Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Máquinas de aprendizaje

Reporte: Mejora de un modelo de machine learning

Docente: Abraham Sánchez López

Alumno	Matrícula
Taisen Romero Bañuelos	202055209

Proyecto: mejora de un modelo de ML

Los datos son sobre préstamos del banco. Lo que se pretende es predecir si un préstamo será aprobado o rechazado. Esto ya nos indica que debemos usar clasificación para poder predecir una clasificación binaria de **loan_status** igual a 1 o 0. Dado que es un problema de clasificación podemos hacer una selección de modelos a utilizar. Las opciones son:

- Regresión logística.
- Árboles de decisión.
- Random Forest.
- GBM.
- XGBoost.
- K-NN.
- Redes neuronales.

Usar árboles de decisión puede ser un buen acercamiento al inicio del desarrollo de la solución, aunque siendo franco, prefiero dedicar más tiempo al refinamiento de los modelos que a la selección de estos, por lo que de momento lo descartamos. Lo mismo sucede con la regresión logística, aunque como adelanto puedo decir que se usará en una parte del documento, por lo que tendremos que hacer un tratamiento de los datos particular para poder usar regresión. Como vimos en prácticas pasadas, un modelo de clustering puede ser bueno a primera vista pero al comparar su rendimiento con los mejores modelos se queda muy por detrás. Y finalmente, las redes neuronales las puedo utilizar en el proyecto final pero no recuerdo si para este también. Para no hacer más larga esta parte, elegiré RF y XGBoost como modelos iniciales, y después pienso usar el método de apilamiento de modelos para mejorar los resultados. En el KDD y el EDA se verá la importancia de usar árboles. Para el stacking pienso usar RF, regresión logística y XGBoost (GBM no lo considero por ahora debido a que sería el tercer modelo de árboles que usaría para el stacking, y quiero favorecer la diversidad de respuestas para obtener un modelo más completo). Para comparar el rendimiento de los modelos usaré Kappa y validación cruzada de k-Folds debido a que son las métricas que me parecen más objetivas.

Knowledge Discovery in Databases (KDD)

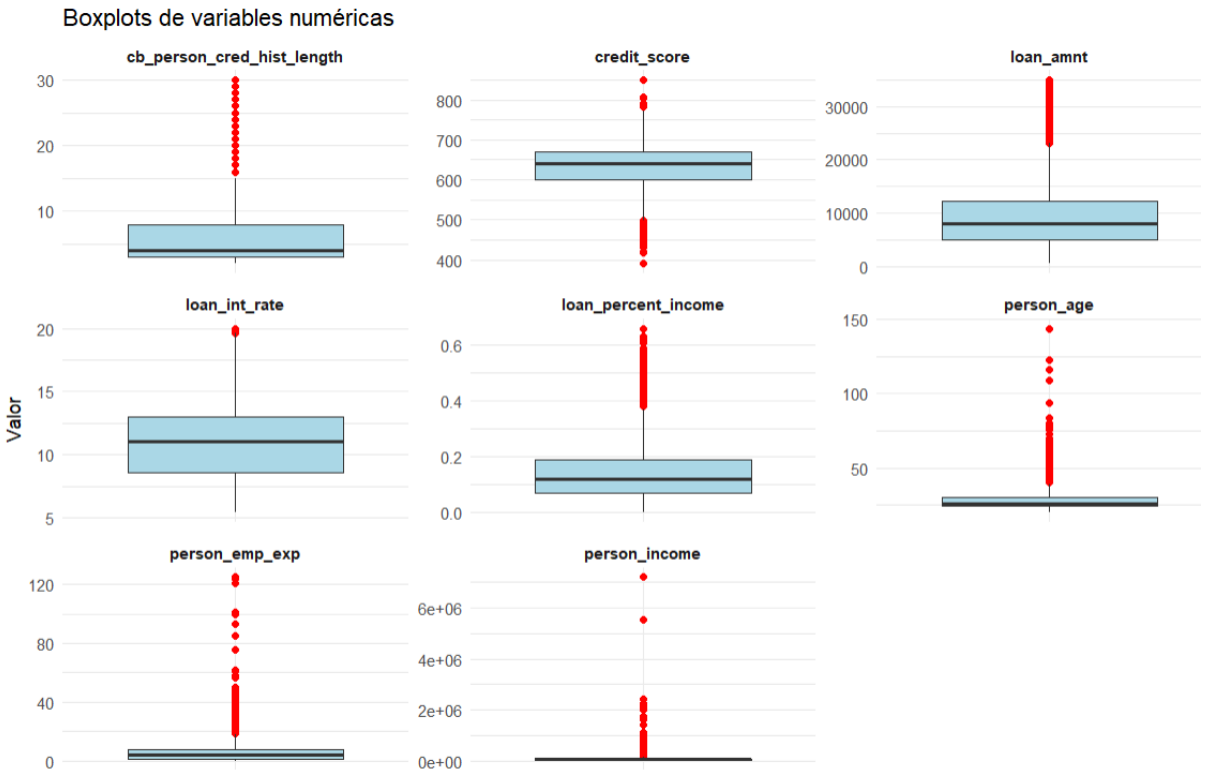
Para empezar haré un KDD y luego del paso 3 empezaré el EDA para conocer a fondo los datos. Recordemos que el KDD es una metodología compuesta por 5 pasos.

1. Data Selection.
2. Data Cleaning and Preprocessing.
3. Data Transformation and Reduction.
4. Data Mining.
5. Evaluation and Interpretation of Results.

```
> str(data)
'data.frame':   45000 obs. of  14 variables:
 $ person_age                    : num  22 21 25 23 24 21 26 24 24 21 ...
 $ person_gender                 : chr  "female" "female" "female" "female" ...
 $ person_education              : chr  "Master" "High School" "High School" "Bachelor" ...
 $ person_income                 : num  71948 12282 12438 79753 66135 ...
 $ person_emp_exp                : int  0 0 3 0 1 0 1 5 3 0 ...
 $ person_home_ownership         : chr  "RENT" "OWN" "MORTGAGE" "RENT" ...
 $ loan_amnt                     : num  35000 1000 5500 35000 35000 2500 35000 35000 35000 1600 ...
 $ loan_intent                   : chr  "PERSONAL" "EDUCATION" "MEDICAL" "MEDICAL" ...
 $ loan_int_rate                 : num  16 11.1 12.9 15.2 14.3 ...
 $ loan_percent_income           : num  0.49 0.08 0.44 0.44 0.53 0.19 0.37 0.37 0.35 0.13 ...
 $ cb_person_cred_hist_length    : num  3 2 3 2 4 2 3 4 2 3 ...
 $ credit_score                  : int  561 504 635 675 586 532 701 585 544 640 ...
 $ previous_loan_defaults_on_file: chr  "No" "Yes" "No" "No" ...
 $ loan_status                   : int  1 0 1 1 1 1 1 1 1 1 ...
> summary(data)
   person_age     person_gender      person_education   person_income     person_emp_exp   person_home_ownership   loan_amnt     loan_intent
 Min.   : 20.00   Length:45000       Length:45000       Min.   :   8000   Min.   :  0.00   Length:45000           Min.   :  500   Length:45000
 1st Qu.: 24.00   Class :character   Class :character   1st Qu.:  47204   1st Qu.:  1.00   Class :character       1st Qu.: 5000   Class :character
 Median : 26.00   Mode  :character   Mode  :character   Median :  67048   Median :  4.00   Mode  :character       Median : 8000   Mode  :character
 Mean   : 27.76                                        Mean   :  80319   Mean   :  5.41                          Mean   : 9583
 3rd Qu.: 30.00                                        3rd Qu.:  95789   3rd Qu.:  8.00                          3rd Qu.:12237
 Max.   :144.00                                        Max.   :7200766   Max.   :125.00                          Max.   :35000
 loan_int_rate   loan_percent_income cb_person_cred_hist_length  credit_score   previous_loan_defaults_on_file  loan_status
 Min.   : 5.42   Min.   :0.0000      Min.   : 2.000             Min.   :390.0   Length:45000                   Min.   :0.0000
 1st Qu.: 8.59   1st Qu.:0.0700      1st Qu.: 3.000             1st Qu.:601.0   Class :character               1st Qu.:0.0000
 Median :11.01   Median :0.1200      Median : 4.000             Median :640.0   Mode  :character               Median :0.0000
 Mean   :11.01   Mean   :0.1397      Mean   : 5.867             Mean   :632.6                                  Mean   :0.2222
 3rd Qu.:12.99   3rd Qu.:0.1900      3rd Qu.: 8.000             3rd Qu.:670.0                                  3rd Qu.:0.0000
 Max.   :20.00   Max.   :0.6600      Max.   :30.000             Max.   :850.0                                  Max.   :1.0000
```

El primer acercamiento resulta bastante revelador. Para empezar, hay que hacer un tratamiento del tipo de variables, pero más interesante es que con el summary() podemos notar que hay outliners muy fuertes en algunas variables. En variables como person_age bien podrían ser datos erróneos, pero en otras variables como loan_percent_income podrían ser valores inusuales pero que siguen siendo válidos. De hecho, creo que esta clase de desbalances son más representativos del mundo real ya que no es extraño que en muy pocas personas haya una acumulación desproporcionada de capital, estos burgueses vendrían siendo los outliers del mundo real. Y hablando de desbalances, aquí podemos empezar a preguntarnos si será necesario hacer una normalización o escalamiento de los datos. Pese a que los árboles toleran mejor los datos atípicos, los outliners del dataset son demasiado fuertes (y no son pocos), por lo que habrá que considerar esto en el EDA. También podemos notar algo curioso, loan_status tiene una media de 0.222, lo que indica que sólo el 22% de los casos son aprobados, lo que significa que está desbalanceado (al parecer la normalización no sólo sería beneficioso para los outliners).

Bueno, el plan de acción es sencillo, convertir algunas variables a factor, otras a dummies y explorar los outliers. Antes de tratar los outliners hice el tratamiento de variables para tener una mejor representación de los mismos sobre la versión final del dataset.



Ahora resulta más que evidente el desbalance de las clases y la fuerte presencia de los outliers. También podemos ver con más claridad que en algunas variables hay datos atípicos que pueden ser válidos. En loan_amnt muchos valores cercanos al tope (35,000) son frecuentes, pero no necesariamente incorrectos porque puede ser el máximo permitido por política. Esto refuerza la decisión de usar más árboles en el modelado, ya que si se trata de la naturaleza de los datos, los árboles son más aptos para tratar con este dataset ya que en principio se deberían ver menos afectados que un modelo de regresión, por ejemplo. De todas formas, hice un filtro basado en valores más realistas, ya que por ejemplo, una persona de 140 años de edad es tan rara que es preferible omitir para que no afecte al modelado. Inicialmente el filtro eliminaba aquellos datos que no pasaban el filtro, pero esto provocaba que se perdieran aproximadamente 1,200 observaciones, por lo que decidí que **si una variable excede un valor máximo razonable, se sustituye por ese máximo en vez de eliminar la fila**. Tal vez 1,200 observaciones no sean tantas comparadas con el total de 45,000 observaciones, pero dichos outliners podrían tener un valor informativo si se ajustan en lugar de eliminarse.

```
> str(data)
'data.frame':   45000 obs. of  14 variables:
```

Figura. 1. Dataset original.

```
> str(data)
'data.frame':   44864 obs. of  14 variables:
```

Figura. 2. Dataset luego de aplicar el filtro original.

```
> str(data)
'data.frame':   45000 obs. of  14 variables:
```

Figura. 3. Dataset luego de aplicar el filtro refinado.

Ahora que hemos hecho el tratamiento de los datos atípicos, aprovechemos para hacer un dataframe nuevo con el dataset modificado para que sea adecuado para un modelo de regresión logística. Así, tenemos tanto un dataset para árboles como para regresión.

Los pasos a seguir fueron duplicar el dataset, aplicar One-hot encoding y normalizar el dataset. Pero, ¿fue la normalización efectiva?, la respuesta rápida es no. Tanto los boxplots como la función summary() nos demuestran que los datos todavía están fuertemente desbalanceados.

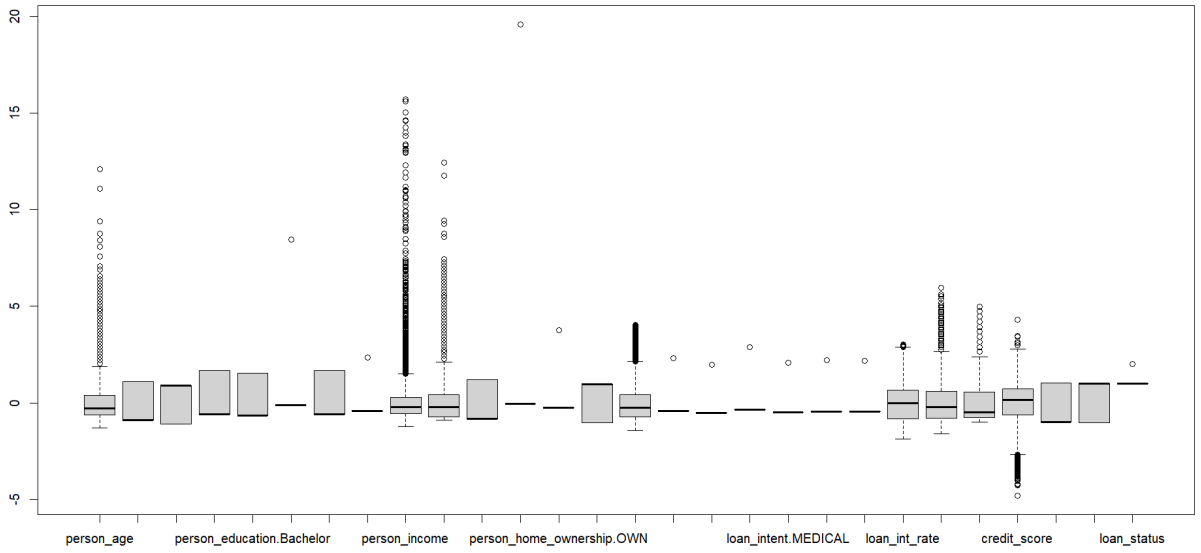


Figura. 4. Boxplot después de la normalización z-score.

Cuantitativamente podemos ver que la diferencia entre el máximo y la media y el promedio sigue siendo considerable en algunas variables.

```
> summary(data_reg_z) #Todavía hay outliers, los máximos se alejan mucho de la media
   person_age      person_gender.female person_gender.male person_education.Associate
 Min.   :-1.2981   Min.   :-0.9008      Min.   :-1.1101    Min.   :-0.604
 1st Qu.:-0.6290   1st Qu.:-0.9008      1st Qu.:-1.1101    1st Qu.:-0.604
 Median :-0.2944   Median :-0.9008      Median : 0.9008    Median :-0.604
 Mean   : 0.0000   Mean   : 0.0000      Mean   : 0.0000    Mean   : 0.000
 3rd Qu.: 0.3748   3rd Qu.: 1.1101      3rd Qu.: 0.9008    3rd Qu.: 1.656
 Max.   :12.0852   Max.   : 1.1101      Max.   : 0.9008    Max.   : 1.656
```

Figura. 5. Salida de la función summary sobre los datos normalizados.

Por eso consideré la posibilidad de que el escalado min-max diera mejores resultados, y así fue. Aunque seguían persistiendo los outliers, esta vez estaban en un rango más aceptable, por lo que estos serán los datos que use para el modelo de regresión.

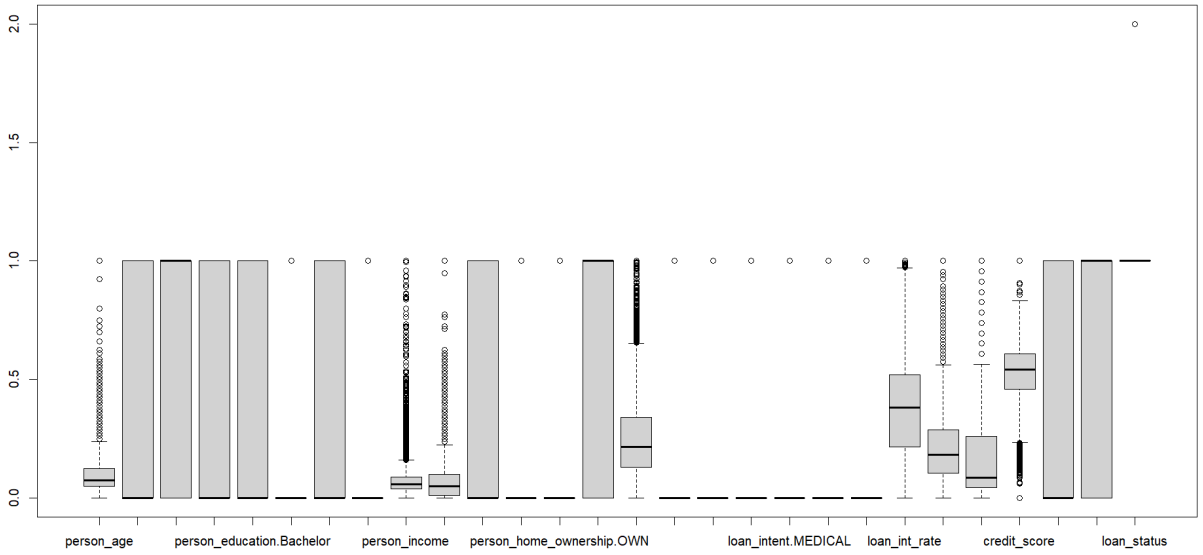


Figura. 6. Boxplot después del escalado.

```
> summary(data_reg)
   person_age     person_gender.female person_gender.male person_education.Associate
 Min.   :0.000   Min.   :0.000        Min.   :0.000      Min.   :0.0000
 1st Qu.:0.050   1st Qu.:0.000        1st Qu.:0.000      1st Qu.:0.0000
 Median :0.075   Median :0.000        Median :1.000      Median :0.0000
 Mean   :0.097   Mean   :0.448        Mean   :0.552      Mean   :0.2673
 3rd Qu.:0.125   3rd Qu.:1.000        3rd Qu.:1.000      3rd Qu.:1.0000
 Max.   :1.000   Max.   :1.000        Max.   :1.000      Max.   :1.0000
```

Figura. 7. Salida de la función summary sobre los datos escalados.

Exploratory data analysis (EDA)

Bien, para realizar el EDA decidí seguir la siguiente estructura y empezar a darle un rumbo al trabajo con algunas preguntas de investigación (por cierto, usé los datos para los modelos de árboles para esta parte del reporte ya que considero que son el punto fuerte del trabajo).

A) Estadísticas básicas (descriptivas):
   - Distribuciones (edad, ingresos, score crediticio...).
   - Medidas de tendencia central y dispersión (mean, std, min, max...).
   - Balanceo de clases (proporción de loan_status).
   - ¿Qué % de ingreso se pide en préstamo (loan_percent_income)?

B) Ploteos:
   - Histogramas para variables numéricas.
   - Boxplots (detectar outliners).
   - Matriz de correlación para variables numéricas.
   - Gráficas de pares (pairplot) para identificar clusters o relaciones.
   - Gráficos de densidad o violin plot (comparar score de aprobados o rechazados).

C) Exploración multivariada:
   - ¿Cómo influye la interacción entre score de crédito y % de ingreso en el préstamo?
   - ¿Se aprueban más préstamos con ciertos propósitos (loan_intent)?
   - ¿Qué combinaciones de educación y home_ownership tienden a correlacionarse con aprobación?

Los histogramas los compacté en un gráfico facetado para ahorrar tiempo. Podemos notar algo curioso. Para la variable loan_percent_income hay un sesgo hacia la derecha mostrando que la mayoría de la gente es algo conservadora con respecto a su deuda ya que la mayoría pide préstamos menores al 20% de su sueldo anual. También se puede notar que en loan_amnt hay varios picos, lo que podría reflejar un patrón de conducta de los clientes (como una preferencia por ciertos préstamos). Pero en términos más generales podemos decir que la mayoría de la población es relativamente joven (22-30 años) y con pocos años de experiencia. La juventud y la poca experiencia podrían ser la razón por la que el historial crediticio tiende a no ser mayor a los 5 años. También que la mayoría gana por debajo de los 150k anuales. El rango que podemos considerar aceptable para credit_score está entre 600 y 700.

Probablemente credit_score y loan_percent_income muy predictivas del estado del préstamo por lo que habrá que observar que nos dice la matriz de correlación sobre esto.

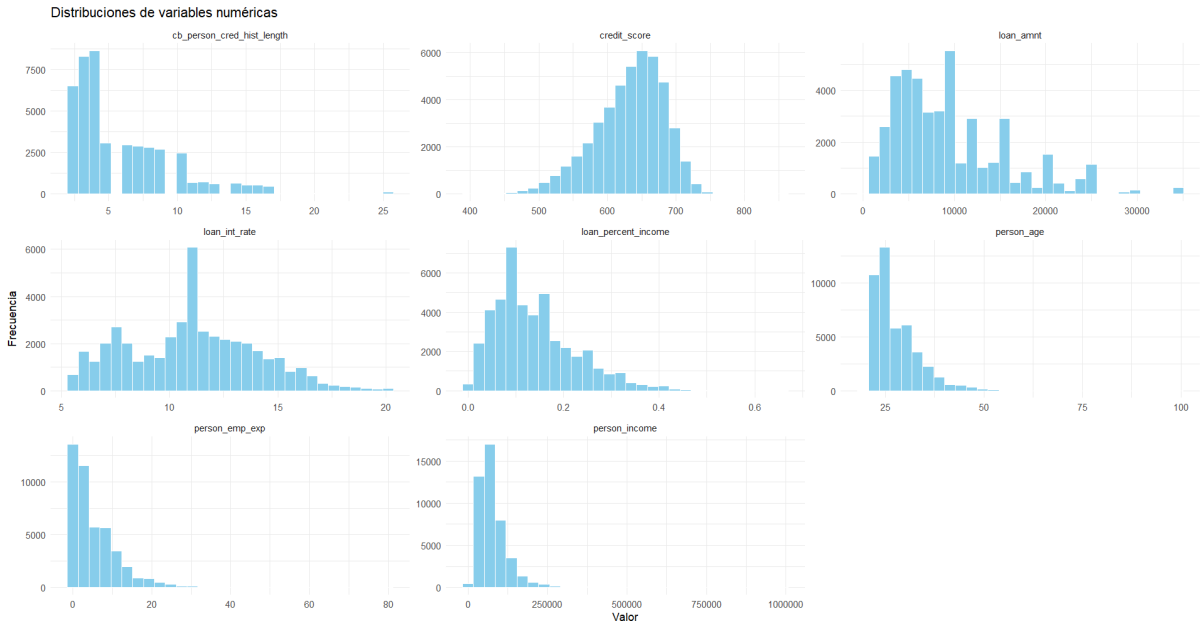


Figura. 8. Histogramas de variables numéricas.

Continuando con los ploteos, podemos notar que un mayor credit_score está positivamente asociado con la probabilidad de ser aprobado. También, en el boxplot podemos notar que aprobaron a personas que piden un porcentaje mayor del ingreso, lo que sugiere que otras variables pueden tener más peso en la decisión (o que el banco quería endeudar de por vida a alguien). De hecho, es posible que esas variables que tengan más peso puedan relacionarse con lo que notamos en gráfico de la figura 11. Parece que el propósito del préstamo influye en la aceptación, pues, los préstamos de menor riesgo (educación, venture) son más rechazados. Y finalmente, retomando una pregunta de investigación, parece que el nivel educativo no es un predictor fuerte para determinar la aprobación.

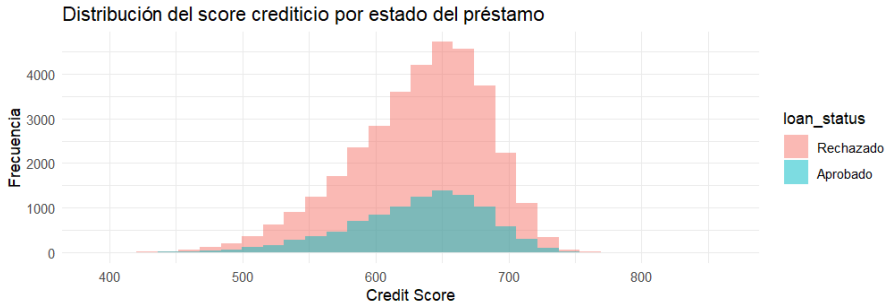


Figura. 9. Rechazados dominan en scores bajos (500–600).

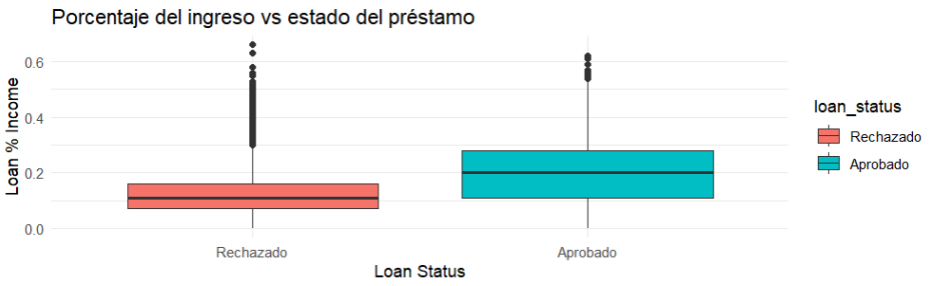


Figura. 10. Aprobados tienen proporciones mayores (mediana ≈ 20%)..

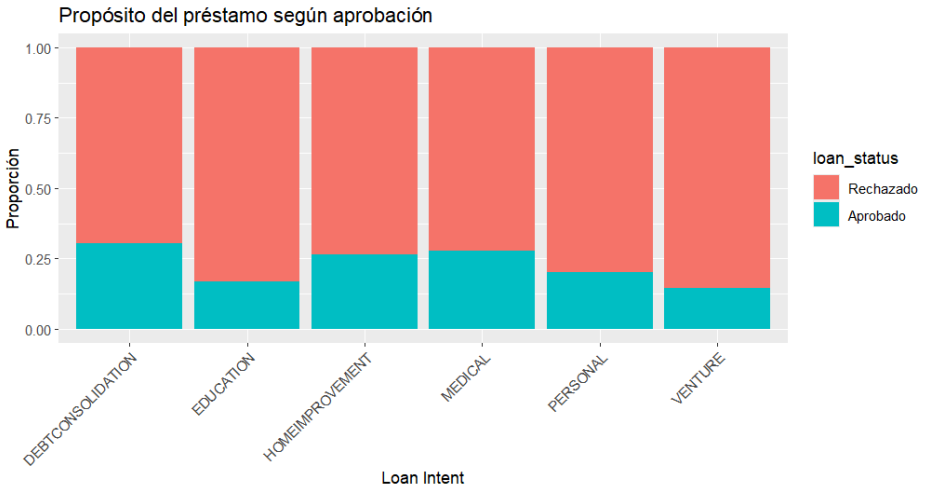


Figura. 11. Education y Venture son los que presentan mayor tasa de rechazo.

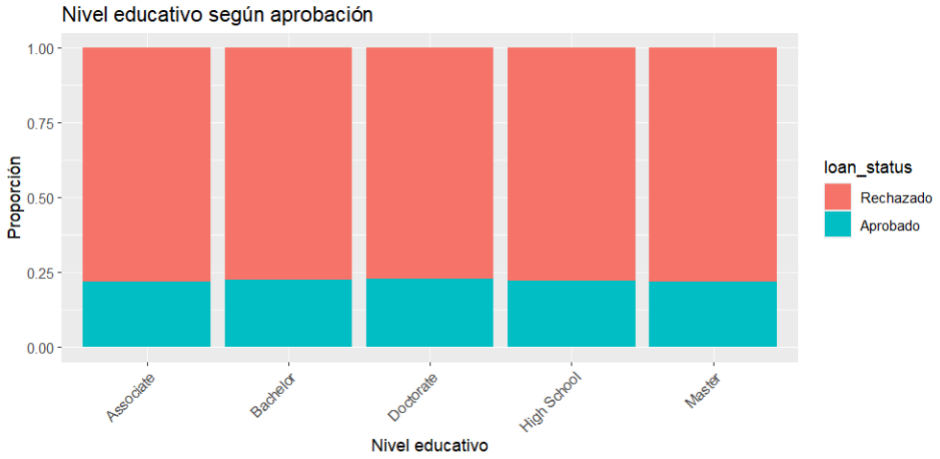


Figura. 12. La proporción entre aprobados y rechazados es muy similar en todos los niveles educativos.

Bien, pasemos a la correlación entre variables. Normalmente para esto me gusta usar un pair plot de la librería GGally ya que resume bien mucha información, sin embargo, en esta ocasión no pude generar el gráfico del conjunto completo debido a que mi equipo no terminaba de generar el gráfico. De todas formas, generé un pair plot de las variables numéricas, aunque creo que para efectos prácticos será más útil analizar la matriz generada en consola.

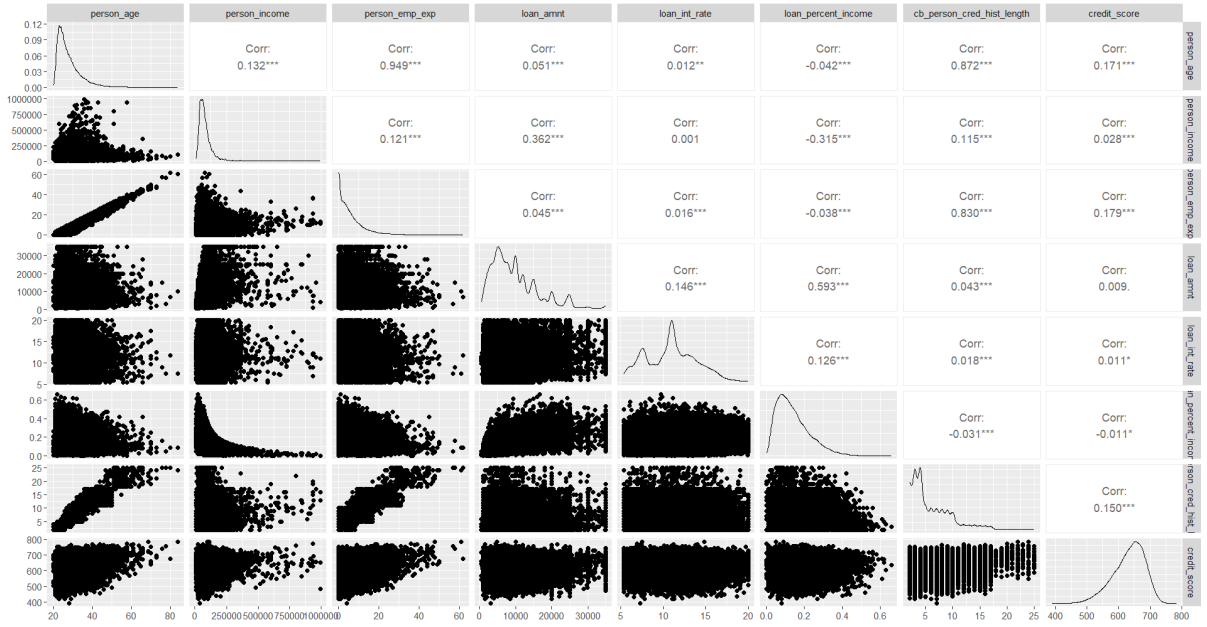


Figura. 13. Pair plot de numeric_vars.

La matriz de correlación nos indica que hay una correlación muy fuerte entre la edad y la experiencia (0.953). Además, estas dos variables también tienen una correlación muy fuerte con cb_person_cred_hist_length (aproximadamente 0.8 cada una). Si bien esto puede ser interesante a primera vista hay que considerar que las tres variables miden aspectos muy similares del tiempo de vida o trayectoria de una persona, por lo que aportan información repetida o muy parecida en el modelo (o sea, hay una posible **redundancia**). Las variables loan_amnt y loan_percent_income también presentan una correlación moderada (0.59), lo cual es lógico dado que los préstamos más altos representan mayor proporción del ingreso anual. En contraste, credit_score muestra correlaciones muy bajas con otras variables (<0.19), destacándose como un predictor independiente y potencialmente muy relevante para el modelado. Dado que Random Forest y XGBoost manejan bien multicolinealidad no será necesario eliminar variables para los modelos de árboles (pese a la redundancia que mencioné apenas).

Modelado

Como ya mencioné al inicio, en esta parte del documento no profundizaré tanto en los detalles para poder dedicarle más atención a la sección de mejora. Para comparar todos los modelos (incluyendo los mejorados) usaré k-Fold CV (con k=10) y Kappa como métricas de rendimiento. Usé k=10 porque esta cantidad de fold me permitirá capturar mejor la información (recordemos que el dataset consta de 45k observaciones). Aunque más adelante se reajustará este parámetro. Además, configuré el entrenamiento de RF con 200 árboles (cosa que me costó un tiempo de ejecución algo largo).

```
> # ----> Comparación de los modelos
> resamples_list <- resamples(list(RandomForest = modelo_rf_cv, XGBoost = modelo_xgb_cv))
> summary(resamples_list)

Call:
summary.resamples(object = resamples_list)

Models: RandomForest, XGBoost
Number of resamples: 10

Accuracy
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
RandomForest 0.9248889 0.9272222 0.9290000 0.9285333 0.9299444 0.9322222    0
XGBoost      0.9315556 0.9325556 0.9346667 0.9346000 0.9360000 0.9384444    0

Kappa
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
RandomForest 0.7704881 0.7775472 0.7824464 0.7813135 0.7847512 0.7910164    0
XGBoost      0.7940565 0.7977584 0.8038548 0.8038777 0.8077158 0.8154017    0
```

Figura. 14. Comparación de rendimiento de los modelos iniciales.

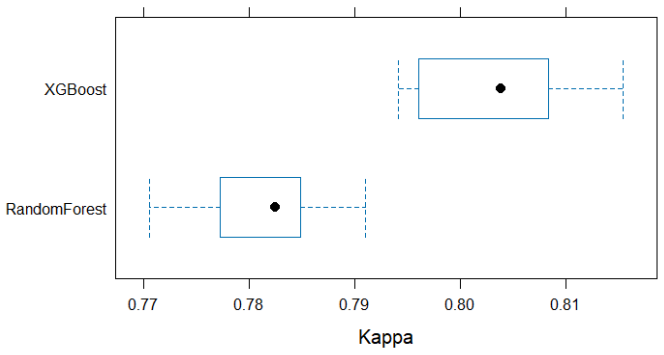


Figura. 15. Comparación gráfica. Observe que XGB tiene un kappa promedio más alto (~0.80) y una dispersión más baja, lo que indica mejor rendimiento y mayor estabilidad en las particiones.

Mejoras de rendimiento

La parte buena está por comenzar. La verdad, hacer el stacking fue más desafiante de lo que pensé. Hay que procurar que todos los modelos y todas las particiones de datos estén en los mismos términos y con hiper parámetros que no generen conflictos entre los resultados de un modelo y otro, de lo contrario, surgirán errores. Por ejemplo, tuve que convertir algunas variables del dataset de árboles a dummy porque si no lo hacía los resultados no serían compatibles con los del dataset de regresión.

Para la arquitectura del stacking me basé en las imágenes del PDF.

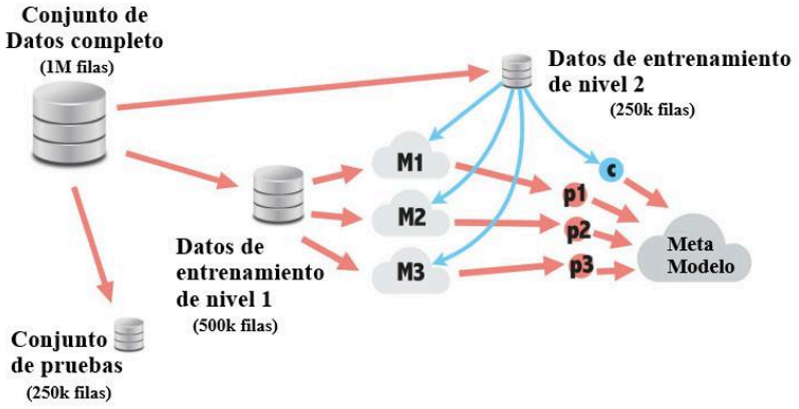


Figura. 16. Arquitectura de ejemplo de stacking.

Hice una partición del 50% para el nivel 1, 25% para el nivel 2 y 25% para la prueba final. Y para medir el rendimiento apliqué k-Fold CV (k=10) sólo al meta modelo porque ese modelo es el que da mejores resultados de todos los que están en el stacking. Cabe mencionar que el modelo de regresión no logró converger, lo que significa que no logró encontrar un resultado óptimo (recordar este "presagio"). Si bien esto podría generar preocupación, recordemos que los modelos de nivel 1 pueden ser menos rigurosos, de hecho, es el resultado de crear varios modelos simples pero con gran diversidad. Pero basta de cháchara, pasemos a los resultados obtenidos.

```
> print(conf_matrix_meta)
Confusion Matrix and Statistics

          Reference
Prediction  Rechazado Aprobado
  Rechazado      8498      519
  Aprobado        252     1981

               Accuracy : 0.9315
                 95% CI : (0.9266, 0.9361)
    No Information Rate : 0.7778
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7939

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

            Sensitivity : 0.9712
            Specificity : 0.7924
         Pos Pred Value : 0.9424
         Neg Pred Value : 0.8871
             Prevalence : 0.7778
         Detection Rate : 0.7554
   Detection Prevalence : 0.8015
      Balanced Accuracy : 0.8818

       'Positive' Class : Rechazado
```

Figura. 17. Matriz de confusión del meta modelo.

Como podemos ver, los resultados no fueron tan buenos como se podría haber esperado. Si comparamos el rendimiento de todos los modelos notaremos que el meta modelo fue ligeramente mejor que random forest.

```
> print(tabla_comparativa)
                  Modelo  Accuracy     Kappa
1          Random Forest 0.9285333 0.7813135
2                XGBoost 0.9346000 0.8038777
3 Stacking (Meta-Modelo) 0.9314667 0.7938809
```

Figura. 18. Tabla comparativa de los tres modelos.

Para este dataset en particular la métrica kappa es una métrica útil para datos desbalanceados (como los nuestros). Entonces, ¿por qué nuestro meta modelo no superó con creces a XGB?, me temo que podría deberse a que el meta modelo fue un modelo de regresión logística, y como vimos al inicio del documento, la regresión logística se puede ver afectada por los fuertes desbalances y outliers de nuestros datos. Como vimos en el modelo de nivel 1 de regresión, no se pudo alcanzar la convergencia incluso después de haber tratado los datos atípicos y escalado los datos. Esto le da mucho más peso a la decisión inicial de usar modelos de árboles al inicio.

Aunque mi decepción fue grande, creo que podemos extraer una información clave de estos resultados. La mejora de un modelo también va en dirección a lo que es apropiado para los datos. No por ser una técnica sofisticada implica que funcionará tan bien como se espera. Entonces, si lo pensamos, tenemos dos caminos a seguir. El primero es cambiar el meta modelo, en lugar de usar regresión podemos usar un

modelo de árboles (o incluso podríamos eliminar por completo la regresión de nuestro stacking); y el segundo es probar otro método de mejora del rendimiento, como usar una rejilla de ajuste de hiper parámetros para mejorar nuestro modelo de XGB. Debido a la cantidad de trabajo que representó el stacking y los resultados poco agradables, creo que optaré por el segundo camino.

Bueno, el uso del grid no fue tan efectivo como pensé, tomó mucho tiempo para entrenarse al grado de que pasaron 40 minutos y no terminaba. Incluso probé con usar los datos de entrenamiento de nivel 1 para usar sólo el 50% de los datos pero el tiempo de ejecución no mejoró.

La situación presente me incitó a aplicar PCA al conjunto de entrenamiento de nivel 1 para reducir todavía más los datos para analizar sin perder información relevante, y a su vez, decidí ajustar el grid para que sea menos demandante a la hora de ejecutar el modelo.

Tras 35 minutos aproximadamente, el modelo se entrenó por completo, haciendo evidente la necesidad de combinar técnicas de minería a nuestros ejercicios. Sin embargo, no todo son buenas noticias, el mejor valor de Kappa alcanzado fue de apenas 0.73.

```
> m_xgb_fine$bestTune   #Mejores hiperparámetros encontrados
    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
122     150         4 0.1   0.1              0.8                1       0.8
> max(m_xgb_fine$results$Kappa)  #Mejor valor de Kappa alcanzado
[1] 0.7309819
```

Figura. 19. Rendimiento del modelo que usó el 50% de los datos y al que se le aplicó PCA.

Es posible que el resultado no haya sido tan favorable debido a los ajustes que se tuvieron que hacer para que el modelo fuera entrenado en un tiempo razonable, aunque también es posible que usar el dataset de entrenamiento haya afectado negativamente. Me explico, para que los datos de los modelos de stacking fueran compatibles fue necesario convertir a dummy algunas variables, por lo tanto, esto aumenta el número de combinaciones que hacer ya que aumenta el número de variables. Incluso si esto no fuera cierto, esta observación me hizo darme cuenta de que realmente hay muchas variables desde un inicio (14 para ser exacto), por lo que ahora la jugada será trabajar con los datos de árboles (los que se usaron para entrenar el primer modelo de XGB). Aplicaré un **muestreo aleatorio estratificado** para mejorar los tiempos de ejecución y así poder usar un grid más robusto. También fue necesario balancear las muestras estratificadas porque habían muchas más observaciones de “rechazado” que de “aprobado”, y esto podría provocar que el modelo tenga dificultades para aprender adecuadamente sobre la clase minoritaria.

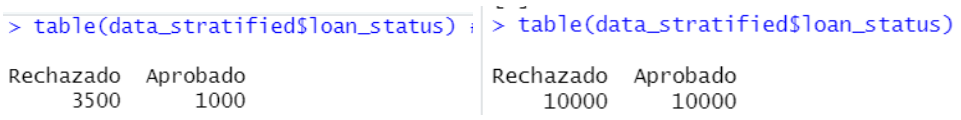


Figura. 20. Antes y después de la clase objetivo para el muestreo estratificado.

No funcionó usar el remuestreo y hacer más robusto el grid. Pasaron dos horas y no terminaba de entrenar el modelo. Sin embargo, al "podar" algunos hiper parámetros y reducir a la mitad el k de k-Fold por fin logré mejorar el rendimiento del modelo. Similarmente a Sísifo empujando su roca, cada ajuste y cada modelo nos enseñan que el valor está en el intento, aunque el resultado no siempre alcance la cima. Sin embargo yo no soy Sísifo y si logré mejorar un poco el rendimiento del modelo, así que pasemos a explicar los resultados.

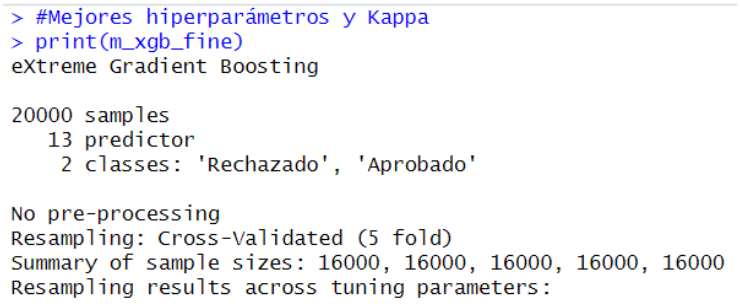


Figura. 21. Rendimiento de XGB con muestreo estratificado pt. 1.

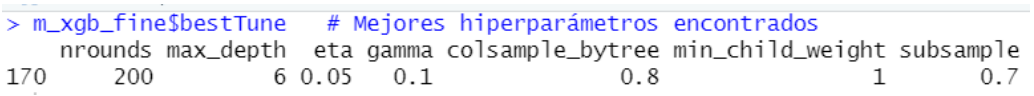


Figura. 22. Rendimiento de XGB con muestreo estratificado pt. 2.

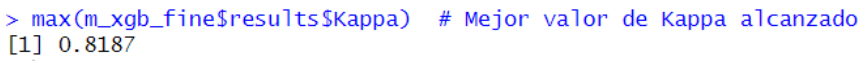


Figura. 23. Rendimiento de XGB con muestreo estratificado pt. 3.

Bien, ahora si los resultados son satisfactorios, pero ¿por qué no funcionó PCA ni ninguno de los intentos anteriores por mejorar el modelo?, creo que la clave está en el balance de las clases (y también la potencia de cómputo, pero eso es tema aparte). Desde el EDA e incluso el KDD notamos que habían fuertes desbalances en el dataset. En su momento argumenté que eso no debería ser un problema tan grande si usábamos árboles de decisión (de hecho, creo que eso se ve demostrado en que los modelos de árboles funcionaron muy bien a comparación de la regresión), sin embargo, lo que no consideré como posibilidad es que quizá para mejorar esas tres milésimas mi kappa pude haber optado por tratar el desbalance de clases. Es decir, a primera impresión no

parecía algo urgente, y por eso mismo es que lo pasé por alto. Recordemos que aunque los árboles no se ven tan afectados por estos detalles, si los desbalances y outliers son demasiado fuertes si podrían llegar a afectar (lo suficiente para que el kappa mejore o no esas milésimas). PCA, aunque útil para acelerar procesamiento, no hubiera corregido este desbalance ni garantizado una mejor capacidad de discriminación porque no atacaba este problema. A veces, mejorar un modelo no depende sólo de técnicas más sofisticadas, sino de entender cuál es el verdadero problema estructural que enfrenta el aprendizaje. Aunque también sería preciso seguir explorando otros enfoques para poder confirmar que el balanceo era un problema tan crítico o si hay otras formas más efectivas de mejorar el rendimiento.