 ...
$ EDAD   : int  73 55 51 34 46 43 64 20 65 60 ...
$ SEXO   : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 1 1 ...
$ ENT    : Factor w/ 32 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
$ DOMINIO: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
$ REGION : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 2 2 2 2 2 2 ...
$ ESTRATO: Factor w/ 4 levels "1","2","3","4": 3 3 3 3 3 2 3 3 3 3 ...
```

3.b. Sustitución (imputación).

3.b.i. De los valores NA.

El dataset ya no cuenta con valores NA:

```
> sum(is.na(datosMasLimpios4))
[1] 0
```

3.b.ii. De las respuestas de No sabe/No responde (9), si se considera apropiado o no.

Anteriormente, consideraba las siguientes estrategias respecto a esas respuestas:

- Imputación aleatoria.
- Eliminar las filas que contengan un 9.
- Sustituirlos por la moda.

Sin embargo, la maestra me dió un buen consejo, el cual es convertir esos valores a 0, la razón es la siguiente:

- El 0 tiene mayor significancia al 9 respecto a que no exista un valor.
- Es un valor fuera del rango normal ya que no pertenece a los números naturales.
- Preservamos los adtos en lugar de eliminarlos.
- Hay modelos que admiten el 0 como valor válido.

```
> datosMasLimpios5 <- datosMasLimpios4
> datosMasLimpios5$P1 <- as.numeric(as.character(datosMasLimpios5$P1))
> datosMasLimpios5$P1[datosMasLimpios5$P1 == 9] <- 0
> datosMasLimpios5$P1 <- as.factor(datosMasLimpios5$P1)
> str(datosMasLimpios5)
'data.frame':        28112 obs. of  8 variables:
$ P1     : Factor w/ 8 levels "0","1","2","3",..: 1 1 1 2 2 1 1 2 1 1 ...
```

```
$ P4     : Factor w/ 2 levels "No","Si": 2 1 2 2 1 2 1 2 1 2 ...
$ EDAD   : int  73 55 51 34 46 43 64 20 65 60 ...
$ SEXO   : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 1 1 ...
$ ENT    : Factor w/ 32 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
$ DOMINIO: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
$ REGION : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 2 2 2 2 2 2 ...
$ ESTRATO: Factor w/ 4 levels "1","2","3","4": 3 3 3 3 3 2 3 3 3 3 ...
```

3.c. Consideraciones en variables categóricas.

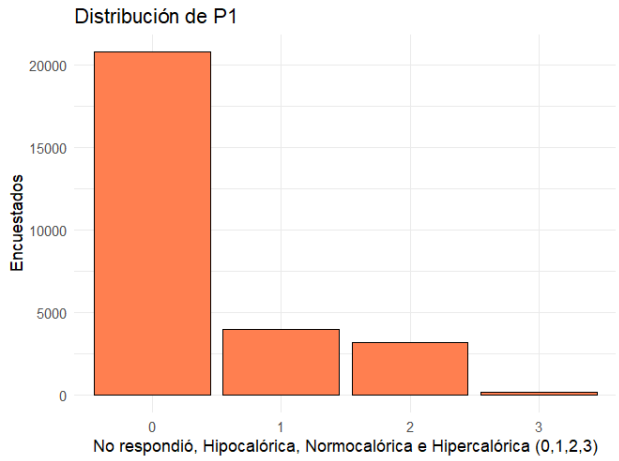
3.c.i. Agrupar para no manejar demasiados valores.

En la columna P1 tenemos bastantes posibles valores, lo cual se me hace un poco excesivo, para que nuestra información sea más facil de manejar y más significativa, ajustaré la información respecto a los tipos de dieta que una persona en México necesita en promedio:

- Hipocalórica (1) - menos de 1,800 calorias.
- Normocalórica (2) - entre 1,800 y 2,500 calorias.
- Hipercalórica (3) - más de 2,500 calorias.

```
> datosMasLimpios5$P1 <- as.numeric(as.character(datosMasLimpios5$P1))
> datosMasLimpios6 <- datosMasLimpios5 %>% mutate(P1 = case_when(P1 == 0 ~ 0, P1 >= 1 &
↪  P1 <= 3 ~ 1, P1 >= 4 & P1 <= 5 ~ 2, P1 >= 6 & P1 <= 7 ~ 3,))
> datosMasLimpios6$P1 <- as.factor(datosMasLimpios6$P1)
> table(datosMasLimpios6$P1)

    0     1     2     3
20797  3966  3181   168
```



Versión mejorada de P1, podemos ver que la gente tiende a comer poquito.

Planeaba cambiar la edad a caracter en lugar de que sea entera pero le pregunté a ChatGPT si valía la pena o no y me respondió lo siguiente:

- Mantén `EDAD` como numérica (`int`) si planeas usarla en cálculos o modelos.
- Convierte a `factor` o `character` solo si la vas a tratar como categoría.

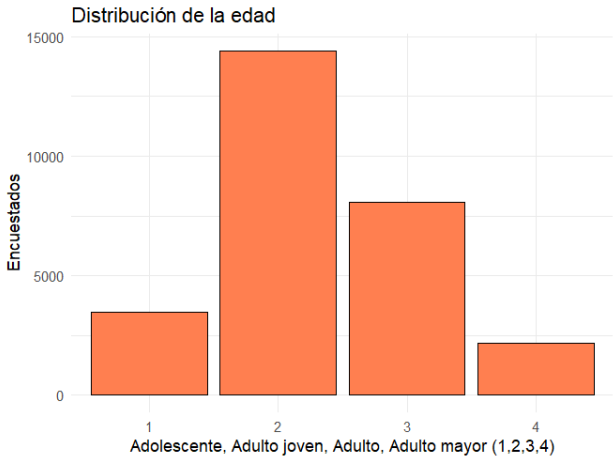
3.c.ii. Eliminar renglones con menos de 10 ocurrencias o agruparlos.

En la variable de edad tenemos muchos valores distintos y hay bastantes con pocas ocurrencias, creo que vale la pena que agrupemos las edades en grupos y despues convertir la variable a factor, los posibles valores son los siguientes:

- Adolescente (2) - 15 a 24 años.
- Adulto joven (3) - 25 a 44 años.
- Adultos (4) - 45 a 64 años.
- Adultos mayores (5) - 65+ años.

```
> datosMasLimpios7 <- datosMasLimpios6 %>% mutate(EDAD = case_when(EDAD >= 15 & EDAD <=
↪  24 ~ 1, EDAD >= 25 & EDAD <= 44 ~ 2, EDAD >= 45 & EDAD <= 64 ~ 3, EDAD >= 65 ~ 4,))
> datosMasLimpios7$EDAD <- as.factor(datosMasLimpios7$EDAD)
> table(datosMasLimpios7$EDAD)

   1     2     3     4
3454 14404  8073  2181
```



Ahora podemos ver que la mayoría son adultos jovenes.

Ahora todo nuestro dataset está en factor y honestamente no me gusta eliminar filas, siento que no vale la pena perder información y siempre hay que ver qué hacemos con esos datos que tal vez parezca que tenemos de sobra.

```
> str(datosMasLimpios7)
'data.frame':        28112 obs. of  8 variables:
$ P1     : Factor w/ 4 levels "0","1","2","3": 1 1 1 2 2 1 1 2 1 1 ...
$ P4     : Factor w/ 2 levels "No","Si": 2 1 2 2 1 2 1 2 1 2 ...
$ EDAD   : Factor w/ 4 levels "1","2","3","4": 4 3 3 2 3 2 3 1 4 3 ...
$ SEXO   : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 1 1 ...
$ ENT    : Factor w/ 32 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
$ DOMINIO: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
$ REGION : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 2 2 2 2 2 2 ...
$ ESTRATO: Factor w/ 4 levels "1","2","3","4": 3 3 3 3 3 2 3 3 3 3 ...
```

3.c.iii. Crear variables binarias.

Honestamente no veo la necesidad de crear variables binarias.

3.d. Consideraciones en variables cuantitativas, sean discretas o continuas.

3.d.i. Omitir el percentil 1 y el 99 de los datos, para disminuir outliers.

Eso no se puede hacer debido a que todas nuestras variables son factores.

3.d.ii. Crear categorías por ejemplo en edad si es bebe, niño, adolescente, adulto y adulto mayor.

Eso ya lo hicimos.

3.e. Realizar actividades de entendimiento de los datos, si se considera necesario ante los cambios.

Los datos que tenemos ya son faciles de entender, los únicos que no entendiamos eran EST DIS y UPM DIS.

4. Modelado

La sección de conocimiento previo se omitirá debido a que solo soy un integrante realizando el proyecto.

4.b. Experimentos.

4.b.i. Especificar por algoritmo las variables a usar, así como si se van a escalar (normalizar o estandarizar) o no.

- Algoritmos descartados:
  - kNN: Todos mis datos son factores y no va a funcionar.
  - Arboles de regresión: Mi variable dependiente no es continua.
  - Regresión lineal: Mi variable dependiente no es continua.
- Algoritmos aprobados:
  - Naive Bayes: Funciona muy bien con variables categóricas, la unica desventaja es que el algoritmo va a suponer que todas las variables son independientes entre sí, lo cual no es muy realista pero la verdad no creo que me afecte. La variable a usar es `P4` y no se va a escalar porque no es una variable numérica continua.
  - Arboles de decisión: Son ideales para trabajar con variables categóricas y con respuestas de "Sí" y "No", los datos no se escalan prque se dividen los datos según condiciones lógicas.
  - Regresión logística: Se puede modelar la probabilidad de que ocurra P4 utilizando una función logística, la estandarización ya la maneja internamente este algoritmo.
  - Redes neuronales: Podemos convertir nuestra variable P4 a binaria y P1 a numérica, despues de hacer eso regresamos a las variables a su estado anterior, como usaremos la función sigmoide, sí hay que escalar los datos.
  - Máquina de vector soporte: Funciona bastante bien con variables binarias como "Sí" y "No", las otras variables estan bastante preprocesadas y eso puede permitirnos encontrar el mejor hiperplano, como este algoritmo utiliza distancias, sí tenemos que escalar los datos.

4.b.ii. Explicar la estrategia de creación de CE, CV y CP, junto al método de remuestreo a utilizar.

Lo que se realizará para todos los algoritmos a probar es lo siguiente:

- Separar mi dataset una sola vez en 70 % CE y 30 % CP, me parece que esa es la forma más balanceada de hacerlo por los resultados que vi a lo largo del curso.
- Al final se usará el mismo CP para todos los modelos.
- Para cada CE de cada algoritmo, podrá variar el método de remuestreo.

Naive Bayes: Realizaremos de nuestro CE un remuestreo de validación de Monte Carlo ya que busco que el resultado simule la vida real lo más posible.

Arboles de decisión: Utilizaré validación cruzada porque esta vez quiero usar todos los datos, tanto para entrenar como para validar.

Regresión logística: Estaba pensando usar validación cruzada dejando uno fuera pero con un dataset como el nuestro creo que mi laptop no lo soportará, usaré el default (validación cruzada).

Redes neuronales: Utilizaré validación cruzada porque quería usar validación de Monte Carlo pero ya no me da tanta confianza. Sin embargo, la función que planeaba usar, la cual es `neuralnet()`, no soporta `caret` y no se puede hacer la validación cruzada, intenté hacer la validación cruzada a mano pero se trabó mi computadora, este es el código que utilicé:

```
CE$P4Binaria <- ifelse(CE$P4 == "Si", 1, 0)
CE$P1 <- as.numeric(as.character(CE$P1))
k <- 10
CE$fold <- sample(1:k, nrow(CE), replace = TRUE)
for (i in 1:k){
    CEentrena <- CE[CE$fold != i, ]
    CEvalida  <- CE[CE$fold == i, ]
    clasifRedNeuronal <- neuralnet(
    formula = P4Binaria ~ P1,
    data = CE,
    hidden = 3,
    linear.output = FALSE
    )
}
```

No pude usar remuestreo para redes neuronales, eso tal vez afecte a mi modelo...

Para el uso de redes neuronales, se tiene que elegir una variable con la que se basará P4 para logar una predicción, para esto, con ayuda de ChatGPT aprendí un poco de estadística inferencial y encontre algo llamado la prueba de chi-cuadrada, lo cual evalúa si existe una asociación entre dos variables, eso se determina en base a algo llamado p-value. Vamos a realizar esta prueba con todas nuestras variables para elegir la mejor:

```
> variables <- c("P1", "EDAD", "SEXO", "ENT", "DOMINIO", "REGION", "ESTRATO")
> resultados <- sapply(variables, function(var) {test <- chisq.test(table(CE[[var]],
↪  CE$P4))test$p.value})
> resultados
```

```
           P1          EDAD          SEXO           ENT
5.557623e-167  1.250937e-05  5.683843e-05  2.856345e-85
     DOMINIO        REGION       ESTRATO
4.372898e-02  2.309008e-16  3.100018e-12
```

La variable ganadora es P1, está increible esta prueba, no sabía que se podía calcular la asociación de una variable con otra tan facil.

Maquina de vector soporte: Para este algoritmo haremos algo diferente, utilizaremos una validación cruzada repetida, la cual consiste en repetir k-fold varias veces con diferentes particiones, lo cual promete ser más estable.

4.b.iii. Definir los valores de los hiperparámetros a probar.

- Para todos los algoritmos:
  - `metric = kappa`: Quiero revisar el resultado con una métrica estricta.
- Naive Bayes:
  - `laplace = 1`: Para proporcionar un suavizado simple.
  - `number = 10`: Diez repeticiones.
  - `p = 0.70`: Mantener silimitud con nuestra división original.
- Arboles de decisión:
  - `number = 10`: Diez ”folds” para más precisión.
- Regresión logística:
  - `number = 10`: Diez ”folds” para más precisión.
  - `family = "binomial"`: Tipo de regresión binaria.
- Redes neuronales:
  - `formula = P4Binaria~P1Scaled`: Se usa P1 por lo demostrado anteriormente.
  - `hidden = 3`: Número de neuronas en la capa oculta.
- Maquina de vector soporte:
  - `preProcess = c(''center'', ''scale'')`: Se escalan los datos.
  - `sampling = "smote"`: Nos ayuda si las clases están desbalanceadas, lo cual es nuestro caso.
  - `number = 10`: Diez ”folds” para más precisión.
  - `repeat = 3`: Repite la validación cruzada de 10 ”folds” tres veces.

4.b.iv. Construir los modelos de todos los algoritmos bajo las condiciones establecidas.

Naive Bayes

```
> ctrlNaiveBayes <- trainControl(method = "LGOCV", number = 10, p = 0.70, summaryFunction
↪  = defaultSummary, savePredictions = "final")
> modeloNaiveBayes <- train(P4 ~ ., data = CE, method = "naive_bayes", trControl =
↪  ctrlNaiveBayes, metric = "Kappa", tuneGrid = data.frame(laplace = 1, usekernel =
↪  FALSE, adjust = 1))
```

```
> print(modeloNaiveBayes)
    Accuracy   Kappa
    0.5880908  0.1620114
```

Hasta ahora tristemente Naive Bayes no nos dió buenos resultados, el Kappa es muy bajo.

Arboles de Decisión

```
> ctrlArbolDecision <- trainControl(method = "cv", number = 10, summaryFunction =
↪  defaultSummary, savePredictions = "final")
> modeloArbolDecision <- train(P4 ~ ., data = CE, method = "rpart", trControl =
↪  ctrlArbolDecision, metric = "Kappa")
> print(modeloArbolDecision)
    cp           Accuracy   Kappa
    0.006734405  0.5954050  0.10278999
    0.007206994  0.5870725  0.10838429
    0.011735980  0.5743690  0.07876037

Kappa was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.007206994.
```

Nuestro Kappa salió aun más bajo, usar arboles de decisión al parecer no conviene.

Regresión Logística

```
> ctrlRegLog <- trainControl(method = "cv", number = 10, summaryFunction =
↪  defaultSummary, savePredictions = "final")
> modeloRegLog <- train(P4 ~ ., data = CE, method = "glm", family = "binomial", trControl
↪  = ctrlRegLog, metric = "Kappa")
> print(modeloRegLog)
    Accuracy   Kappa
    0.6168002  0.1904014
```

Nuestro Kappa no es tan grande como yo quisiera pero con este algoritmo logramos subirlo.

Red Neuronal

```
> CE$P4Binaria <- ifelse(CE$P4 == "Si", 1, 0)
> CE$P1 <- as.numeric(as.character(CE$P1))
> CE$P1Scaled = as.numeric(scale(CE$P1, scale = TRUE))
> mean(CE$P1Scaled)
[1] 4.813893e-17
> sd(CE$P1Scaled)
[1] 1
> clasifRedNeuronal <- neuralnet(formula = P4Binaria ~ P1Scaled, data = CE, hidden = 3,
↪  linear.output = FALSE, stepmax = 1e6)
> predicciones <- compute(clasifRedNeuronal, data.frame(P1Scaled =
↪  CE$P1Scaled))$net.result
> pred_clase <- ifelse(predicciones > 0.5, 1, 0)
> conf <- confusionMatrix(factor(pred_clase, levels = c(0,1)), factor(CE$P4Binaria,
↪  levels = c(0,1)))
> print(conf$overall["Kappa"])
```

```
Kappa
    0
```

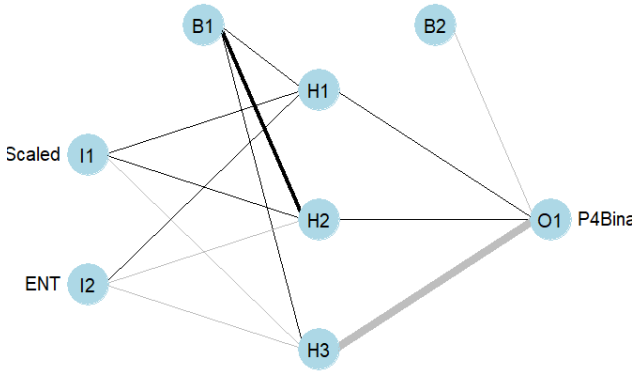
Obtuvimos un Kappa = 0, eso quiere decir que la red neuronal clasificó todo como 1 o como 0. Eso está pésimo, intenté investigar que pasó y me di cuenta que la proporción de las clases está desbalanceada:

```
> prop.table(table(CE$P4Binaria))
    0         1
0.4301032 0.5698968
```

De nuestras variables evaluadas con la prueba de chi-cuadrada, elegiré una variable extra para usarla en la red neuronal, la variable con más asociación aparte de P1 es ENT con un p-value de 2.8e-85.

```
> CE$P4Binaria <- ifelse(CE$P4 == "Si", 1, 0)
> CE$P1 <- as.numeric(as.character(CE$P1))
> CE$ENT <- as.numeric(as.character(CE$ENT))
> CE$P1Scaled = as.numeric(scale(CE$P1, scale = TRUE))
> clasifRedNeuronal <- neuralnet(formula = P4Binaria ~ P1Scaled + ENT, data = CE, hidden
↪  = 3, linear.output = FALSE, stepmax = 1e6)
> predicciones <- compute(clasifRedNeuronal, data.frame(P1Scaled = CE$P1Scaled, ENT =
↪  CE$ENT))$net.result
> pred_clase <- ifelse(predicciones > 0.5, 1, 0)
> conf <- confusionMatrix(factor(pred_clase, levels = c(0,1)), factor(CE$P4Binaria,
↪  levels = c(0,1)))
> print(conf$overall["Kappa"])
     Kappa
0.07998979
```

Tanto tiempo implementando este algoritmo para que me diera el Kappa más bajo hasta ahora, que mal, así quedó la red neuronal en forma visual:



Maquina de Vector Soporte

```
> ctrlSVM <- trainControl(method = "repeatedcv", number = 10, repeats = 3, sampling =
↪  "smote" )
> modeloSVM <- train(P4 ~ ., data = CE, method = "svmLinear", preProcess = c("center",
↪  "scale"), trControl = ctrlSVM, metric = "Kappa")
> print(modeloSVM)
Laptop trabada.........
```

El remuestreo que propuse no sirve en mi laptop, usaré validación cruzada estandar con 5 ”folds”:

```
> ctrlSVM <- trainControl(method = "cv", number = 5, sampling = "smote")
> modeloSVM <- train(P4 ~ ., data = CE, method = "svmLinear", preProcess = c("center",
↪  "scale"), trControl = ctrlSVM, metric = "Kappa")
> print(modeloSVM)
    Accuracy   Kappa
    0.5770111  0.1849493
```

Nuestro Kappa fue el segundo más alto, este algoritmo no está tan mal.

En todos los modelos que usamos ya obtuvimos un Kappa previo, vamos a comparar ese Kappa con el Kappa que obtendremos en la parte de evaluación, que ya es comparando con CP.

5. Evaluación

5.a. Probar todos los modelos con su hiperparámetro afinado en el CP.

Naive Bayes

```
> prediccionesNaiveBayes <- predict(modeloNaiveBayes, newdata = CP)
> confusionMatrix(prediccionesNaiveBayes, CP$P4)
Confusion Matrix and Statistics

            Reference
Prediction   No   Si
        No 1926 1647
        Si 1701 3159

Accuracy : 0.603
95% CI : (0.5925, 0.6135)
No Information Rate : 0.5699
P-Value [Acc > NIR] : 3.954e-10

Kappa : 0.1887

Mcnemar's Test P-Value : 0.3597

Sensitivity : 0.5310
Specificity : 0.6573
Pos Pred Value : 0.5390
Neg Pred Value : 0.6500
Prevalence : 0.4301
Detection Rate : 0.2284
Detection Prevalence : 0.4237
Balanced Accuracy : 0.5942

'Positive' Class : No
```

Arboles de Decisión

```
> prediccionesArbolDecision <- predict(modeloArbolDecision, newdata = CP)
> confusionMatrix(prediccionesArbolDecision, CP$P4)
Confusion Matrix and Statistics

              Reference
Prediction       No   Si
            No 2184 1957
            Si 1443 2849

Accuracy : 0.5968
95% CI : (0.5863, 0.6073)
No Information Rate : 0.5699
P-Value [Acc > NIR] : 2.936e-07

Kappa : 0.1916

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6022
Specificity : 0.5928
Pos Pred Value : 0.5274
Neg Pred Value : 0.6638
Prevalence : 0.4301
Detection Rate : 0.2590
Detection Prevalence : 0.4910
Balanced Accuracy : 0.5975

'Positive' Class : No
```

Regresión Logística

```
> prediccionesRegLog <- predict(modeloRegLog, newdata = CP)
> confusionMatrix(prediccionesRegLog, CP$P4)
Confusion Matrix and Statistics

           Reference
Prediction   No   Si
        No 1498 1054
        Si 2129 3752

Accuracy : 0.6226
95% CI : (0.6121, 0.6329)
No Information Rate : 0.5699
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.201

Mcnemar's Test P-Value : < 2.2e-16
```

```
Sensitivity : 0.4130
Specificity : 0.7807
Pos Pred Value : 0.5870
Neg Pred Value : 0.6380
Prevalence : 0.4301
Detection Rate : 0.1776
Detection Prevalence : 0.3026
Balanced Accuracy : 0.5969

'Positive' Class : No
```

Redes Neuronales

```
> CE$P1 <- as.numeric(as.character(CE$P1))
> mediaCEP1 <- mean(CE$P1)
> sdCEP1 <- sd(CE$P1)
> CP$P1 <- as.numeric(as.character(CP$P1))
> CP$P1Scaled <- (CP$P1 - mediaCEP1) / sdCEP1
> entradasCP <- data.frame(P1Scaled = CP$P1Scaled, ENT =
↪  as.numeric(as.character(CP$ENT)))
> prediccionesCP <- compute(clasifRedNeuronal, entradasCP)$net.result
> predClaseCP <- ifelse(prediccionesCP > 0.5, 1, 0)
> CP$P4Binaria <- ifelse(CP$P4 == "Si", 1, 0)
> confusionMatrix(factor(predClaseCP, levels = c(0, 1)), factor(CP$P4Binaria, levels =
↪  c(0, 1)))

Confusion Matrix and Statistics

            Reference
Prediction      0    1
            0  854  743
            1 2773 4063

Accuracy : 0.5831
95% CI : (0.5725, 0.5936)
No Information Rate : 0.5699
P-Value [Acc > NIR] : 0.007485

Kappa : 0.0868

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.2355
Specificity : 0.8454
Pos Pred Value : 0.5348
Neg Pred Value : 0.5944
Prevalence : 0.4301
Detection Rate : 0.1013
Detection Prevalence : 0.1894
Balanced Accuracy : 0.5404
```

```
'Positive' Class : 0
```

Maquina de Vector Soporte

```
> prediccionesSVM <- predict(modeloSVM, newdata = CP)
> confusionMatrix(prediccionesSVM, CP$P4)
Confusion Matrix and Statistics

            Reference
Prediction   No   Si
        No 2699 2534
        Si  928 2272

Accuracy : 0.5895
95% CI : (0.5789, 0.6)
No Information Rate : 0.5699
P-Value [Acc > NIR] : 0.0001446

Kappa : 0.2057

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.7441
Specificity : 0.4727
Pos Pred Value : 0.5158
Neg Pred Value : 0.7100
Prevalence : 0.4301
Detection Rate : 0.3201
Detection Prevalence : 0.6205
Balanced Accuracy : 0.6084

'Positive' Class : No
```

5.b. Hacer una tabla comparativa del rendimiento de los modelos.

Vamos a hacer una tabla que compare los valores Kappa solo sobre CE y los valores Kappa nuevos comparando con CP de los modelos que utilizamos:

Modelo	**Kappa$_{CE}$**	**Kappa$_{CP}$**
Naive Bayes	0.1620	0.1887
Árboles de decisión	0.1083	0.1916
Regresión logística	0.1904	0.2010
Redes neuronales	0.0799	0.0868
Máquina de vector soporte	0.1849	0.2057

5.c. Plantear el problema de clasificación final.

El ganador fue...

Con un valor Kappa de 0.2057, este es el modelo
que clasifica de forma más consistente y significativa.

5.d. Ejecutar pruebas para aumentar el rendimiento del algoritmo con mejor desempeño, procurando ser más sensible.

Mejoré el desempeño del algoritmo ganador realizando lo siguiente:

- Eliminé el remuestreo.
- Cambíe el método a `svmLinear2`, el cual según tiene un backend mejor optimizado.
- Creé un CE de muestra final para que ocupara menos datos.

Pueden ser cambios un poco bruscos pero en verdad iba muy lento antes y mejorá bastante haciendo todo eso, además, el Kappa no se vió tan afectado:

```
> CEMuestraFinal <- CE[sample(nrow(CE), 3000), ]
> ctrlSVMNuevo <- trainControl(method = "none", sampling = "smote")
> modeloSVMNuevo <- train(P4 ~ ., data = CEMuestraFinal, method = "svmLinear2",
↪  preProcess = c("center", "scale"), trControl = ctrlSVMNuevo, metric = "Kappa",
↪  tuneGrid = data.frame(cost = 1))
> prediccionesSVMNuevo <- predict(modeloSVMNuevo, newdata = CP)
> confusionMatrix(prediccionesSVMNuevo, CP$P4)
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 2619 2493
        Si 1008 2313

Accuracy : 0.5848
95% CI : (0.5742, 0.5954)
```

```
No Information Rate : 0.5699
P-Value [Acc > NIR] : 0.002855

Kappa : 0.1936

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.7221
Specificity : 0.4813
Pos Pred Value : 0.5123
Neg Pred Value : 0.6965
Prevalence : 0.4301
Detection Rate : 0.3106
Detection Prevalence : 0.6062
Balanced Accuracy : 0.6017

'Positive' Class : No
```

6. Despliegue

Implementar el código en R (crear el programa) con el mejor modelo y los siguientes pasos.

Nombre del archivo: despliegueFinalJulio.R

```
# Su directorio aquí
setwd("")
# Leer pruebaFinal.csv
datosFinales <- read.csv("datosTF.csv")
# Quitar P7
datosFinales0.5 <- subset(datos, select = -P7)
# Eliminar NA y personas que no saben o no respondieron en P4
datosFinales0.7 <- datosFinales0.5 %>% filter(!is.na(P4) & P4 != 9)
datosFinales0.8 <- datosFinales0.7 %>% filter(P4 != 9)
# Sustituir valores en variable dependiente
datosFinales1 <- datosFinales0.8 %>% mutate(P4 = case_when(P4 == 1 ~ "Si", P4 == 2 ~
↪  "No"))
# Borrar EST_DIS y UPM_DIS
datosFinales2 <- datosFinales1 %>% select(-EST_DIS, -UPM_DIS)
# Borrar P3
datosFinales3 <- datosFinales2 %>% select(-P3)
# Todos a factor
datosFinales3$P1 <- as.factor(datosFinales3$P1)
datosFinales3$P4 <- as.factor(datosFinales3$P4)
datosFinales3$SEXO <- as.factor(datosFinales3$SEXO)
datosFinales3$ENT <- as.factor(datosFinales3$ENT)
datosFinales3$DOMINIO <- as.factor(datosFinales3$DOMINIO)
datosFinales3$REGION <- as.factor(datosFinales3$REGION)
datosFinales3$ESTRATO <- as.factor(datosFinales3$ESTRATO)
# Convertir 9s a 0s
datosFinales4 <- datosFinales3
```

```
datosFinales4$P1 <- as.numeric(as.character(datosFinales4$P1))
datosFinales4$P1[datosFinales4$P1 == 9] <- 0
datosFinales4$P1 <- as.factor(datosFinales4$P1)
# Agrupar P1
datosFinales4$P1 <- as.numeric(as.character(datosFinales4$P1))
datosFinales5 <- datosFinales4 %>% mutate(P1 = case_when(P1 == 0 ~ 0, P1 >= 1 & P1 <= 3 ~
↪  1, P1 >= 4 & P1 <= 5 ~ 2, P1 >= 6 & P1 <= 7 ~ 3,))
datosFinales5$P1 <- as.factor(datosFinales5$P1)
# Agrupar EDAD
datosFinales6 <- datosFinales5 %>% mutate(EDAD = case_when(EDAD >= 15 & EDAD <= 24 ~ 1,
↪  EDAD >= 25 & EDAD <= 44 ~ 2, EDAD >= 45 & EDAD <= 64 ~ 3,EDAD >= 65 ~ 4,))
datosFinales6$EDAD <- as.factor(datosFinales6$EDAD)
# Dividimos CE y CP
set.seed(2711)
div = sample.split(datosMasLimpios7$P4, SplitRatio = 0.5)
CEFinalYa = subset(datosMasLimpios7, div == TRUE)
CPFinalYa = subset(datosMasLimpios7, div == FALSE)
# Creamos y aplicamos modelo
CEMuestraFinalFinal <- CE[sample(nrow(CEFinalYa), 3000), ]
ctrlSVMNuevoFinal <- trainControl(method = "none", sampling = "smote"
)
modeloSVMNuevoFinal <- train(P4 ~ ., data = CEMuestraFinalFinal, method = "svmLinear2",
↪  preProcess = c("center", "scale"), trControl = ctrlSVMNuevoFinal, metric = "Kappa",
↪  tuneGrid = data.frame(cost = 1))
prediccionesSVMNuevoFinal <- predict(modeloSVMNuevo, newdata = CPFinalYa)
confusionMatrix(prediccionesSVMNuevoFinal, CPFinalYa$P4)
```

7. Interpretación de los Resultados

7.a. Exponer todos los aprendizajes, que se consideran podrían ayudar a la persona de Negocio. Busca explicar e interpretar los resultados, apoyados con la información de todos los modelos.

Yo diría esto: los datos que me diste están bien, muestran correlaciones importantes y pudimos obtener un algoritmo que clasificó bien, sin embargo, si me hubieras dado datos donde las variables tuvieran más correlación a lo mejor pudieramos ser más precisos con las predicciones.

7.b. Conclusión dirigida a la experiencia que aportó el proceso realizado.

Fue un proyecto divertido y me ayudó bastante a aterrizar todo lo que vi en el semestre, la parte de modelado estuvo bastante pesada, creo que con esto me llevo una idea de lo que es hacer machine learning y la verdad es que me gustó bastante, me mantuve de cierta manera entretenido y no se me hizo tan pesado estar tantas horas trabajando, sí me veo en el futuro haciendo esto.

8. Referencias

Patiño, C. V. (2025). *Apuntes de Machine Learning*. Universidad Anáhuac Campus Norte.