MODELO DE RIESGO DE CRÉDITO ENFOCADO EN ENTIDADES FINANCIERAS

Jonathan urrego Zea **Juan Felipe Usuga Villegas**

Johan Sebastián Cano garcía

Julián David Ruiz Herrera

Raúl Vladimir Gaitán Vaca

18 de octubre de 2022

1. Introducción

El objetivo de este proyecto es construir en Python un modelo de riesgos de crédito que permita aproximar la probabilidad de que una persona incumpla con sus obligaciones crediticias usando un conjunto de datos previamente seleccionado y técnicas de machine learning. Se parte desde el procesamiento de los datos y finalmente se obtiene una scorecard a partir de la cual se puede asignar un puntaje crediticio a una persona y mediante este puntaje se obtiene información sobre el riesgo de otorgarle un crédito a dicha persona.

Se dará un enfoque general y no se tratarán en detalle todos los métodos utilizados para llegar al resultado final. Todos los procedimientos realizados están disponibles en el Notebook de Google Colab (click aquí) y cualquier persona puede realizar una consulta real o ficticia a través de la app web desarrollada conjuntamente a este modelo.

2. Sobre el conjunto de datos

Utilizamos un conjunto de datos disponible en Kaggle que contiene información sobre más de 450,000 créditos otorgados por Lending Club entre 2007 y 2014, este incluye información sobre el estado actual del crédito y sobre el deudor y su comportamiento. En el diccionario de datos se puede encontrar más información sobre cada una de las variables del conjunto de datos.

2.1. Exploración inicial de los datos

Como primer vistazo de los datos se puede ver que hay variables con más de 80 % de valores faltantes y por tanto cualquier método de imputación probablemente resulte en resultados desacertados. Además existen variables que no están relacionadas al riesgo crediticio o que no aportan nada al modelo y por tanto se eliminan.

```
# drop columns with more than 80% null values
df.dropna(thresh = df.shape[0]*0.2, how = 'all', axis = 1, inplace = True)

#drop all redundant and forward-looking columns
df.drop(columns = [ 'id', 'member_id', 'sub_grade', 'emp_title', 'url',
                           'desc', 'title',
                          'zip_code', 'next_pymnt_d', 'recoveries', '
                              collection_recovery_fee',
```

```
                              'total_rec_prncp', 'total_rec_late_fee'], inplace = True)

df.drop(columns = ["out_prncp","total_pymnt","total_rec_int","last_pymnt_amnt"],
    inplace = True)
```

2.2. Variable objetivo

Se selecciona la variable `loan_status` como variable objetivo dado que describe claramente si se ha cumplido adecuadamente con las obligaciones crediticias o no; podemos convencernos de esto viendo los estados posibles y sus proporciones tal como se indica en la tabla 1.

loan_status	Proporción
Current	0.480878
Fully Paid	0.396193
Charged Off	0.091092
Late (31-120 days)	0.014798
In Grace Period	0.006747
Does not meet the credit policy. Status:Fully Paid	0.004263
Late (16-30 days)	0.002612
Default	0.001784
Does not meet the credit policy. Status:Charged...	0.001632

Cuadro 1: Proporción por estado de crédito

En ese sentido se van a clasificar los siguientes estados como estados por defecto y se les asigna el número `0`:

- Charged Off
- Default
- Late (31–120 days)
- Does not meet the credit policy. Status:Charged Off

Todos los demás valores se los clasifica como buenos y se les asigna el número `1`

```
# create a new column based on the loan_status column that will be our target variable
df['good_bad'] = np.where(df.loc[:, 'loan_status'].isin(['Charged Off', 'Default','
    Late (31-120 days)', 'Does not meet the credit policy. Status:Charged Off']), 0, 1)

# Drop the original 'loan_status' column
df.drop(columns = ['loan_status'], inplace = True)
```

2.3. Datos de entrenamiento y de test

Ahora se divide el conjunto de datos separando un 80 % de los datos elegidos aleatoriamente los cuales constituyen el conjunto de entrenamiento y el 20restante constitye el conjunto de test. Además como un test preliminar se utilizará el método de validación cruzada sobre el conjunto de entrenamiento y posteriormente se va a validar el conjunto de test.

Es de notar que los datos están fuertemente sesgados hacia los buenos préstamos y por tanto se va a estratificar la división del conjunto de datos, de manera que el conjunto de test mantenga la distribución del conjunto inicial.

```
# split data into 80/20 while keeping the distribution of bad loans in test set same
    as that in the pre-split dataset
X = df.drop('good_bad', axis = 1)
y = df['good_bad']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 42, stratify = y)

```

```
# hard copy the X datasets to avoid Pandas' SetttingWithCopyWarning when we play around with this data later on.
# this is currently an open issue between Pandas and Scikit-Learn teams
X_train, X_test = X_train.copy(), X_test.copy()
```

2.4. Limpieza de datos

Con el objetivo de hacer más mánejables los datos para nuestros propósitos se realizan una serie de transformaciones. Para esto definiremos una función específica para cada una para usarlas tanto sobre el conjunto de entrenamiento como en el de test, a saber:

- Se remueve el texto de la columna `emp_length` y se convierte en numérica

```
def emp_length_converter(df, column):
    df[column] = df[column].str.replace('\+ years', '')
    df[column] = df[column].str.replace('< 1 year', str(0))
    df[column] = df[column].str.replace(' years', '')
    df[column] = df[column].str.replace(' year', '')
    df[column] = pd.to_numeric(df[column])
    df[column].fillna(value = 0, inplace = True)
```

- Las variables con fechas se convierten al formato `datetime`

```
def date_columns(df, column):
    # store current month
    today_date = pd.to_datetime('2020-08-01')
    # convert to datetime format
    df[column] = pd.to_datetime(df[column], format = "%b-%y")
    # calculate the difference in months and add to a new column
    df['mths_since_' + column] = round(pd.to_numeric((today_date - df[column]) / np.timedelta64(1, 'M')))
    # make any resulting -ve values to be equal to the max date
    df['mths_since_' + column] = df['mths_since_' + column].apply(
            lambda x: df['mths_since_' + column].max() if x < 0 else x)
    # drop the original date column
    df.drop(columns = [column], inplace = True)
```

- Se remueve el texto de la columna `term` y se convierte en numérica.

```
def loan_term_converter(df, column):
    df[column] = pd.to_numeric(df[column].str.replace(' months', ''))
```

3. Selección de variables

El siguiente paso es seleccionar las variables que itlizaremos apra nuestro modelo. Para esto se utiliza un test χ^2 para las variables categóricas y el estadístico F de ANOVA para las variables numéricas. El primer paso es separar el conjunto de entrenamiento en variables categóricas y variables numéricas.

```
X_train_cat = X_train.select_dtypes(include = 'object').copy()
X_train_num = X_train.select_dtypes(include = 'number').copy()
```

Una vez separado el conjunto de entrenamiento realizamos el test χ^2 y se obtiene que los p-valores de nuestras variables categóricas son los que se ven en la tabla 2.

De estas se toman las primeras cuatro variables. Para las variables numéricas antes de utilizar el estafístico F de ANOVA se imputan los valores faltantes con 0 directamente y se procede a seleccionar las 16 variables con mayor puntaje.

	Variable	p-valor
0	grade	0.000000
1	home_ownership	0.000000
2	verification_status	0.000000
3	purpose	0.000000
4	addr_state	0.000000
5	initial_list_status	0.000000
6	pymnt_plan	0.000923
7	application_type	1.000000

Cuadro 2: p-valores de test χ^2

```
    # since f_class_if does not accept missing values, we will do a very crude
        imputation of missing values
    X_train_num.fillna(X_train_num.mean(), inplace = True)
    # Calculate F Statistic and corresponding p values
    F_statistic, p_values = f_classif(X_train_num, y_train)
    # convert to a DF
    ANOVA_F_table = pd.DataFrame(data = {'Numerical_Feature': X_train_num.columns.
        values, 'F-Score': F_statistic, 'p values': p_values.round(decimals=10)})
    ANOVA_F_table.sort_values(by = ['F-Score'], ascending = False, ignore_index = True
        , inplace = True)
```

Y obtenemos

	Numerical_Feature	F-Score	p values
0	mths_since_last_pymnt_d	23513.805570	0.0
1	total_pymnt_inv	14784.534040	0.0
2	int_rate	11462.788313	0.0
3	out_prncp_inv	9633.442129	0.0
4	mths_since_last_credit_pull_d	7020.218888	0.0
5	mths_since_issue_d	2816.028871	0.0
6	inq_last_6mths	2003.820465	0.0
7	term	1590.811890	0.0
8	revol_util	931.880533	0.0
9	dti	863.811228	0.0
10	annual_inc	861.298449	0.0
11	tot_cur_bal	784.596617	0.0
12	mths_since_earliest_cr_line	447.541677	0.0
13	total_rev_hi_lim	377.070243	0.0
14	total_acc	208.751175	0.0
15	emp_length	146.954085	0.0

Cuadro 3: Test F de anova

Se realiza una verificación de colinealidad para identificar posibles varibles correlacionadas. Tal como vemos en la figura 1 las variables `out_prncp_inv` y `total_pymnt_inv` están altamente correlacionadas y por tanto se eliminan.

En la tabla 4 se pueden ver las variables seleccionadas para desarrollar el modelo

3.1. One-hot-encoding y conjunto de test

Un paso adicional que es necesario para el modelo es darle a las variables categóricas un valor numérico; esto se hará mediante one-hot-encoding. Este procedimiendo consiste en asignar a cada variable categórica una serie de variables ficticias que corresponden a cada una de sus categorías y asignar el valor 1 a la variable ficticia correspondiente a la categoría a la que pertenece la obervación y 0 a las demás. La convención que utilizamos para asignar el nombre a cada variable ficticia es: `nombre_de_la_variable:categoria`. Este procedimiento junto con todos los anteriores

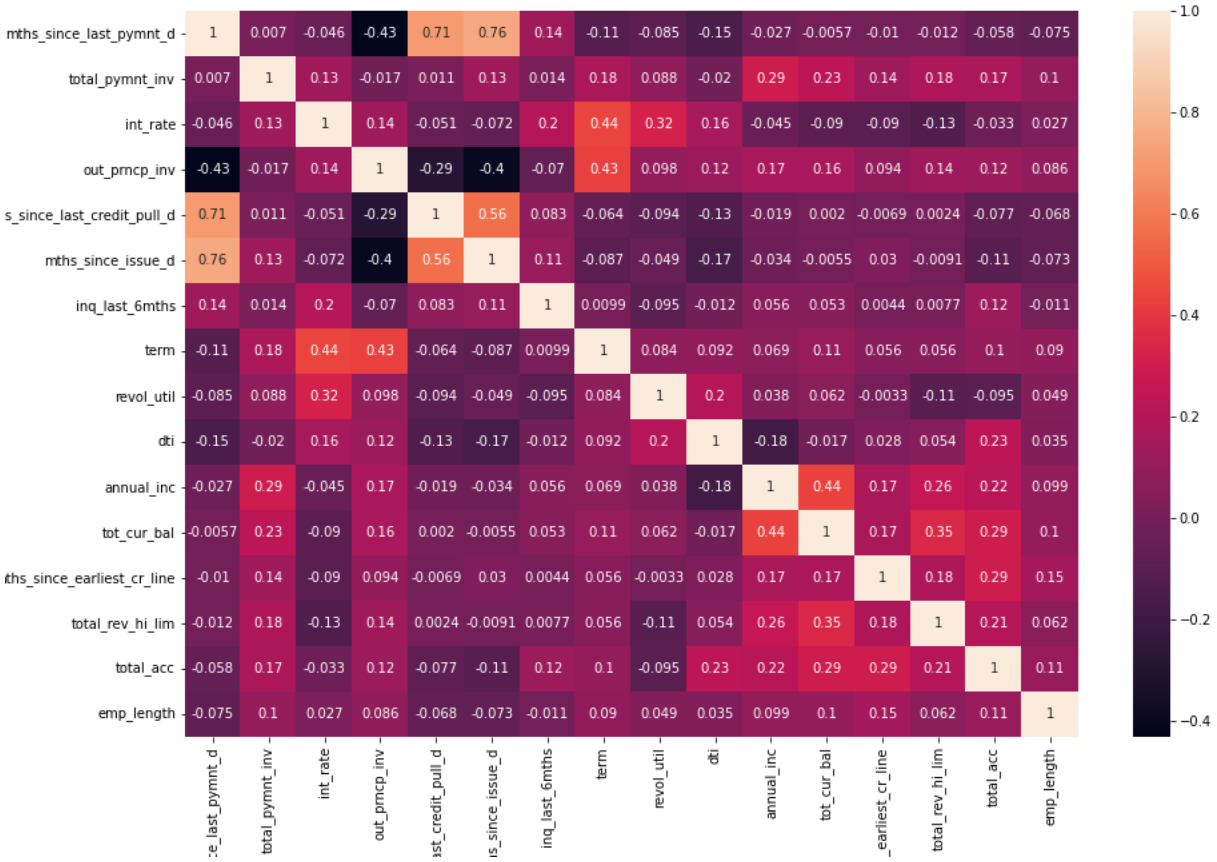

Figura 1: Correlación entre variables numéricas preseleccionadas

se hacen tanto para el conjunto de entrenamiento como para el conjunto de test. Los detalles del código necesario para hacer esto están disponibles en el notebook de Google Colab

4. Peso de evidencia (WoE) y valor de la informacion (IV)

En esta sección se discutirán dos valores que permiten conocer el poder predictivo de una variable, estos son el Peso de evidencia (WoE por sus siglas en inglés) y el valor de la información (IV por sus siglas en inglés); a partir de ahora nos referiremos a ellos como WoE e IV respectivamente. En esta documentación no se va a visualizar y discutir en detalle los valores de WoE e IV de cada variable; el lector puede ver el desarrollo y la visualización de estos en el notebook de Google Colab.

4.1. WoE

Para calcular el WoE se debe categorizar las variables numéricas de el modelo. Esto es posible separando las variables por bins y asignando una categoría específica para los valores faltantes; una vez hecho esto se puede conocer el WoE de cada categoría para una variable específica con la ecuación 1. El WoE nos da información sobre el poder predictivo de cada una de las categorías asociadas a una variable.

$$WoE = \ln(\frac{\%\text{ de buenos clientes}}{\%\text{ de malos clientes}}) \tag{1}$$

Nombre de variable	Tipo
term	int64
int_rate	float64
grade	object
emp_length	float64
home_ownership	object
annual_inc	float64
verification_status	object
purpose	object
dti	float64
inq_last_6mths	float64
revol_util	float64
total_acc	float64
tot_cur_bal	float64
total_rev_hi_lim	float64
mths_since_earliest_cr_line	float64
mths_since_issue_d	float64
mths_since_last_pymnt_d	float64
mths_since_last_credit_pull_d	float64

Cuadro 4: Variables seleccionadas

4.2. IV

El IV permite conocer el poder predictivo de la variable a partir del WoE; a diferencia de este último, con el IV podemos concoer el poder predictivo acumulado de todas las categorías de una variable y no solamente el de cada una de ellas. Se calcula mediante la ecuación 2

$$\sum(\ \%\ \text{de buenos clientes} - \ \%\ \text{de malos clientes}) \cdot WoE \tag{2}$$

En la tabla 5 se puede ver una regla común para clasificar el poder de predicción de una variable a partir del IV.

IV	Poder de predicción
<0.02	Predictor Inutil
(0.02, 0.1]	Predictor débil
(0.1, 0.3]	Predictor medio
(0.3, 0.5]	Predictor fuerte
>0. 5	Predictor suspicaz

Cuadro 5: Interpretación de IV

4.3. Variables finales

Una vez calculados y visualizados los IV de cada variable unimos algunas de las categorías que obtuvimos para calcular el WoE inicialmente siguiendo estas reglas:

- No se combina ni se deja una categoría particular para valores faltantes.
- Categorías con muy pocas observaciones se combinan con la siguiente.
- Se combinan categorías con WoE muy similar.
- Se ignoran variables con un IV bajo o muy alto.

El resultado final después de hacer las transformaciones correpondientes se puede ver en la tablas 6 y 7 el resultado. Este se separa en dos tablas por el bien de la legibilidad del reporte.

	Nombre de la variable
0	Intercept
1	grade:A
2	grade:B
3	grade:C
4	grade:D
5	grade:E
6	grade:F
7	home_ownership:OWN
8	home_ownership:OTHER_NONE_RENT
9	verification_status:Source Verified
10	verification_status:Verified
11	purpose:debt_consolidation
12	purpose:credit_card
13	purpose:educ__ren_en__sm_b__mov
14	purpose:vacation__house__wedding__med__oth
15	term:36
16	int_rate:<7.071
17	int_rate:7.071-10.374
18	int_rate:10.374-13.676
19	int_rate:13.676-15.74
20	int_rate:15.74-20.281
21	annual_inc:missing
22	annual_inc:<28,555
23	annual_inc:28,555-37,440
24	annual_inc:37,440-61,137
25	annual_inc:61,137-81,872
26	annual_inc:81,872-102,606
27	annual_inc:102,606-120,379
28	annual_inc:120,379-150,000
29	dti:<=1.6
30	dti:1.6-5.599
31	dti:5.599-10.397
32	dti:10.397-15.196
33	dti:15.196-19.195
34	dti:19.195-24.794
35	dti:24.794-35.191

Cuadro 6: Variables de la 1 a la 35

5. Construcción del modelo

En esta sección se ajusta un modelo de regresión logística sobre nuestro conjunto de entrenamiento y se usa validación cruzada sobre el mismo como test preliminar.

5.1. Entrenamiento del modelo

En el modelo de regresión logística se define el parámetro `class_weight` como `balanced` con el fin de que el modelo tenga un aprendizaje sensible a los costos, es decir que penalice más los falsos negativos que los falsos positivos.

Se usará además el área bajo una curva ROC como métrica de evaluación. Se usará el método `RepeatedStratifiedKFold` para la validación de los datos.

```
# define modeling pipeline
reg = LogisticRegression(max_iter=1000, class_weight = 'balanced')
woe_transform = WoE_Binning(X)
pipeline = Pipeline(steps=[('woe', woe_transform), ('model', reg)])

```

	Feature name
36	inq_last_6mths:missing
37	inq_last_6mths:0
38	inq_last_6mths:1-2
39	inq_last_6mths:3-4
40	revol_util:missing
41	revol_util:<0.1
42	revol_util:0.1-0.2
43	revol_util:0.2-0.3
44	revol_util:0.3-0.4
45	revol_util:0.4-0.5
46	revol_util:0.5-0.6
47	revol_util:0.6-0.7
48	revol_util:0.7-0.8
49	revol_util:0.8-0.9
50	revol_util:0.9-1.0
51	total_rev_hi_lim:missing
52	total_rev_hi_lim:<6,381
53	total_rev_hi_lim:6,381-19,144
54	total_rev_hi_lim:19,144-25,525
55	total_rev_hi_lim:25,525-35,097
56	total_rev_hi_lim:35,097-54,241
57	total_rev_hi_lim:54,241-79,780
58	mths_since_earliest_cr_line:missing
59	mths_since_earliest_cr_line:<125
60	mths_since_earliest_cr_line:125-167
61	mths_since_earliest_cr_line:167-249
62	mths_since_earliest_cr_line:249-331
63	mths_since_earliest_cr_line:331-434
64	mths_since_issue_d:<79
65	mths_since_issue_d:79-89
66	mths_since_issue_d:89-100
67	mths_since_issue_d:100-122
68	mths_since_last_credit_pull_d:missing
69	mths_since_last_credit_pull_d:<56
70	mths_since_last_credit_pull_d:56-61
71	mths_since_last_credit_pull_d:61-75

Cuadro 7: Variables de la 36 a la 71

```
# define cross-validation criteria
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=1)

# fit and evaluate the logistic regression pipeline with cross-validation as defined
    in cv
scores = cross_val_score(pipeline, X_train, y_train, scoring = 'roc_auc', cv = cv)
AUROC = np.mean(scores)
GINI = AUROC * 2 - 1
```

Una vez que la validación es satisfactoria se ajusta el pipeline al conjunto de entrenamiento completo. En la tabla 8 se observa una muestra de 20 de los coeficientes que arroja el modelo.

5.2. Hora de predecir

Una vez el modelo esté entrenado el siguiente paso es validarlo. Para esto se predice la probabilidad de que se dé un valor positivo en el conjunto de test, esto es, que el valor sea . El resultado se guarda en un dataframe distinto junto con la clase verdadera. En la tabla 9 se puede observar un ejemplo en 10 observaciones.

	Nombre de la variable	Coeficiente
0	Intercept	-1.850463
1	grade:A	1.187734
2	grade:B	0.906924
3	grade:C	0.731165
4	grade:D	0.578035
5	grade:E	0.377329
6	grade:F	0.230648
7	home_ownership:OWN	-0.035009
8	home_ownership:OTHER_NONE_RENT	-0.136284
9	verification_status:Source Verified	-0.092189
10	verification_status:Verified	-0.106930
11	purpose:debt_consolidation	-0.096456
12	purpose:credit_card	-0.004621
13	purpose:educ__ren_en__sm_b__mov	-0.398020
14	purpose:vacation__house__wedding__med__oth	-0.037681
15	term:36	0.063185
16	int_rate:<7.071	1.309163
17	int_rate:7.071-10.374	0.727694
18	int_rate:10.374-13.676	0.404068
19	int_rate:13.676-15.74	0.248962

Cuadro 8: Muestra de 20 coeficientes

	Clase real	Probabilidad obtenida
395346	1	0.537477
376583	1	0.839575
297790	1	0.562390
47347	1	0.377425
446772	0	0.322891
415533	1	0.770404
355671	1	0.765598
11223	1	0.473343
411384	1	0.762354
21993	1	0.711522

Cuadro 9: Respuestas de ejemplo

Además, se dibujan las curva ROC y PR como se observa en las figuras 2 y 3 respectivamente.

Además se calcula el área bajo la curva ROC y se obtiene un valor de `0.7172` y el coeficiente GINI y se obtiene un valor de `0.4345`.

5.3. Desarrollo de la scorecard

Para el desarrollo de la scorecard el primer paso es elegir los valores máximos y mínimos que podría tomar el puntaje. Para esto se eligen inicialmente los mismos que utiliza la FICO en Estados Unidos; es decir, el mínimo se toma en 350 y el máximo se toma en 850. Además se junta la tabla que se ilustra mediante la tabla 8 con una tabla que contiene el nombre original de cada una de las variables ficticias. Posteriormente se escala cada uno de los coeficientes que retorna la regresión logística al rango que hemos seleccionado al igual que el intercepto, este es el valor inicial que puede tomar un usuario.

```
# create a new dataframe with one column with values from the 'reference_categories'
    list
df_ref_categories = pd.DataFrame(ref_categories, columns = ['Feature name'])
# We create a second column, called 'Coefficients', which contains only 0 values.
df_ref_categories['Coefficients'] = 0

```

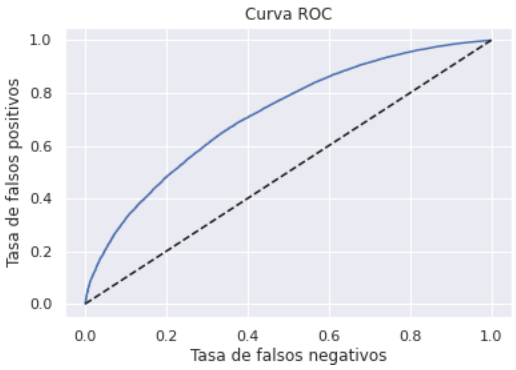

Figura 2: Curva ROC

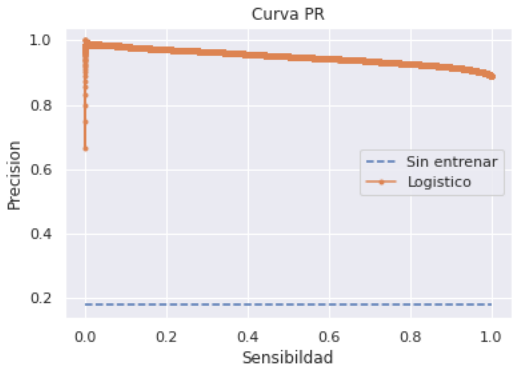

Figura 3: Curva PR

```
# Concatenates two dataframes
df_scorecard = pd.concat([summary_table, df_ref_categories])
# reset the index
df_scorecard.reset_index(inplace = True)

# create a new column, called 'Original feature name', which contains the value of the
    'Feature name' column
df_scorecard['Original feature name'] = df_scorecard['Feature name'].str.split(':').
    str[0]

# Define the min and max threshholds for our scorecard
min_score = 300
max_score = 850

# calculate the sum of the minimum coefficients of each category within the original
    feature name
min_sum_coef = df_scorecard.groupby('Original feature name')['Coefficients'].min().sum
    ()
# calculate the sum of the maximum coefficients of each category within the original
    feature name
max_sum_coef = df_scorecard.groupby('Original feature name')['Coefficients'].max().sum
    ()
# create a new column that has the imputed calculated Score based scaled from the
    coefficients
df_scorecard['Score - Calculation'] = df_scorecard['Coefficients'] * (max_score -
    min_score) / (
        max_sum_coef - min_sum_coef)
# update the calculated score of the Intercept
```

```
df_scorecard.loc[0, 'Score - Calculation'] = (
        (df_scorecard.loc[0,'Coefficients'] - min_sum_coef) /
        (max_sum_coef - min_sum_coef
        )) * (max_score - min_score) + min_score
# round the values of the 'Score - Calculation' column and store them in a new column
df_scorecard['Score - Preliminary'] = df_scorecard['Score - Calculation'].round()

# check the min and max possible scores of our scorecard
min_sum_score_prel = df_scorecard.groupby('Original feature name')['Score -
    Preliminary'].min().sum()
max_sum_score_prel = df_scorecard.groupby('Original feature name')['Score -
    Preliminary'].max().sum()
print(min_sum_score_prel)
print(max_sum_score_prel)
# look like we can get by deducting 1 from the Intercept
df_scorecard['Score - Final'] = df_scorecard['Score - Preliminary']
df_scorecard.loc[0, 'Score - Final'] = 598
```

5.4. Calcular los puntajes de conjunto de test

Finalmente para calcular el puntaje de cada observación en el conjunto de test es suficiente con realizar un producto punto entre el vector de observación y la columna de puntajes finales.

5.5. Selección del puntaje de corte

Para seleccionar el mejor puntaje de corte se utiliza el estadístico J de Youden.

```
# Calculate Youden's J-Statistic to identify the best threshhold
J = tpr - fpr
# locate the index of the largest J
ix = np.argmax(J)
best_thresh = thresholds[ix]
print('Best Threshold: %f' % (best_thresh))
```

El resultado que se obtiene es 0.360325, lo que significa que observaciones con una probabilidad predicha mayor a esta se clasifican como por defecto y viceversa.

Además, en la figura 4 se puede observar la ditribución de los puntajes del conjunto de test. Nótese que los puntajes están entre 500 y 1000, de lo cual se modifica el intervalo de puntajes posibles y se deja como valor mínimo 450 y como máximo 1000.

6. Conclusión

Todo este trabajo ha sido con el objetivo de obtener un modelo fácilmente utilizable por una entidad financiera que permite predecir la probabilidad de que una persona incumpla con sus obligaciones crediticias y se entrega en forma de scorecard. Ahora es posible para cualquier entidad utilizarlo para tomar decisiones sobre los créditos que se le otorgarán al cliente y es fácilmanete integrable con un app web de uso simple tal como se muestra en este ejemplo:

https://main–heartfelt-cupcake-3270df.netlify.app/

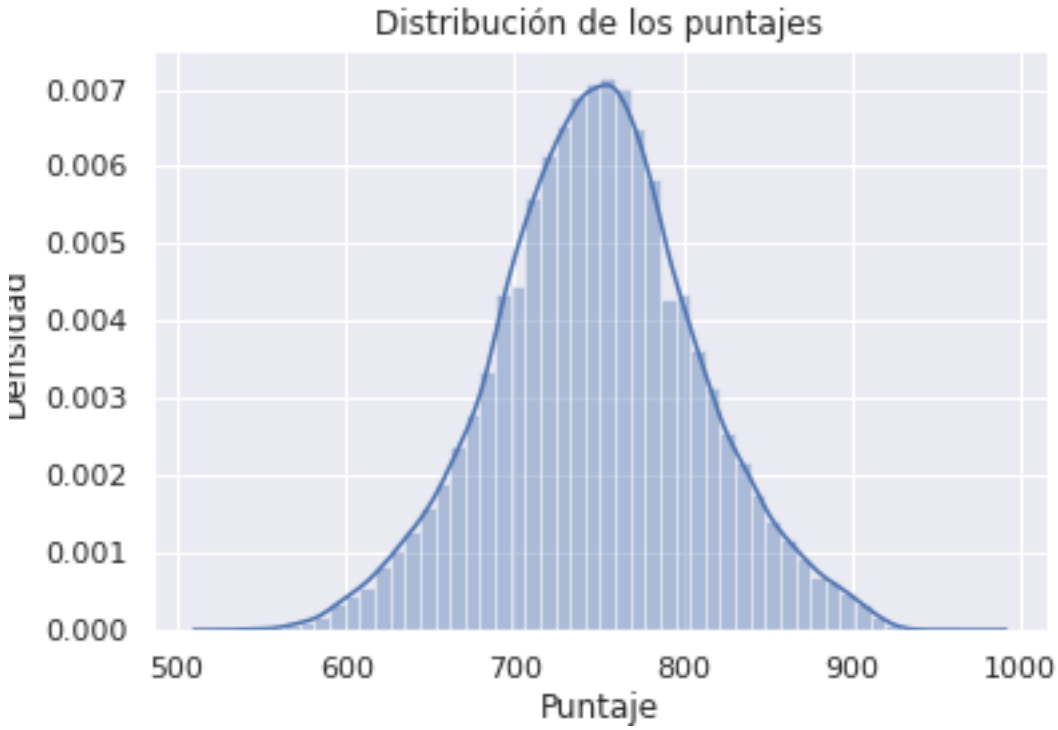

Figura 4: Distribución de los puntajes

Referencias

[1] Gareth James • Daniela Witten • Trevor Hastie Robert Tibshirani: An Introduction to Statistical Learning with Apllications in R ..

[2] https://towardsdatascience.com/how-to-develop-a-credit-risk-model-and-scorecard-91335fc01f03

[3] https://towardsdatascience.com/how-to-find-the-best-predictors-for-ml-algorithms-4b28a71a8a80

[4] https://medium.com/swlh/how-to-quantify-risk-and-creditworthiness-c76725bc2380

[5] https://towardsdatascience.com/how-to-avoid-potential-machine-learning-pitfalls-a08781f3518e

[6] https://towardsdatascience.com/how-to-effectively-predict-imbalanced-classes-in-python-e8cd3b5720c4

[7] https://datapeaker.com/big-data/asistente-virtual-de-ia-usando-python/