



**Proyecto final**

**Gálvez Gaxiola André**

**Mtro. Hinojosa Palafox Eduardo Antonio**

**10/12/2022**

<b>Introducción</b>	<b>3</b>
<b>Descripción del problema a desarrollar</b>	<b>4</b>
<b>Descripción del conjunto de datos</b>	<b>4</b>
<b>Descripción de la solución propuesta</b>	<b>5</b>
<b>Descripción del código utilizado</b>	<b>6</b>
<b>Resultados</b>	<b>10</b>
<b>Conclusiones</b>	<b>11</b>
<b>Referencias</b>	<b>12</b>
<b>Liga a Github</b>	<b>13</b>

# Introducción

El documento habla de un caso de estudio en el cual está basado un banco. Esta compañía busca la manera de llevar a sus clientes con deudas o responsabilidades hacia clientes que tengan créditos personales con el banco.

Se utilizará un dataset que consta de información de 5000 personas, de las cuales solo 480 aceptaron el crédito personal que se les fue ofrecido por la campaña.

El objetivo de esta investigación es incrementar el porcentaje de personas que podrán obtener créditos personales utilizando el mínimo presupuesto en futuras campañas.

# Descripción del problema a desarrollar

El problema a solucionar es el poder predecir si el cliente comprará o no un crédito personal, basándonos en la información que tenemos sobre la campaña anterior y el comportamiento que reflejaron las personas.

## Descripción del conjunto de datos

El archivo contiene datos de 5000 clientes. La información incluye datos geográficos, relación del cliente con el banco y la respuesta del cliente a la última campaña de créditos personales.

- **ID** : ID del cliente
- **Age** : Edad del cliente
- **Experience** : #años de experiencia
- **Income** : Ingreso anual de cliente (\$000)
- **ZIP Code** : Código ZIP de casa.
- **Family** : Número de miembros familiares
- **CC Avg** : Promedio de Gasto en Tarjetas de crédito por mes (\$000)
- **Education** : Nivel de Educación.
  - 1: No graduado;
  - 2: Graduado;
  - 3: Profesional
- **Mortgage** : Valor de hipoteca, si existiera. (\$000)
- **Personal Loan** : Aceptó el cliente el crédito personal ofrecido en la última campaña?
- **Securities Account** : El cliente tiene una cuenta con activos financieros con el banco?
- **CD Account** : El cliente tiene una cuenta certificada de depósito(CD) con el banco?
- **Online** : ¿El cliente usa medios de banca por internet?
- **Credit card** : El cliente usa tarjeta de crédito manejada por algún otro banco?

## Descripción de la solución propuesta

Elegir un modelo óptimo que me indique si el cliente comprará o no el préstamo usando Machine Learning y la data adjunta.

Está relacionada a evaluar el modelo de regresión logística y el de árboles de decisión e indicar cuál vendría a ser el modelo óptimo y con qué características para decidir si el cliente tomará o no el préstamo personal

# Descripción del código utilizado

Procesar el zipcode . característica categórica, en consecuencia predictor de la variable objetivo. Luego analizamos algún patrón en la localización para aquellos clientes que han realizado algún préstamo en la previa campaña. Tratando de reducir las categorías:

```
[7] df_loan.drop(['ID'],axis=1,inplace=True)

[10] df_loan.rename(columns={"ZIP Code":"ZIPCode","Personal Loan":"PersonalLoan","Securities Account":"SecuritiesAccount",
                           "CD Account":"CDAccount"},inplace=True)
```

Creamos un diccionario de datos

```
[14] import zipcodes as zcode # to get zipcodes

dict_zip={}
for zipcode in list_zipcode:
    my_city_county = zcode.matching(zipcode.astype('str'))
    if len(my_city_county)==1: # si existe zipcode entonces asignar un estado,sino asignar el zipcode al estado
        county=my_city_county[0].get('county')
    else:
        county=zipcode
    dict_zip.update({zipcode:county})
```

Actualizamos los estados


```
df_loan['Estado']=df_loan['ZIPCode'].map(dict_zip)
```

```
df_loan.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Age                   5000 non-null   int64
1   Experience             5000 non-null   int64
2   Income                5000 non-null   int64
3   ZIPCode               5000 non-null   int64
4   Family                5000 non-null   int64
5   CCAvg                 5000 non-null   float64
6   Education             5000 non-null   int64
7   Mortgage              5000 non-null   int64
8   PersonalLoan          5000 non-null   int64
9   SecuritiesAccount     5000 non-null   int64
10  CDAccount              5000 non-null   int64
11  Online                5000 non-null   int64
12  CreditCard            5000 non-null   int64
13  Estado                5000 non-null   object
dtypes: float64(1), int64(12), object(1)
memory usage: 547.0+ KB
```


Cambiamos unas variables de tipo entero y objeto a categoricas

```
category_col = ['PersonalLoan', 'SecuritiesAccount', 'Family', 'CDAccount', 'Online', 'CreditCard', 'ZIPCode', 'Education', 'Estado']  
df_loan[category_col] = df_loan[category_col].astype('category')
```

✓ 0 s  df\_loan.info()

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5000 entries, 0 to 4999  
Data columns (total 14 columns):  
#   Column                Non-Null Count  Dtype    
---  ---                  
0   Age                   5000 non-null   int64    
1   Experience             5000 non-null   int64    
2   Income                5000 non-null   int64    
3   ZIPCode               5000 non-null   category  
4   Family                5000 non-null   category  
5   CCAvg                5000 non-null   float64  
6   Education             5000 non-null   category  
7   Mortgage              5000 non-null   int64    
8   PersonalLoan          5000 non-null   category  
9   SecuritiesAccount      5000 non-null   category  
10  CDAccount             5000 non-null   category  
11  Online                5000 non-null   category  
12  CreditCard            5000 non-null   category  
13  Estado                5000 non-null   category  
dtypes: category(9), float64(1), int64(4)  
memory usage: 266.4 KB
```

Hacemos el procesamiento y validamos negativos como valores ceros para la Experiencia

✓ 0 s  df\_loan[df\_loan['Experience']<0]['Age'].describe()

```
count    52.00000  
mean     24.51923  
std       1.47516  
min      23.00000  
25%      24.00000  
50%      24.00000  
75%      25.00000  
max      29.00000  
Name: Age, dtype: float64
```

Siguen existiendo valores negativos en varias edades, así que trabajaremos con valores absolutos de experiencia

```
df_loan.loc[df_loan['Experience']<0, 'Experience']=np.abs(df_loan['Experience'])
df_loan[df_loan['Experience']==0]['Age'].describe()
```

count	66.00000
mean	25.63636
std	1.14538
min	24.00000
25%	25.00000
50%	26.00000
75%	26.00000
max	30.00000
Name: Age, dtype: float64	

Dividimos la data de entrenamiento en una relación 30-70

```
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.30, random_state = 1, stratify=Y)
```

Estandarizamos y hacemos un escalado de datos

```
[75] from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fitting Standard Scaller
X_scaler = scaler.fit(X_train)

# Scaling data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)

X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X_test.columns)

X_train_scaled_df.index=np.arange(len(X_train_scaled_df))
X_test_scaled_df.index=np.arange(len(X_test_scaled_df))
y_train.index=np.arange(len(y_train))
y_test.index=np.arange(len(y_test))
```



Obtenemos el performance de nuestro modelo en base a métricas

```
[78] lr = LogisticRegression(solver='newton-cg', random_state=1, fit_intercept=False, class_weight={0:0.15, 1:0.85})
      model = lr.fit(X_train_scaled_df, y_train)

      statmodel=0

      scores_Sklearn = get_metrics_score(model, X_train_scaled_df, X_test_scaled_df, y_train, y_test, statmodel)
```

MODEL PERFORMANCE			
Accuracy	: Train:	1.0	Test: 0.997
Recall	: Train:	1.0	Test: 1.0
Precision	: Train:	1.0	Test: 0.966
F1	: Train:	1.0	Test: 0.983

Hacemos lo mismo que con el otro modelo, y usamos one hot encoding para nuestras variables “dummys”

```
[79] df_Decision = df_loan.copy()

[80] #Eliminar columnas no necesarias
      df_Decision.drop(columns=["ZIPCode", "Estado", 'Experience'], inplace=True)

[81] X_dt = df_Decision.drop('PersonalLoan', axis=1)
      y_dt = df_Decision['PersonalLoan']

[82] X_dt = df_Decision.drop('PersonalLoan', axis=1)
      y_dt = df_Decision['PersonalLoan']

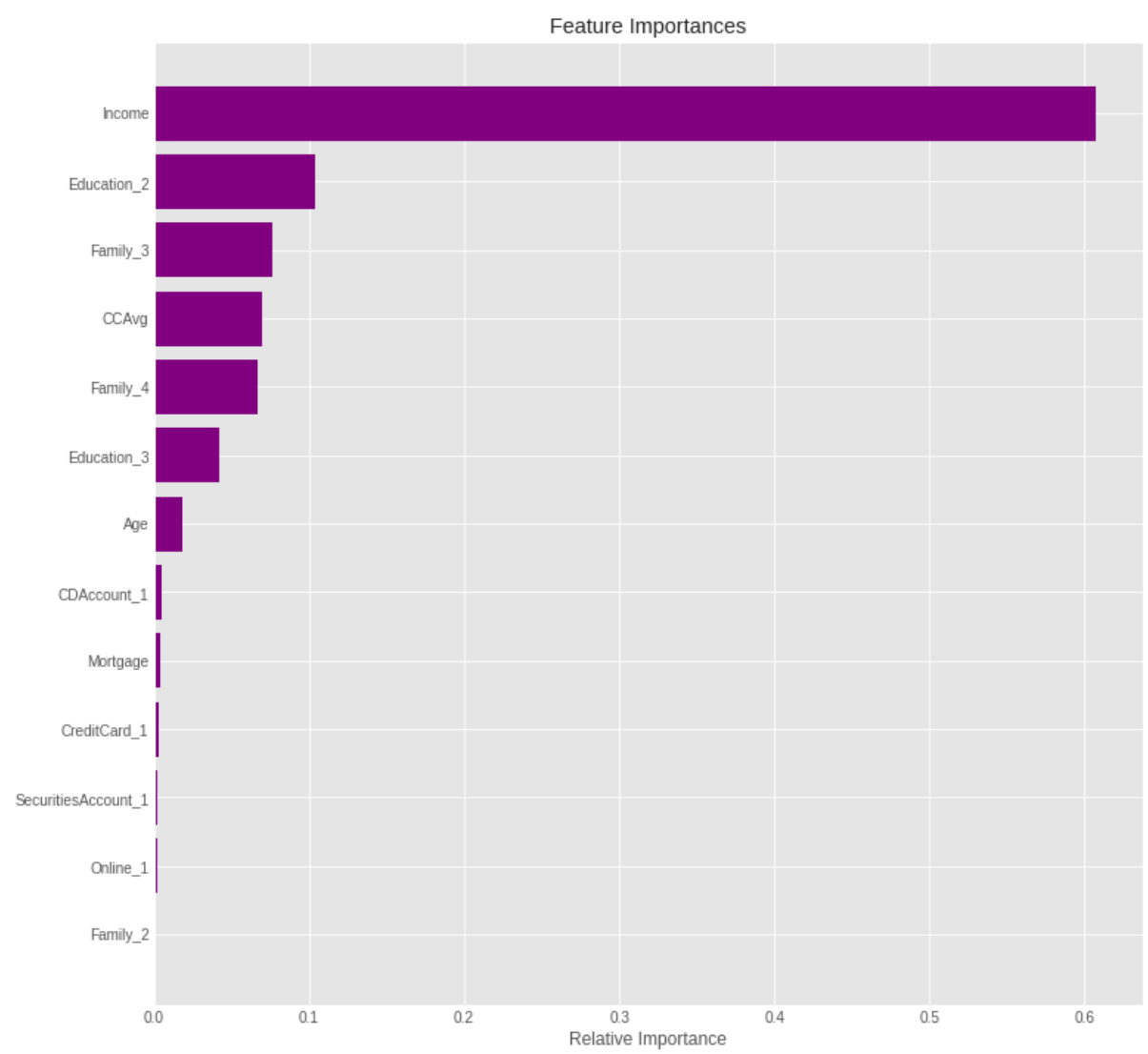
[83] oneHotCols=X_dt.select_dtypes(exclude='number').columns.to_list()
      X_dt=pd.get_dummies(X_dt,columns=oneHotCols,drop_first=True)
      # Spliting data set
      X_train_dt, X_test_dt, y_train_dt, y_test_dt = train_test_split(X_dt, y_dt, test_size=0.3, random_state=1, stratify=y_dt)
```

Y obtenemos las métricas de nuestro modelo

```
[85] model = DecisionTreeClassifier(criterion = 'gini', class_weight={0:0.15, 1:0.85}, random_state=1)
      model.fit(X_train_dt, y_train_dt)
      get_recall_score(model)
```

Accuracy	: Train :	1.0	Test: 0.9786666666666667
Recall	: Train :	1.0	Test: 0.875

# Resultados



	Model	Train_accuracy	Test_accuracy	Train_Recall	Test_Recall
0	Logisitic Regression with Optimal Threshold 0.104	0.92000	0.91000	0.90000	0.88000
1	Initial decision tree model	1.00000	0.98000	1.00000	0.86000
2	Decision treee with hyperparameter tuning	0.99000	0.98000	0.92000	0.84000

3	Decision tree with post-pruning	0.98000	0.97000	0.98000	0.96000
---	---------------------------------	---------	---------	---------	---------

## Conclusiones

Analizamos los datos para la campaña de préstamos personales usando análisis exploratorio y diferentes modelos como Análisis de regresión Logística y clasificación de árbol de decisión para construir la probabilidad en la compra de un préstamo.

Primero se usó la regresión logística y la métrica de rendimiento fue el Recall. Las más importantes características para la clasificación fueron: Income, Education, CD account, Family and CC Avg.

Árbol de decisión pueden fácilmente hacer overfit. Ellos necesitan preprocesar menos datos comparados a la regresión logística son fáciles de entender.

Se usó árboles de decisión con pre pruning y post pruning. El modelo pos pruning dió un 96% recall con 97 % de exactitud.

Ingresos(Income), Clientes con grado de graduado, clientes con 3 miembros de familia son algunos de los mayores variables en la predicción de si el cliente comprará o no un préstamo personal.

# Referencias

<https://www.kaggle.com/datasets/luisenriquesguerrero/creditos-personales-actualizado>

Liga a Github

<https://github.com/a-galvez08/Proyecto-AprendizajeAutomatico>