

Momento de Retroalimentación: Módulo 2

Implementación de una técnica de aprendizaje máquina sin el uso de un framework. (Portafolio Implementación).

Fabián Erubiel Rojas Yáñez, A01706636

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro, México

A01706636@tec.mx

Resumen - En este proyecto se aborda el desarrollo de una técnica de aprendizaje máquina sin el uso de frameworks. En esta implementación abordaremos el uso de una regresión logística para la clasificación de una variable binaria, en este caso y debido a la naturaleza de mi dataset, se usa como predicción para una estrategia de marketing de un banco, en el que se busca predecir si un cliente abrirá un depósito con el banco después de algunas llamadas por parte de la institución.

I. INTRODUCCIÓN.

La regresión logística es una técnica de machine learning o aprendizaje máquina, utilizada para la predicción o clasificación de problemas lineales. Para lograr hacer esto usa funciones como la “sigmoide”, que se encarga de mapear valores y asignarlos a un rango probabilístico, lo que permite clasificar de una excelente forma valores entre 0 y 1, por eso en clasificación binaria es una excelente técnica.

En este trabajo, se presenta la implementación de una regresión logística desde 0, sin usar frameworks o librerías dedicadas a machine learning. Esto con el enfoque de desarrollar un mejor entendimiento de los modelos y posibles aplicaciones de machine learning. También aporta una comprensión sólida de técnicas de clasificación.

El dataset y la problemática a tratar es acerca de una campaña de marketing de un banco portugués, que busca predecir si un cliente se suscribirá o no a un depósito de plazo fijo, este acercamiento,

normalmente es por llamadas telefónicas y dentro del dataset se incluyen datos acerca de estas mismas, como la duración y los contactos realizados posteriormente (si son mayores a 1).

II. Dataset y su división.

Dentro del dataset escogido, se muestran las siguientes variables dentro del dataset:

Las variables categóricas y numéricas detectadas son las siguientes:

Las variables numéricas son aquellas que tienen un tipo de dato int64 y representan cantidades o medidas que se pueden ordenar y operar matemáticamente.

- age: Edad del cliente (int64)
- balance: Saldo promedio anual en euros (int64)
- day: Día del mes en que se realizó el último contacto (int64)
- duration: Duración del último contacto en segundos (int64)
- campaign: Número de contactos realizados durante esta campaña (int64)
- pdays: Número de días desde el último contacto en una campaña anterior (int64)
- previous: Número de contactos realizados antes de esta campaña (int64)
- Variables Categóricas

Las variables categóricas son aquellas que tienen un tipo de dato object y - representan categorías o etiquetas que no tienen un orden inherente.

- job: Tipo de trabajo (object)
- marital: Estado civil (object)
- education: Nivel educativo (object)
- default: Si tiene crédito en incumplimiento (sí/no) (object)
- housing: Si tiene préstamo hipotecario (sí/no) (object)
- loan: Si tiene préstamo personal (sí/no) (object)
- contact: Tipo de contacto (object)
- month: Mes del último contacto (object)
- poutcome: Resultado de la campaña anterior (object)
- y: Si el cliente se suscribió a un depósito a plazo fijo (sí/no) (object) - (variable objetivo)

Preprocesamiento del dataset:

Para hacer el procesamiento o “limpieza” del dataset usado, se usaron varias técnicas como el eliminar valores nulos o vacíos, valores duplicados valores “outliers” que son valores que al ser muy variables dentro del rango de valores obtenidos pueden afectar de forma negativa a el modelo, para hacer esto se usaron gráficas de caja, para establecer unos valores “medio” y detectar variables fuera de estos rangos medios.

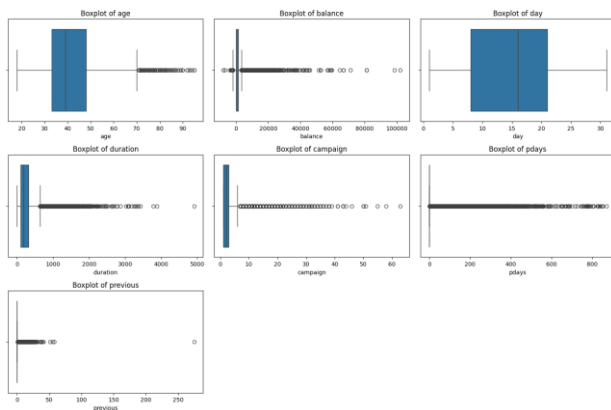


Fig 1. Gráficos de caja para las variables numéricas.

Para poder eliminar estos outliers, use la técnica del valor intercuartílico (IQR), que básicamente es una medida que ayuda a entender la dispersión de los datos entre dos distintos cuartiles Q1 Y Q3:

- Q1 = Es el primer cuartil, de el valor por debajo del 25%.
- Q3 = Es el tercer cuartil, de el valor que esté por

debajo del 75%.

Siguiendo la siguiente fórmula:

$$IQR = Q3 - Q1$$

El resultado de esto nos da un rango central donde se encuentren los datos de alrededor del 50%, los demás datos que están fuera de este rango podrían afectar negativamente modelos que posteriormente se implementaran, ya que son valores o muy grandes o muy pequeños que podrían causar un ruido que se puede evitar.

- Límite Inferior: Se define como $Q1 - 1.5 * IQR$. Cualquier valor por debajo de este límite se considera un outlier inferior.

- Límite Superior: Se define como $Q3 + 1.5 * IQR$. Cualquier valor por encima de este límite se considera un outlier superior.

Posteriormente procedí a eliminar columnas que considero son irrelevantes para el modelo:

- day
- month
- pdays
- previous
- poutcome
- contact

Considero que no tienen demasiada relevancia para la predicción que busco hacer.

Después procedi a disminuir las variables de trabajo de 12 a solamente 7,

```
job_map = {
    'management': 'Management',
    'admin.': 'Management',
    'entrepreneur': 'Management',
    'technician': 'Technical',
    'blue-collar': 'Technical',
    'services': 'Technical',
    'self-employed': 'Technical',
    'housemaid': 'Unskilled',
    'unemployed': 'Unemployed',
    'retired': 'Unemployed',
```

```

'student': 'Student',
'unknown': 'Other'
}

```

Esto con la finalidad de agrupar datos de la columna de trabajo y al momento de trabajar con ellas, sea más sencillo al tener menos variables disponibles. Esto debido a que se puede simplificar por el hecho de que no es tan relevante y podría hacerse más, puesto que sería más sencillo si se sabe que se trabaja o no, ya que esto afecta en la economía o aceptación del plazo con el banco.

Después, aplique la técnica de One Hot Encoding para las variables categóricas y convertirlas a variables binarias, esto con el fin de trabajarlas de forma numérica (binaria), ya que éstas considero que sí son relevantes en el análisis de la información y el desarrollo del modelo, por lo que trabajarlas de forma numérica me resulta más sencillo.

También use la técnica de escalamiento o normalización de datos, para los datos numéricos, ya que quería manejarlos de forma numérica y que fuera más sencillo para mí. Y que las grandes diferencias entre valores de las variables no afectarán negativamente al modelo.

III. Regresión Logística.

Elegí la regresión logística para abordar este problema, ya que se busca predecir si un cliente se suscribe o no a la institución bancaria, es decir es un problema binario que se reduce a “sí” o “no”. Para este tipo de problemas la regresión logística es muy eficiente y precisa, también es más sencilla de implementar que una red neuronal, al ser un problema lineal, si se puede ver de cierta forma una red neuronal es un conjunto de regresiones logísticas.

Use la función sigmoide de la regresión logística para mapear los valores entre 0 y 1, que básicamente es fundamental y el objetivo de la problemática, esto para arrojar valores lo más cercano posibles a 0 o 1.

También use la función de Cross Entropy para hacer los cálculos de la diferencia de los valores reales y los predichos, esta función guía al modelo

de aprendizaje para aprender de forma “Correcta” predecir o clasificar los datos.

De igual forma hice uso del algoritmo de Gradiente

descendiente para disminuir el error del modelo, en el entrenamiento del modelo de regresión, más en concreto desglosar este algoritmo, ya que es uno de los más importantes de mi modelo:

En concreto lo que busco de este algoritmo es ajustar los pesos y el bias para disminuir el error o el costo del modelo, esto para que las predicciones se acerquen más a las etiquetas deseadas y hacer preciso el modelo. Sirve para monitorear cómo va mejorando el algoritmo tras cada iteración o grupo de iteraciones.

Use una función de predicción para establecer las salidas de cada valor, es decir predice si es “1” o “0”. Esta se alimenta con las entradas, el bias y los pesos, en base a un cálculo realizado con ellos y usando la función sigmoide es que se establecen los valores si son “1” o “0”.

IV. Resultados obtenidos.

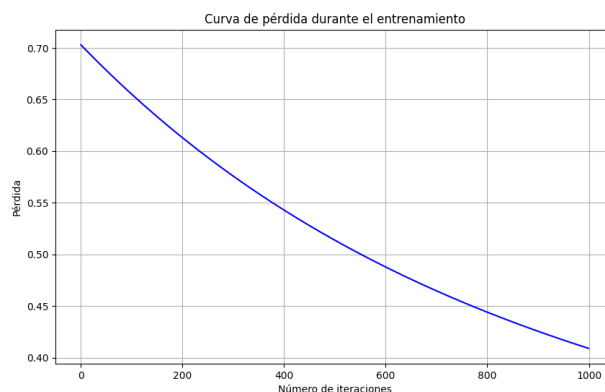


Fig 2. En esta gráfica se puede observar la pérdida durante el entrenamiento.

La interpretación de esta gráfica, quiere decir que el modelo va aprendiendo cada vez que hay más iteraciones, por lo que podemos deducir que va ajustando los parámetros de forma correcta para lograr aumentar su precisión de las predicciones. Por la forma de la curva de esta gráfica se puede deducir que no se observa un sobreajuste (overfitting) y al ser tan suave, también puedo deducir que el modelo es

estable y bueno, inicialmente la prueba se realizó con 1000 iteraciones, pero posteriormente se mostrara el resultado con más iteraciones.

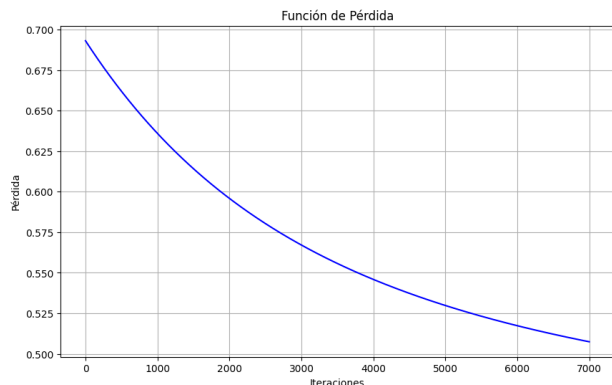


Fig 3. Pérdida después de 7000 iteraciones.

En esta primera prueba de este algoritmo revelaron áreas importantes en la mejora del algoritmo, a pesar de que el modelo logro identificar ciertos patrones en los datos, este tuvo una baja precisión al momento de predecir si los clientes se suscribirán o no al servicio, esto se puede ver por el gran numero entre falsos positivos y falsos negativos, lo que hace ver que aun existe un margen de mejora con la finalidad de aumentar la precisión del modelo. Un ejemplo de esto es porque posiblemente los hiperparametros de este modelo no sean los adecuados para los datos específicos con los que se trabaja. Además, es posible que se tenga que revisar los datos de entrada del modelo para reconsiderar su preprocesamiento y el cómo se manejan.

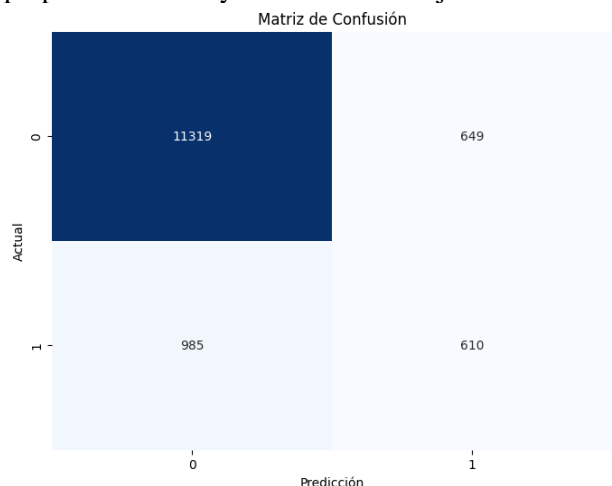


Fig 4. Grafica de la matriz de confusión.

Debido a estos resultados obtenidos con el dataset en estas primeras pruebas, con una separación de los datos de un 70% en datos de entrenamiento y

un 30% en datos de prueba (test), considere el realizar algunas mejoras en el modelo con la finalidad de incrementar su precisión como el sobre muestreo (oversampling) de los datos de entrada en la salida de “Positivos” o respuestas “si” a la suscripción de los clientes al servicio del banco, ya que al ser minoría, afecta de la detección de los patrones del modelo y esto es una clara característica que se refleja en la matriz de confusión mostrada arriba. AL detectar muchos falsos negativos y falsos positivos. Además de tener una detección muy baja de muy pocos verdaderos positivos. Esto junto a los siguientes valores de hiperparametros, que son 5000 iteraciones y 0.001 de learning rate.

Perdida/Costo 5800:	0.5196166115305304
Perdida/Costo 5900:	0.5184726448589444
Perdida/Costo 6000:	0.5173549409306336
Perdida/Costo 6100:	0.5162626672953344
Perdida/Costo 6200:	0.5151950237505871
Perdida/Costo 6300:	0.5141512409139357
Perdida/Costo 6400:	0.5131305788639765
Perdida/Costo 6500:	0.5121323258468187
Perdida/Costo 6600:	0.5111557970446758
Perdida/Costo 6700:	0.510200333403474
Perdida/Costo 6800:	0.5092653005165111
Perdida/Costo 6900:	0.5083500875613471

Fig 5. Función perdida costo con 7000 iteraciones.

IV. Mejoras del Algoritmo.

Como antes se había mencionado el uso de sobre muestreo para los casos positivos del dataset, ya que son minoritarios a los negativos lo que en pruebas anteriores sesgaba al algoritmo a tendencias de respuestas negativas. Esto ocurre ya que los algoritmos tienden a optimizar la precisión en general lo que conlleva a que favorezcan de cierta forma a los datos mayoritarios (en este caso negativos), lo que resulto en que este algoritmo ignoraba los casos positivos de salida.

Como mejora al algoritmo anterior decidí agregar nuevas métricas para poder analizar de mejor forma el desempeño del algoritmo, ya que solo con la función de perdida/costo y la matriz de confusión es difícil evaluar completamente el algoritmo: En este caso decidí incluir el F1-score, el recall y el AUC-ROC.

```

Perdida/Costo 0: 0.6930241499667426, Precisión: 0.5
Perdida/Costo 100: 0.4870321804685597, Precisión: 0.6939078486084281
Perdida/Costo 200: 0.4595428930616649, Precisión: 0.7315411032410388
Perdida/Costo 300: 0.44934427057836285, Precisión: 0.7484438720755527
Perdida/Costo 400: 0.44413208412287025, Precisión: 0.7588359447664019
Perdida/Costo 500: 0.4410388498862951, Precisión: 0.7627352078414538
Perdida/Costo 600: 0.43902747468076253, Precisión: 0.7656685984116763
Perdida/Costo 700: 0.43763172377211695, Precisión: 0.7681548257852185
Perdida/Costo 800: 0.4366138460823453, Precisión: 0.7693353366244544
Perdida/Costo 900: 0.4358419845541333, Precisión: 0.7706231666308936
Perdida/Costo 1000: 0.4352382521485959, Precisión: 0.7715353795521214
Perdida/Costo 1100: 0.4347542045673491, Precisión: 0.772358159834013
Perdida/Costo 1200: 0.43435834065482215, Precisión: 0.7731093940044359
Perdida/Costo 1300: 0.43402935876645327, Precisión: 0.7734850110896473

```

Fig 6. Función Perdida/costo y precisión del modelo inicial.

```

Perdida/Costo 3100: 0.43202600487707954, Precisión: 0.7752021177648994
Perdida/Costo 3200: 0.43199221655987036, Precisión: 0.775166344709165
Perdida/Costo 3300: 0.4319613114390841, Precisión: 0.775200042927667
Perdida/Costo 3400: 0.43193299593269974, Precisión: 0.7752915504042355
Perdida/Costo 3500: 0.43190701162166173, Precisión: 0.7752557773485012
Perdida/Costo 3600: 0.43188313033968967, Precisión: 0.7753273234599699
Perdida/Costo 3700: 0.43186115005507963, Precisión: 0.7753988695714388
Perdida/Costo 3800: 0.4318408913900599, Precisión: 0.775416756099306
Perdida/Costo 3900: 0.4318221947250709, Precisión: 0.7754525291550404
Perdida/Costo 4000: 0.4318049176064724, Precisión: 0.7754704156829076
Perdida/Costo 4100: 0.4317889326982348, Precisión: 0.7754525291550404
Perdida/Costo 4200: 0.43177412590235426, Precisión: 0.7753630965157043
Perdida/Costo 4300: 0.4317603947879443, Precisión: 0.775416756099306
Perdida/Costo 4400: 0.4317476472258624, Precisión: 0.7754704156829076
Perdida/Costo 4500: 0.43173580020484964, Precisión: 0.7754346426271732
Perdida/Costo 4600: 0.43172477880188415, Precisión: 0.7753988695714388
Perdida/Costo 4700: 0.4317145152840655, Precisión: 0.7753988695714388
Perdida/Costo 4800: 0.43170494832310113, Precisión: 0.775416756099306
Perdida/Costo 4900: 0.43169602230652354, Precisión: 0.7754346426271732

```

Fig 7. Función Perdida/costo y precisión del modelo final.

En la Fig.6 y 7 se puede observar la mejora sustancial del algoritmo conforme al ajuste de los hiperparametros, ya que en este caso el learning rate se modifico a 0.1 lo que resulto en una convergencia más rápida e incluí la precisión conforme avanzaba el numero de iteraciones y es visible como mejora de inicio a fin. Esto incluso cuando baje el numero de iteraciones de 7000 a solo 5000.

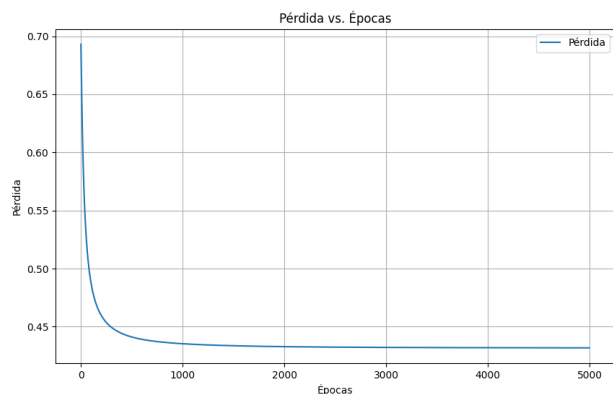


Fig 8. Gráfica de la pérdida vs épocas.

En la figura 8 se puede observar la pérdida del modelo conforme avanzan las iteraciones, al inicio se puede observar como inicia en un valor muy alto

(alrededor de .70) y como con solo las primeras iteraciones disminuye drásticamente lo que puedo interpretar como que el modelo aprende rápidamente desde el inicio, pero conforme avanza se ve como el modelo se acerca a la convergencia lo que significa que el modelo deja de hacer cambios grandes o drásticos a los pesos y a la pérdida deja de ser significativa.

La convergencia de la curva del modelo se da alrededor de las 2000 iteraciones/épocas y después se ve un aplanamiento de la línea, lo que quiere decir que a partir de las 2000 iteraciones el modelo, esto sugiere que el modelo no mejorara sustancialmente con más iteraciones después de las 2000, ya que no hay cambios significativos después de estas iteraciones.

Otra métrica añadida y en la que se puede corroborar la grafica anterior es en comparando la precisión del modelo con las épocas.

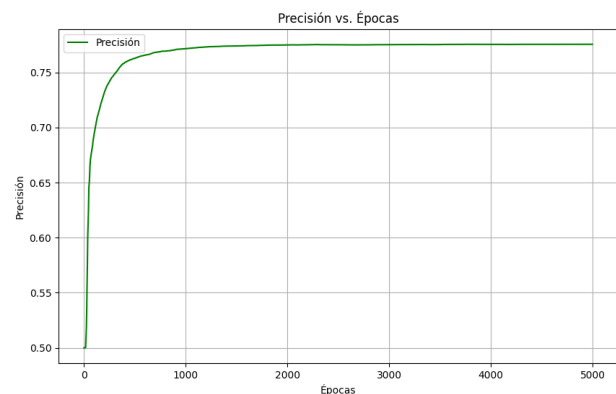


Fig 9. Gráfica precisión vs épocas.

En esta grafica se puede ver el desempeño del modelo y como alcanza una estabilidad alrededor de las 2000 iteraciones, se observa que es alrededor del 80%.

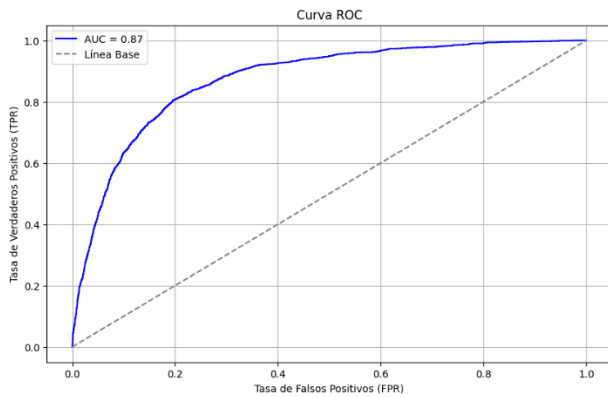


Fig 11. Curva ROC.

En la figura 11, es posible observar la curva ROC del modelo, esta se puede interpretar diciendo que el modelo tiene una precisión del 87% para distinguir entre positivo o negativo, entre si el cliente se suscribirá o no al servicio del banco.

Reporte de Métricas:				
	precision	recall	f1-score	support
0	0.95	0.90	0.92	11968
1	0.46	0.63	0.53	1595
accuracy			0.87	13563
macro avg	0.70	0.77	0.73	13563
weighted avg	0.89	0.87	0.88	13563

Fig 11. Reporte F1-score.

Este reporte analiza el desempeño y rendimiento del modelo en base a las métricas que se genera, su objetivo es evaluar la capacidad de predecir las clases a pesar del contexto de desbalance de datos que intente reducir con la técnica de sobre muestreo (oversampling).

- Para la clase negativa se puede observar tiene una precisión del 95% lo que quiere decir que clasifica correctamente el 95 de las muestras etiquetadas como negativas, sin embargo, el recall es del 90% que son el las muestras que realmente perteneces a la clase negativa. El F1 score es del 92% lo que muestra un balance positivo dentro de la detección de negativos.
- Para la clase positiva solo presenta una precisión del 46% con un recall del 63% y un F1 score de 53% lo que se puede interpretar como que no predice de forma tan correcta

los valores positivos.

- En las métricas globales el modelo tiene una precisión del 87%, pero esta es una métrica engañosa ya que como antes se pudo observar con la precisión por cada clase, esta dominada por la clase mayoritaria que es negativa.

Es posible ver el impacto negativo que tiene la clase negativa al tener mas datos o ser mayoritaria con 11968 que la clase positiva con tan solo 1595.

A pesar de aplicar la técnica de sobre muestreo (SMOTE), aun para el modelo es impreciso para la clase positiva o mejor dicho al momento de predecir si el cliente se suscribirá o no al servicio del banco.

III. Regresión Logística.

El modelo tiene una buena precisión para diferenciar entre las clases negativas y positivas, esto se puede observar en la grafica ROC, lo que se puede interpretar como que el modelo tiene una mayor probabilidad de asignar una puntuación alta a una muestra positiva que a muestras negativas, tiene un rendimiento bueno según la curva que se observa en la gráfica ROC.

Sin embargo, al momento de realizar el análisis más a profundidad usando métricas para medirlo como el F1-score y la matriz de confusión, se muestra u desbalance en las muestras positivas y negativas, ya que originalmente la clase mayoritaria es la respuesta negativa por parte de los clientes y a pesar de aplicar técnicas de sobre muestreo en el dataset, el modelo presenta una mejora muy baja, lo que sugiere que es necesario aplicar técnicas de procesamiento de datos distintas a la utilizada o utilizar otro tipo de algoritmos, ya que finalmente se puede ver como el algoritmo queda segado a la respuesta negativa que es dominante sobre la positiva

Referencias:

Dataset: *UCI Machine Learning Repository*. (2014). Uci.edu.
<https://archive.ics.uci.edu/dataset/222/bank+marketing>

Momento de Retroalimentación: Módulo 2

Implementación de una técnica de aprendizaje máquina con el uso de un framework. (Portafolio Implementación).

Fabián Erubiel Rojas Yáñez, A01706636

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro, México

A01706636@tec.mx

Resumen – Este proyecto es la continuación del anterior ya que el proyecto anterior ya que en el proyecto pasado se pretendía usar el dataset de un banco para poder determinar si los clientes se suscribirían o no al servicio prestado, en este proyecto busco aumentar la precisión del modelo al momento de predecir los valores positivos, ya que en el anterior no logre aumentar esta precisión del modelo. Cabe destacar que en este proyecto buscare implementar un algoritmo diferente que es el Random Forest como clasificador y también que en este hare uso de bibliotecas y frameworks para su implementación.

no fue la esperada debido a que el algoritmo quedo sesgado a la respuesta negativa, en busca de un mejor resultado, llevare a cabo la implementación de un nuevo algoritmo, que es “Random Forest” como antes mencionaba, esperando un resultado mas preciso para un modelo que pueda hacer mejores predicciones.

Para la implementación de este algo se usarán frameworks y librerías, usando el lenguaje “Python” y el entorno de Google Colab para lograr el cometido.

I. Introducción.

El problema para abordar es sobre una institución financiera que busca predecir si los clientes se suscribirán o no en base a una campaña lanzada donde por medio de llamadas a sus clientes, donde por medio de uno o varios contactos telefónicos recopilaron los datos de estas llamadas hacia los clientes.

Para llegar a este objetivo se llevara a cabo la implementación de un “Random Forest Classifier” que es un algoritmo de machine learning para clasificar, basado en combinar múltiples arboles de decisión para hacer mas preciso y robusto el modelo, debido a que en la implementación anterior del modelo de regresión logística la precisión obtenida

II. Algoritmo Random Forest

En esta proyecto se llevara a cabo la implementación de el algoritmo Random Forest con la finalidad de desarrollar un algoritmo clasificador, que es uno de los principales usos que se le dan a este algoritmo. En este caso funciona creando un numero determinado de arboles de decisión que son alimentados con el dataset original mezclado, es decir que reciben datos aleatorios para poder generar resultados diferentes entre ellos, generando subconjuntos para entrenar de forma diferente cada árbol de decisión en base al dataset, eso hace que cada árbol generado tenga un resultado, en este caso al ser un clasificador el resultado que entrega cada árbol es sometido de cierta forma a una votación en donde gana la mayoría de resultados (0 o 1).

Esta aplicación es la idónea para el objetivo

que tiene el dataset usado para esta implementación.

Donde solo para recordar un poco estos son los atributos con los que se alimentara el modelo:

1. .housing
 - a. Descripción: Indica si el cliente tiene un préstamo hipotecario.
 - b. Tipo de dato: Binario ("yes", "no").
 - c. Relevancia: Permite identificar si la deuda hipotecaria puede influir en la decisión del cliente de suscribir un depósito a plazo.
2. Loan
 - a. Descripción: Indica si el cliente tiene un préstamo personal.
 - b. Tipo de dato: Binario ("yes", "no").
 - c. Relevancia: Un préstamo personal activo podría afectar la disposición del cliente a invertir en un depósito a plazo.
3. Y.
 - a. Descripción: Variable objetivo que indica si el cliente ha suscrito un depósito a plazo.
 - b. Tipo de dato: Binario ("yes", "no").
 - c. Relevancia: Es la variable de interés principal para el análisis y el modelo predictivo.
4. job_Management
 - a. Descripción: Variable derivada que indica si el cliente trabaja en el sector de "management".
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece al sector, 0 en caso contrario).
 - c. Relevancia: Captura la posible relación entre la ocupación y la decisión de suscribir un depósito.
5. job_Other
 - a. Descripción: Variable derivada que agrupa a clientes con ocupaciones distintas de las clasificadas explícitamente.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Proporciona un marco para analizar ocupaciones menos frecuentes.
6. job_Student
 - a. Descripción: Variable derivada que indica si el cliente es estudiante.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Refleja la relación entre la etapa académica del cliente y su interés en productos financieros.
7. job_Technical
 - a. Descripción: Variable derivada que indica si el cliente trabaja en el sector técnico.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Explora cómo las ocupaciones técnicas influyen en el comportamiento financiero.
8. job_Unemployed
 - a. Descripción: Variable derivada que indica si el cliente está desempleado.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Permite analizar cómo el desempleo afecta la capacidad de ahorro e inversión.
9. job_Unskilled
 - a. Descripción: Variable derivada que indica si el cliente tiene una ocupación no especializada.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Examina cómo la calificación laboral influye en las decisiones financieras.
10. poutcome_failure
 - a. Descripción: Variable derivada que indica si el resultado de la campaña previa fue un fracaso.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Proporciona información histórica sobre las interacciones previas con el cliente.
11. poutcome_other
 - a. Descripción: Variable derivada que agrupa resultados no clasificados de la campaña previa.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Captura información

- residual de campañas previas que no encajan en otras categorías.
12. `poutcome_success`
 - a. Descripción: Variable derivada que indica si el resultado de la campaña previa fue exitoso.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Una campaña previa exitosa puede aumentar la probabilidad de un resultado positivo en la actual.
 13. `poutcome_unknown`
 - a. Descripción: Variable derivada que indica si el resultado de la campaña previa es desconocido.
 - b. Tipo de dato: Categórico (dummy: 1 si pertenece a esta categoría, 0 en caso contrario).
 - c. Relevancia: Representa la falta de información histórica sobre la relación del cliente con la campaña.
 14. `contact_prev`
 - a. Descripción: Número de contactos previos realizados durante campañas anteriores.
 - b. Tipo de dato: Numérico.
 - c. Relevancia: Permite analizar la persistencia del banco y su posible efecto en la decisión del cliente.
 15. `balance_winsorized`
 - a. Descripción: Balance promedio anual del cliente, ajustado mediante winsorización para limitar los valores atípicos.
 - b. Tipo de dato: Numérico.
 - c. Relevancia: El balance económico del cliente puede ser un predictor clave de su disposición a invertir.
 16. `duration_winsorized`
 17. Descripción: Duración del último contacto en segundos, ajustada mediante winsorización para limitar valores extremos.
 18. Tipo de dato: Numérico.
 19. Relevancia: Es una de las variables más importantes en campañas de marketing, ya que las llamadas más largas suelen estar asociadas con una mayor probabilidad de conversión.
 20. `previous_winsorized`
 - a. Descripción: Número de contactos

previos realizados con el cliente, ajustado mediante winsorización.

- b. Tipo de dato: Numérico.
- c. Relevancia: Mide la persistencia del banco y su posible relación con el resultado actual.

No entrare en mas desarrollo del dataset, ya que el proceso de preprocesamiento se explica en el primer documento, este es solo para un recordatorio de las variables a usar para el modelo.

III. Desarrollo del Modelo:

Debido a que en el algoritmo pasado se quedo sesgado a la clase mayoritaria que era negativa, decidí implementar este nuevo y probar con el mismo dataset, con la finalidad de mejorar la precisión de detección del modelo.

```
# Definir y entrenar el modelo Random Forest con hiperparámetros predefinidos
rf_classifier = RandomForestClassifier(
    n_estimators=200,      # Número de árboles en el bosque
    max_depth=15,         # Profundidad máxima de los árboles
    min_samples_split=10, # Mínimo de muestras requeridas para dividir un nodo
    min_samples_leaf=2,   # Mínimo de muestras por hoja
    class_weight='balanced', # Penalización para clases desbalanceadas
    random_state=0
)
```

Fig 1. Hiperparametros del algoritmo.

En la Fig 1 se pueden observar los hiperparametros definidos para el algoritmo los cuales fueron los que mejor desempeño tuvieron a lo largo de las pruebas hechas en el.

- `N_estimators` : define el numero de arboles de decisión que conforman el modelo, inicialmente se hizo una prueba con 100 que es un número por defecto, sin embargo, al subir la cantidad aumento la precisión del modelo, ya que a pesar de que es muy baja, lo era aún más. Esto por que influye en reducir la varianza de las predicciones. Obviamente también aumento el tiempo que tardo en correr el algoritmo.
- `Max_depth` : Este hiperparametro configura la profundidad máxima de los árboles, esto con la finalidad de limitar el sobreajuste del modelo, el valor de 15 fue el punto de balance para garantizar el ajuste adecuado

de los datos.

- **Min_samples_split** : Aquí se especifica el número mínimo de muestras que un nodo debe de contener para poder dividirse, esto para reducir el riesgo de crear nodos pequeños que podría causar un sobre ajuste del modelo.
- **Min_samples_leaf** : Aquí se establecen el mínimo de muestras que debe de tener una hoja, es decir el ultimo nodo del árbol o el más profundo. Este hiperparametro se ajusto en 2 para evitar que el modelo generara arboles con muy pocos datos, lo cual lo haría muy específico y poco preciso.
- **Class_weight=balanced** : Este parametro propio de la librería sirve para balancear el peso asignado a cada clase de acuerdo a la frecuencia que tiene, para esta problemática en concreto donde la clase negativa era mayoritaria era algo fundamental, ya que mitiga el desbalanceo de clases cuando alguna es mayoritaria.
- **Random_state** : Este se encarga de controlar la aleatoriedad del modelo para replicar los resultados.

IV. Resultados obtenidos.

```
Precisión en los datos de prueba: 0.8054408728988499
Matriz de confusión:
[[9647 2330]
 [ 309 1278]]
Reporte de clasificación:
```

	precision	recall	f1-score	support
0	0.97	0.81	0.88	11977
1	0.35	0.81	0.49	1587
accuracy			0.81	13564
macro avg	0.66	0.81	0.69	13564
weighted avg	0.90	0.81	0.83	13564

Fig. 2 F1-score del algoritmo.

En el resultado del F1-score del algoritmo que se puede observar en la Fig 2. Se puede observar como el algoritmo alcanzo una precisión global del 80%, un dato bastante engañoso, ya que esta métrica contempla todas las muestras, tanto de la clase mayoritaria como de la minoritaria, por lo que al prestar atención e interpretando de la matriz de confusión que muestra realmente los valores predichos frente al conjunto real de datos, en esta se muestran como gran parte de los valores negativos fueron correctamente clasificados, mientras que para la

clase positiva tuvo bastantes problemas al clasificar correctamente ya que son pocas las muestras positivas correctas clasificadas correctamente y existe un gran número de falsos positivos con una clasificación incorrecta.

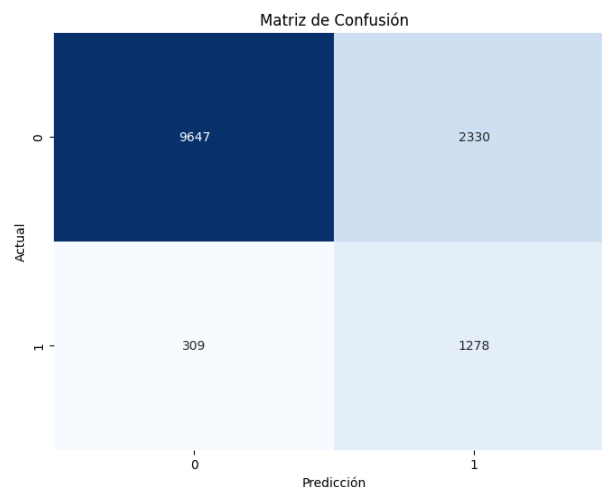


Fig 3 Matriz de confusión del modelo.

Como se puede observar en la Fig. 3 la matriz de confusión del modelo tiene una gran precisión al clasificar la clase mayoritaria negativa, pero problemas al clasificar correctamente la positiva, lo que hace que la precisión global sea una métrica engañosa, ya que esta disminuye dependiendo de las clases.

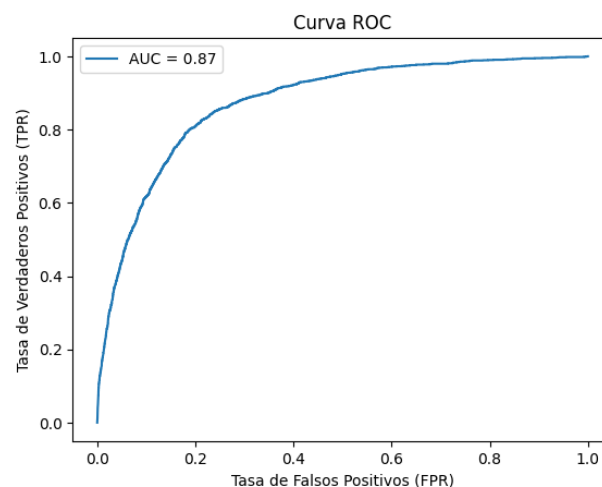


Fig. 4 Curva ROC del modelo

En la Fig 4. Es posible ver la curva ROC del modelo que se usa para evaluar la capacidad del modelo para identificar las clases positivas y negativas en problemas de clasificación como lo es este. En el eje X se observa la tasa de Falsos Positivos y en el eje Y la tasa de verdaderos

positivos.

Al interpretar esta curva obtenida y con el valor de .87 obtenido es posible interpretar que el modelo tiende a identificar de manera mas recurrente un “Si” resultado positivo sobre el negativo, lo que en las pruebas manuales del programa, ingresando datos manualmente como usuario comprobé que es verdad ya que de 10 ingresos 8 resultaron con un resultado afirmativo.

Claro que en esta curva no se refleja el desbalanceo de clase que existe ya que no tiene una alta precisión al detectar verdaderos positivos.

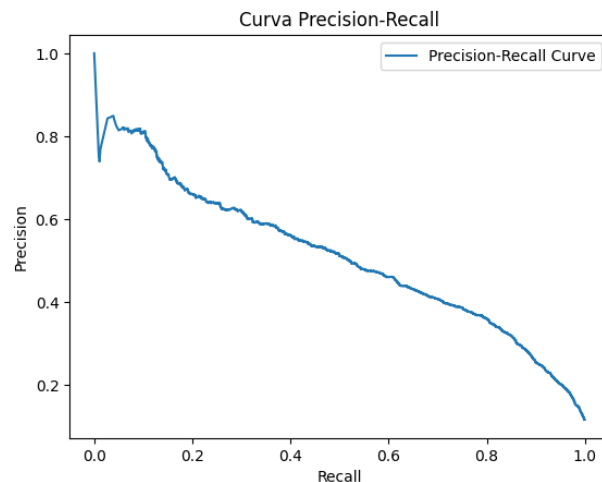


Fig 5. Grafica Precisión vs Recall.

Esta métrica es útil para ver la precisión de un modelo de clasificación, ya que muestra la relación entre la Precisión del modelo y el Recall que es como la tasa de verdaderos positivos. Donde la precisión se puede entender con la proporción de instancias correctamente clasificadas como positivas entre todas las clasificadas como positivas y el Recall como la proporción de instancias positivas correctamente clasificadas entre todas las instancias positivas reales.

En la grafica en el eje X se encuentra el Recall mientras que en el eje Y se muestra la precisión.

De la grafica puedo interpretar que al ser descendente mientras que el Recall (eje X) aumenta, el modelo tiende a detectar mas

positivos lo que hace que la precisión disminuya. Esto hace que el modelo detecte mas falsos positivos.

Es decir que al aumentar el recall aumentan la detección de falsos positivos, el modelo tiende a identificar con mayor frecuencia falsos positivos, ya que clasifica muchas instancias negativas como positivas.

Al inicio de la grafica es posible observar un pico irregular lo que posiblemente podemos interpretar como un indicativo de un desbalance de clases que es cierto al realizar el conteo del dataset, ya que como antes he mencionado la clase mayoritaria es la negativa.

Para comprobar esto realice algunas pruebas ingresando datos manualmente, de forma en que pudiera quedar de forma tabular, debido a que el dataset con el que trabaje y fue entrenado este modelo es de esa forma.

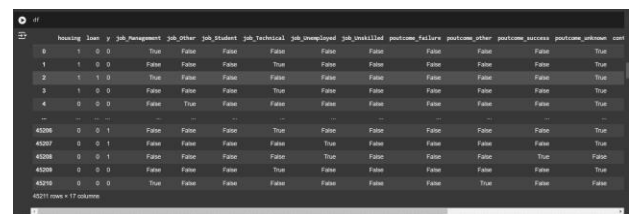


Fig 6. Dataframe utilizado.

Para probar esto, desarrolle una función para interactuar con el usuario donde se podía ingresar de forma manual los datos de las columnas del dataframe para haer pruebas del modelo, los resultados es que con 10 pruebas hechas en el modelo 8 fueron positivas a pesar de las respuestas que se le daban al modelo y solo dos negativas algo que se comprueba con la tabla de Precision vs Recall mostrada en la Fig 5. Ya que el modelo tiene tendencia a los falsos negativos, clasifica erróneamente algunos negativos.

```
?Tienes hipoteca? (1: Si, 0: No): 1
?Tienes préstamo personal? (1: Si, 0: No): 1

Opciones de empleo:
1: Management
2: Other
3: Student
4: Technical
5: Unemployed
6: Unskilled
Selecciona tu empleo (ingresa solo el número): 4
Ingresa tu balance anual (sueldo en euros): 60000
?Has tenido contactos previos con el banco? (1: Si, 0: No): 1
?Cuántos contactos previos has tenido con el banco? 5
Duración total de los contactos previos (en segundos): 600
?Cuántos de esos contactos fueron exitosos? 1

Selecciona el resultado previo de la campaña:
1: failure
2: other
3: success
4: unknown
Elije una opción (número): 1
El modelo predice que el cliente SI SE SUSCRIBIRÁ un depósito a plazo con el banco.
```

Fig 7. Prueba de datos ingresados por el usuario.

Conclusión:

En general este algoritmo demostró cierta robustez al clasificar de forma correcta la clase negativa, a pesar de la limitación presentada en la clase minoritaria y su precisión moderada conforme a esta clase.

En este algoritmo a pesar de aplicarle técnicas de sobrepoblación a la clase minoritaria y aplicar parámetros para tratar de balancear esta hegemonía de la clase mayoritaria negativa, no fue posible el aumentar la capacidad y precisión del modelo para identificar clases positivas.

Todo esto me lleva a considerar que la forma de aumentar la precisión de el modelo no sea aplicando o modificando hiperparametros, ya que se hicieron bastante pruebas con distintas configuraciones de ellos y se presento la que mejores resultados dio, sino que la forma de aborar y mejorar este algoritmo es con un preprocesamiento distinto del dataset, ya que es posible trabajarlo de otras formar y posiblemente encontrar una mayor correlación entre ciertas variables dentro de el o aplicar técnicas distintas para el.