

Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 23/11/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

## Reporte Práctica 10: Clasificación con Regresión Logística

**Objetivo:** Obtener un modelo de clasificación para evaluar casos de crédito hipotecario para la adquisición de una casa con tasa fija a 30 años.

**Fuente de datos:** Se utilizarán datos que ya habíamos utilizado para otras prácticas. La fuente de datos es el archivo *hipoteca.csv*, el cual cuenta con 202 registros con los siguientes datos:

- ingresos: son ingresos mensuales de 1 o 2 personas, si están casados.
- gastos\_comunes: son gastos mensuales de 1 o 2 personas, si están casados.
- pago\_coche
- gastos\_otros
- ahorros
- vivienda: valor de la vivienda.
- estado\_civil: 0-soltero, 1-casado, 2-divorciado
- hijos: cantidad de hijos menores (no trabajan).
- trabajo: 0-sin trabajo, 1-autonomo, 2-asalariado, 3-empresario, 4-autonomos, 5-asalariados, 6-autonomo y asalariado, 7-empresario y autonomo, 8-empresarios o empresario y autónomo
- comprar: 0-alquilar, 1-comprar casa a través de crédito hipotecario con tasa fija a 30 años.

### Características Generales:

En esta práctica, utilizaremos la clasificación por regresión logística para predecir etiquetas que nos permitan conocer si un tumor es benigno o maligno. Para ello, construiremos un modelo a través de un conjunto de entrenamiento y utilizaremos un conjunto de prueba para evaluar dicho modelo.

La regresión logística es un tipo de aprendizaje supervisado cuyo objetivo es predecir valores binarios (0 o 1). Para este tipo de aprendizaje, se utiliza la función logística o sigmoide, la cual se muestra a continuación:

$$\text{Función logística} = \frac{1}{1 + e^{-(a+bX)}}$$

## Desarrollo

Primeramente, tenemos que definir aquellas bibliotecas de Python que nos serán útiles para importar, limpiar y analizar los datos contenidos en el archivo separado por comas *RGeofisicos.csv*. Estas serán: *pandas* (manipulación y análisis de datos), *matplotlib* (para la creación de gráficas y visualización de los datos), *numpy* para utilizar vectores y matrices de  $n$  dimensiones.

Más tarde, importamos el archivo *Hipotecas.csv* y lo primero que hacemos es guardarlo en un DataFrame. La importación del archivo se realizó a partir del explorador de archivos que abrimos utilizando el comando `files.upload()`. Una vez importados los datos, mostramos el DataFrame que los contiene:

```
Hipoteca = pd.read_csv('Hipoteca.csv')
Hipoteca
```

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
0	6000	1000	0	600	50000	400000	0	2	2	1
1	6745	944	123	429	43240	636897	1	3	6	0
2	6455	1033	98	795	57463	321779	2	1	8	1
3	7098	1278	15	254	54506	660933	0	0	3	0
4	6167	863	223	520	41512	348932	0	0	3	1
...	...	...	...	...	...	...	...	...	...	...
197	3831	690	352	488	10723	363120	0	0	2	0
198	3961	1030	270	475	21880	280421	2	3	8	0
199	3184	955	276	684	35565	388025	1	3	8	0
200	3334	867	369	652	19985	376892	1	2	5	0
201	3988	1157	105	382	11980	257580	0	0	4	0

202 rows × 10 columns

Figura 1: Data Frame con algunos de los datos.

Una vez hecho lo anterior, utilizamos una matriz de correlaciones con el propósito de seleccionar variables significativas en el conjunto de datos y reducir la dimensionalidad del mismo. Como se muestra en la figura 2, utilizamos el valor de *Comprar* como *hue* para el *pairplot* de correlación entre las variables. Además, en la figura 3 se muestra la matriz de correlación de Pearson, utilizando colores para distinguir las variables que tienen una mayor dependencia y así poder realizar la selección de variables.

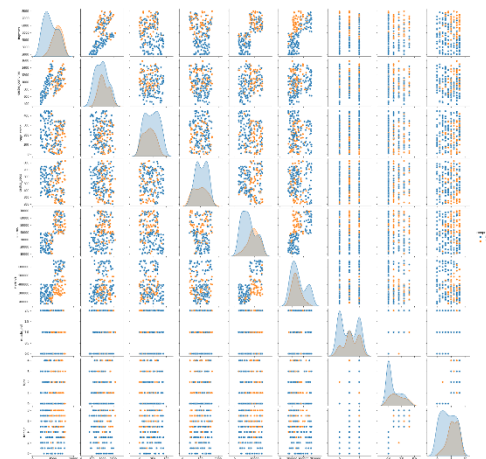


Figura 2: Pairplot con los datos de hipotecas utilizando *Comprar* como *Hue*.

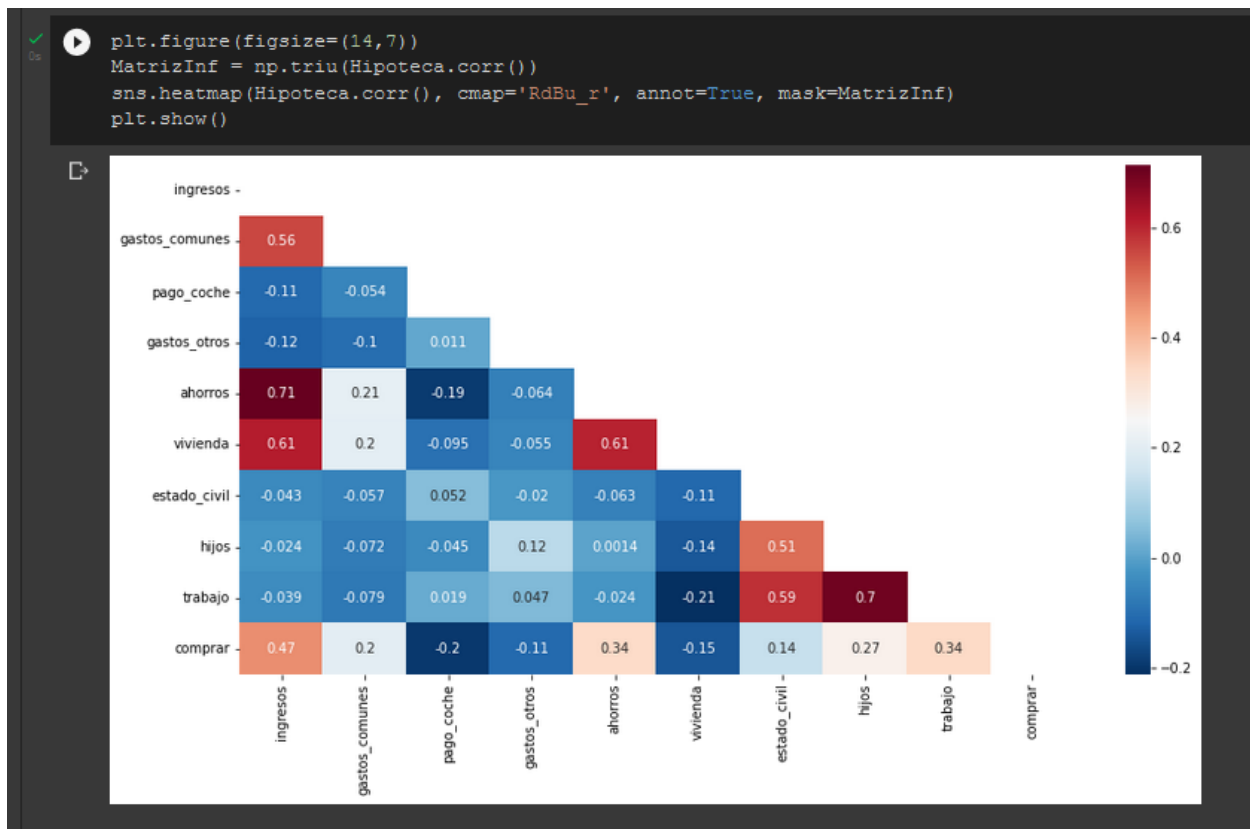


Figura 3: Utilizando el método de correlaciones de Pearson y un mapa de calor para visualizar la dependencia entre las variables de *Hipoteca*.

Una vez hecho lo anterior, se seleccionan las variables *ingresos*, *gastos\_comunes*, *pago\_coche*, *gastos\_otros*, *ahorros*, *vivienda*, *estado\_civil*, *hijos* y *trabajo* como variables predictoras. Por más que *ingresos* y *ahorros* tienen una correlación alta (0.71), se tomarán en cuenta igual para hacer la clasificación hipotecaria. Luego, se define a la variable *comprar* como variable clase, sobre la cual se hará la clasificación binaria.

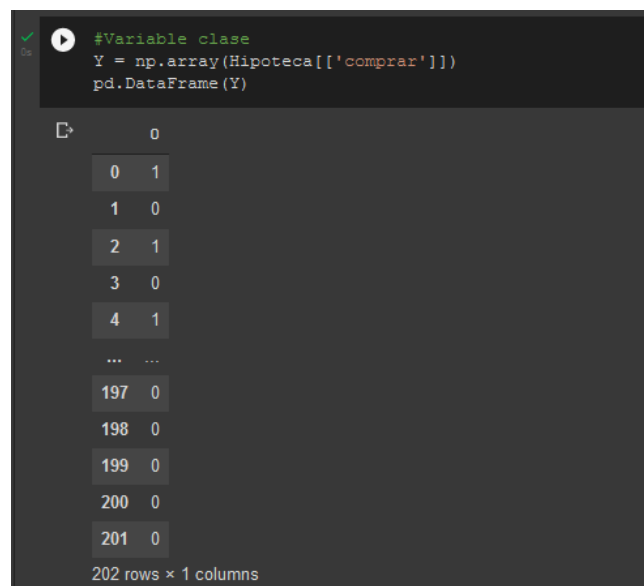


Figura 4: Parte del DataFrame con la variable comprar como valores binarios

Definimos entonces un *array* con las variables predictoras que nos permitirán predecir el valor de *compra* en la clasificación, como se muestra en la figura 5.

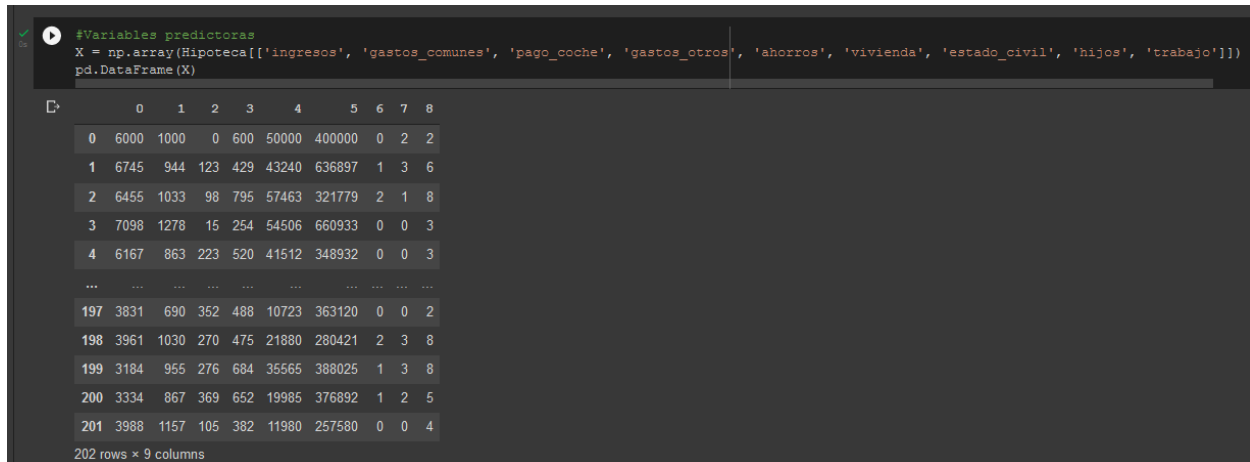


Figura 5: Parte de la matriz con las variables predictoras.

Además, realizamos un *scatter plot* con los valores de *ingresos* y *vivienda* para observar su distribución.

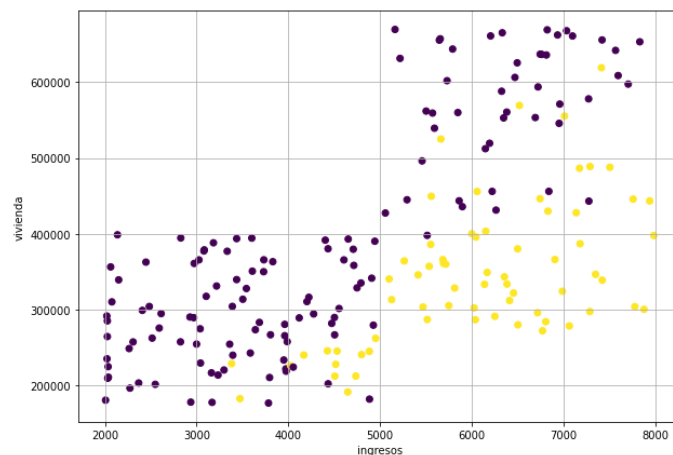


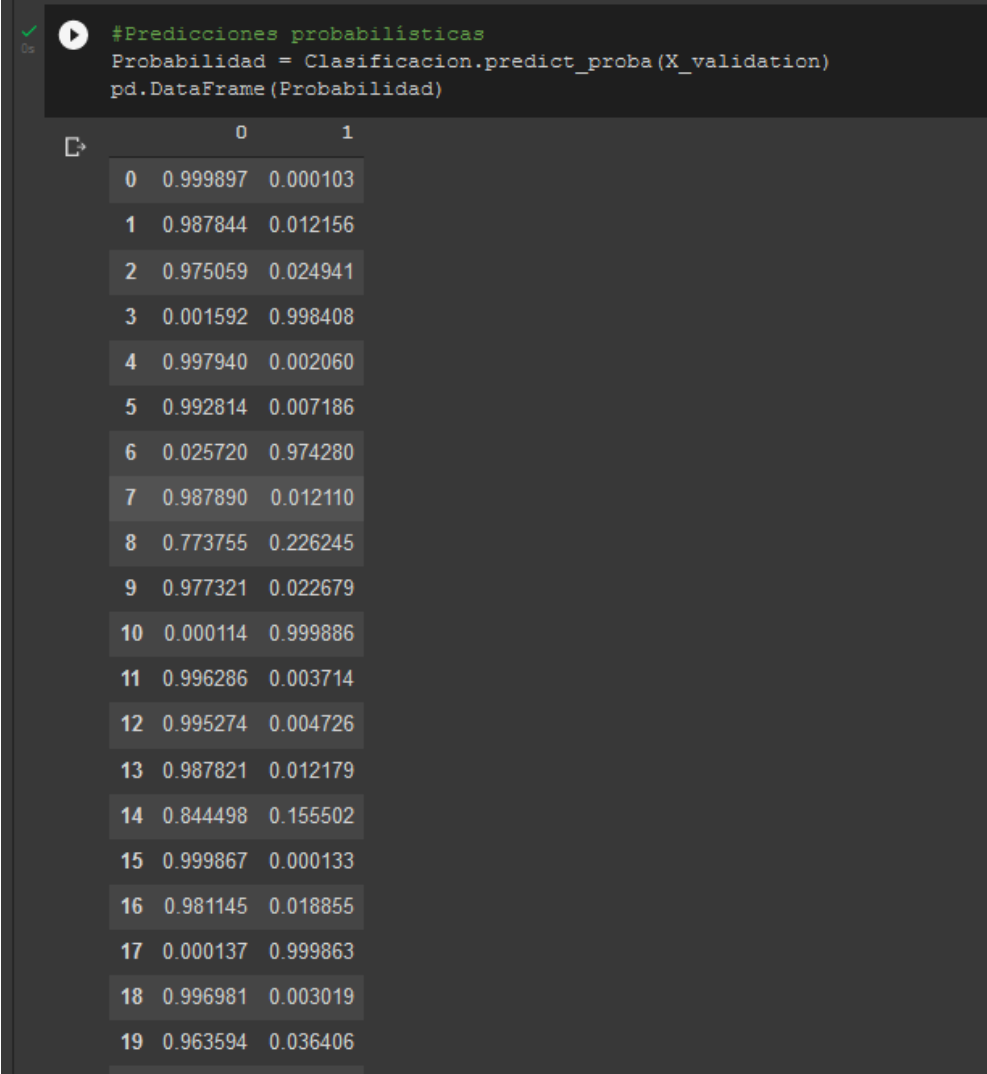
Figura 6: *Scatter plot* con los ingresos vs la vivienda.

Para la aplicación del algoritmo de regresión logística, primero tenemos que hacer la importación de las funciones de *linear\_model* y *model\_selection* de la biblioteca *sklearn* y *classification\_report*, *confusión\_matrix* y *accuracy\_score* de *sklearn.metrics*, respectivamente.

Una vez hecho lo anterior, podemos utilizar la función *model\_selection*, pasando como parámetros las variables predictoras y la variable clase, así como el tamaño del test (en un primer caso se utilizará un grupo de prueba del 20% de la cantidad total de datos), el *random state* (para fijar la posición de los pseudoaleatorios y reproducir los mismos resultados en distintas ejecuciones) y *shuffle* (con valor *True* para que los datos de entrenamiento y prueba sean elegidos de manera aleatoria en el conjunto de datos).

Utilizamos la función *LogisticRegression* y *fit* para crear un objeto para hacer el entrenamiento con los datos de entrada definidos y crear el modelo de clasificación. Luego, generamos las

predicciones probabilísticas de los datos de prueba, mostrando en la primera columna del DataFrame de la figura 7 la probabilidad del valor de compra (si es mayor al 50%, el valor será de 1 y en caso contrario será de 0) y en la segunda columna mostraremos la exactitud de dicha probabilidad.



```
#Predicciones probabilísticas
Probabilidad = Clasificacion.predict_proba(X_validation)
pd.DataFrame(Probabilidad)
```

	0	1
0	0.999897	0.000103
1	0.987844	0.012156
2	0.975059	0.024941
3	0.001592	0.998408
4	0.997940	0.002060
5	0.992814	0.007186
6	0.025720	0.974280
7	0.987890	0.012110
8	0.773755	0.226245
9	0.977321	0.022679
10	0.000114	0.999886
11	0.996286	0.003714
12	0.995274	0.004726
13	0.987821	0.012179
14	0.844498	0.155502
15	0.999867	0.000133
16	0.981145	0.018855
17	0.000137	0.999863
18	0.996981	0.003019
19	0.963594	0.036406

Figura 7: Probabilidades de los datos de prueba. La primera columna representa el valor de la probabilidad (si es mayor a .5, el valor será de 1 y será de 0 en caso contrario) y en la segunda columna se muestra la exactitud de dicha probabilidad.

```
#Predicciones con clasificación final
Predicciones = Clasificacion.predict(X_validation)
pd.DataFrame(Predicciones)
```

	0
0	0
1	0
2	0
3	1
4	0
5	0
6	1
7	0
8	0
9	0
10	1
11	0
12	0

Figura 8: Predicciones con clasificación final (etiqueta de 0 o de 1).

Luego, utilizamos la función *score* para calcular la exactitud promedio de la validación, obteniendo que, para un grupo de prueba del 20%, la exactitud promedio de la validación fue de 0.926829. Más tarde, utilizamos la matriz de clasificación vs el dato real para conocer qué datos de la clasificación realizada difieren de los datos reales (arrojando falsos positivos o falsos negativos).

Clasificación		
	0	1
Reales		
0	28	0
1	3	10

Figura 9: Matriz de clasificación para conocer los falsos positivos y falsos negativos.

Utilizamos la función `classification_report` para obtener un reporte de la clasificación recién realizada. En este reporte, obtenemos el valor de exactitud (grado de fiabilidad del modelo), así como otros datos como la precisión (qué tanto se va a reproducir la clasificación) o el *recall* (que en este caso nos indica que, siendo reales, se equivocó mayormente en los negativos).

```
[21] #Reporte de la clasificación
print("Exactitud", Clasificacion.score(X_validation, Y_validation))
print(classification_report(Y_validation, Y_Clasificacion))
```

	precision	recall	f1-score	support
0	0.90	1.00	0.95	28
1	1.00	0.77	0.87	13
accuracy			0.93	41
macro avg	0.95	0.88	0.91	41
weighted avg	0.93	0.93	0.92	41

Figura 10: Reporte de la clasificación.

Posteriormente, se obtiene la ecuación del modelo con la cual podremos predecir el valor de compra según las variables predictoras de los datos de *hipoteca*.

```
[22] #Ecuación del modelo
print("Intercept:", Clasificacion.intercept_)
print('Coeficientes: \n', Clasificacion.coef_)
```

```
Intercept: [-8.79525793e-06]
Coeficientes:
[[ 3.19356279e-03 -3.37200749e-03 -5.93255515e-03 -4.30261149e-03
  4.14029139e-05 -3.35750119e-05  4.51243836e-05  1.22359214e-04
  2.48109431e-04]]
```

Prob =  $1/1+e^{-(a+bX)}$

$a+bX = 12.025 - 0.195\text{Texture} - 0.011\text{Area} - 0.707\text{Smoothness} - 2.592\text{Compactness} - 1.025\text{Symmetry} - 0.257\text{FractalDimension}$

Figura 11: Ecuación del modelo que podremos aplicar para clasificar el valor de compra según las variables predictoras.

Finalmente, realizamos el mismo procedimiento, pero para un valor de grupo de prueba diferente. En este caso, utilizaremos un grupo de prueba del 30%. Generando las probabilidades, tenemos que:

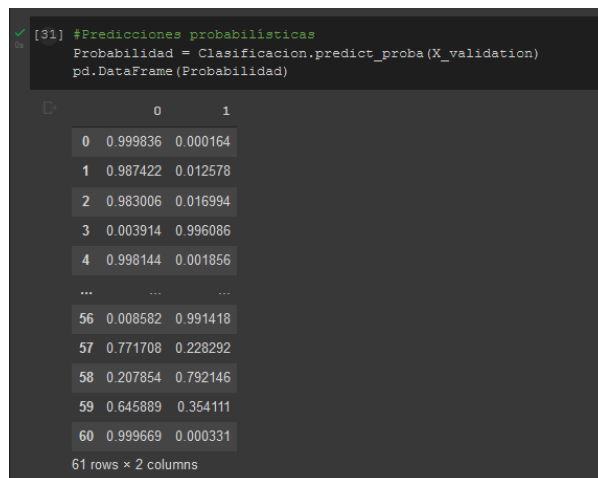


Figura 12: Probabilidades para un grupo de prueba del 30%.

Para este grupo de prueba, se obtuvo una exactitud promedio de la validación de 0.918032 y una matriz de clasificación como se muestra en la figura 14.

Clasificación	0	1
Reales		
0	41	0
1	5	15

Figura 13: Matriz de clasificación

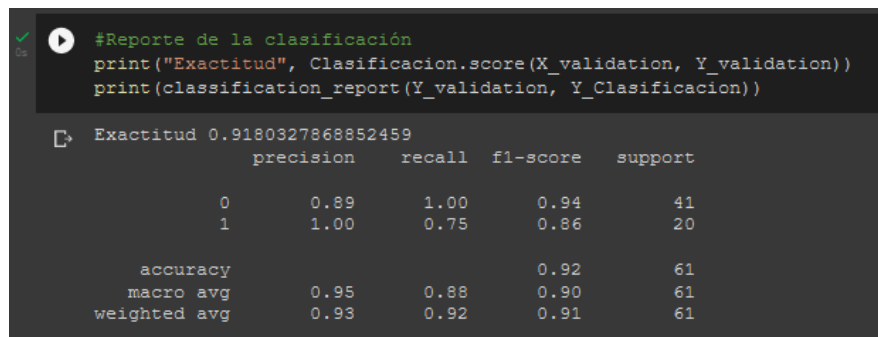


Figura 14: Reporte de la clasificación para grupo de prueba del 30%.

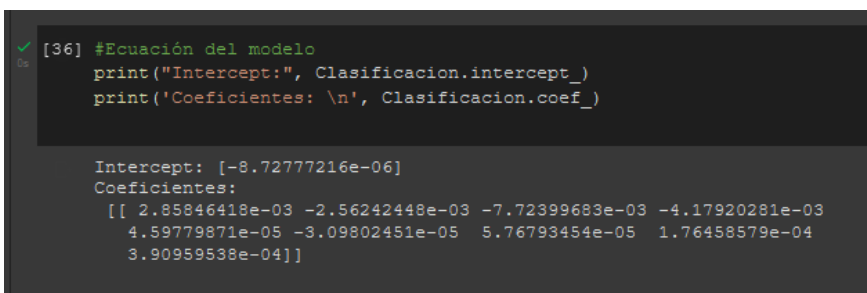


Figura 15: Coeficientes para modelo de clasificación con grupo de prueba del 30%.



## Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener la clasificación de registros de hipoteca, dividiendo los datos en valores de compra o no. Para ello, utilizamos el algoritmo de regresión logística con dos grupos de prueba diferentes (del 20% y del 30% del total del conjunto de datos) y utilizando 9 variables predictoras (*ingresos*, *gastos\_comunes*, *pago\_coche*, *gastos\_otros*, *ahorros*, *vivienda*, *estado\_civil*, *hijos* y *trabajo*).

Al utilizar la regresión logística, logramos hacer una clasificación a partir de los registros de hipotecas y obtener un modelo matemático que nos permita encontrar la clasificación (compra o no) a partir de ciertos datos (los de las variables predictoras).

Este tipo de algoritmo es muy utilizado en diversas aplicaciones. Si bien en esta práctica utilizamos este tipo de aprendizaje para datos de hipotecas, esto se podría traducir a datos de todo tipo como para obtener clasificaciones para datos de experimentos científicos, o la clasificación de tumores en malignos o benignos.

Vimos que, para los distintos tamaños de grupos de prueba, obtuvimos valores diferentes en los modelos matemáticos resultantes. Observamos que, en este caso, la exactitud del modelo fue mayor para un grupo de prueba menor y que en general la precisión del modelo también fue mejor. Se podrían hacer pruebas para distintos tamaños y determinar cuál es el que genera el modelo que se ajuste de mejor manera a los datos con los que se cuenta, evitando caer en un ajuste excesivo o sobre-ajuste de los datos.

En general, creo que se cumplieron los objetivos de la práctica y que ésta fue útil para comprender de mejor manera un ejemplo de aprendizaje supervisado sencillo y las posibles aplicaciones de este tipo de algoritmos.