Proyecto Final Dia 4

1. Importar el conjunto de datos "UCI_Credit_Card.csv"

```
import pandas as pd

df_UCI_Credit_Card = pd.read_csv('UCI_Credit_Card.csv')
```

```
pd.set_option('display.max_columns', None)
```

2. Realizar el analisis exploratorio de datos y la visualizacion

```
df_UCI_Credit_Card
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AI
0	1	20000.0	2	2	1	24	2	2	-1	-1	-2	-2	3913.0	3102.0	689.0	
1	2	120000.0	2	2	2	26	-1	2	0	0	0	2	2682.0	1725.0	2682.0	327
2	3	90000.0	2	2	2	34	0	0	0	0	0	0	29239.0	14027.0	13559.0	1433
3	4	50000.0	2	2	1	37	0	0	0	0	0	0	46990.0	48233.0	49291.0	2831
4	5	50000.0	1	2	1	57	-1	0	-1	0	0	0	8617.0	5670.0	35835.0	2094
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000.0	1	3	1	39	0	0	0	0	0	0	188948.0	192815.0	208365.0	8800
29996	29997	150000.0	1	3	2	43	-1	-1	-1	-1	0	0	1683.0	1828.0	3502.0	897
29997	29998	30000.0	1	2	2	37	4	3	2	-1	0	0	3565.0	3356.0	2758.0	2087
29998	29999	80000.0	1	3	1	41	1	-1	0	0	0	-1	-1645.0	78379.0	76304.0	5277
29999	30000	50000.0	1	2	1	46	0	0	0	0	0	0	47929.0	48905.0	49764.0	3653

30000 rows × 25 columns

```
df_UCI_Credit_Card.describe().round(3)
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6
count	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000	30000.000
mean	15000.500	167484.323	1.604	1.853	1.552	35.486	-0.017	-0.134	-0.166	-0.221	-0.266	-0.291
std	8660.398	129747.662	0.489	0.790	0.522	9.218	1.124	1.197	1.197	1.169	1.133	1.150
min	1.000	10000.000	1.000	0.000	0.000	21.000	-2.000	-2.000	-2.000	-2.000	-2.000	-2.000
25%	7500.750	50000.000	1.000	1.000	1.000	28.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
50%	15000.500	140000.000	2.000	2.000	2.000	34.000	0.000	0.000	0.000	0.000	0.000	0.000
75%	22500.250	240000.000	2.000	2.000	2.000	41.000	0.000	0.000	0.000	0.000	0.000	0.000
max	30000.000	1000000.000	2.000	6.000	3.000	79.000	8.000	8.000	8.000	8.000	8.000	8.000

```
df_UCI_Credit_Card.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
```

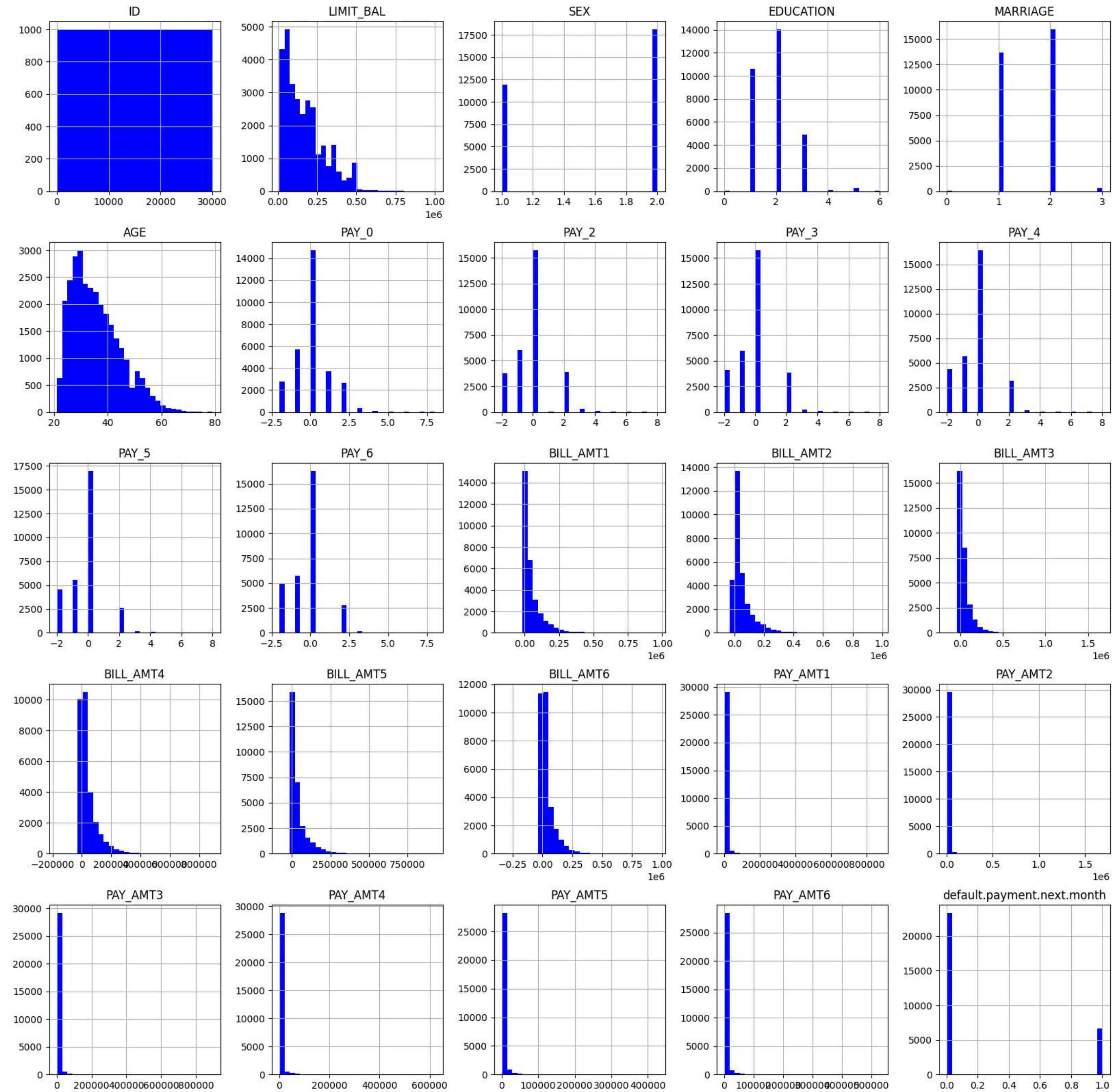
```
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

Al realizar un info al dataset se observa que no poseemos valores nulos en ninguna de sus columnas, de esta forma no tenemos que lidiar con el tratamiento de los nulos

```
df_UCI_Credit_Card.isnull().sum()
```

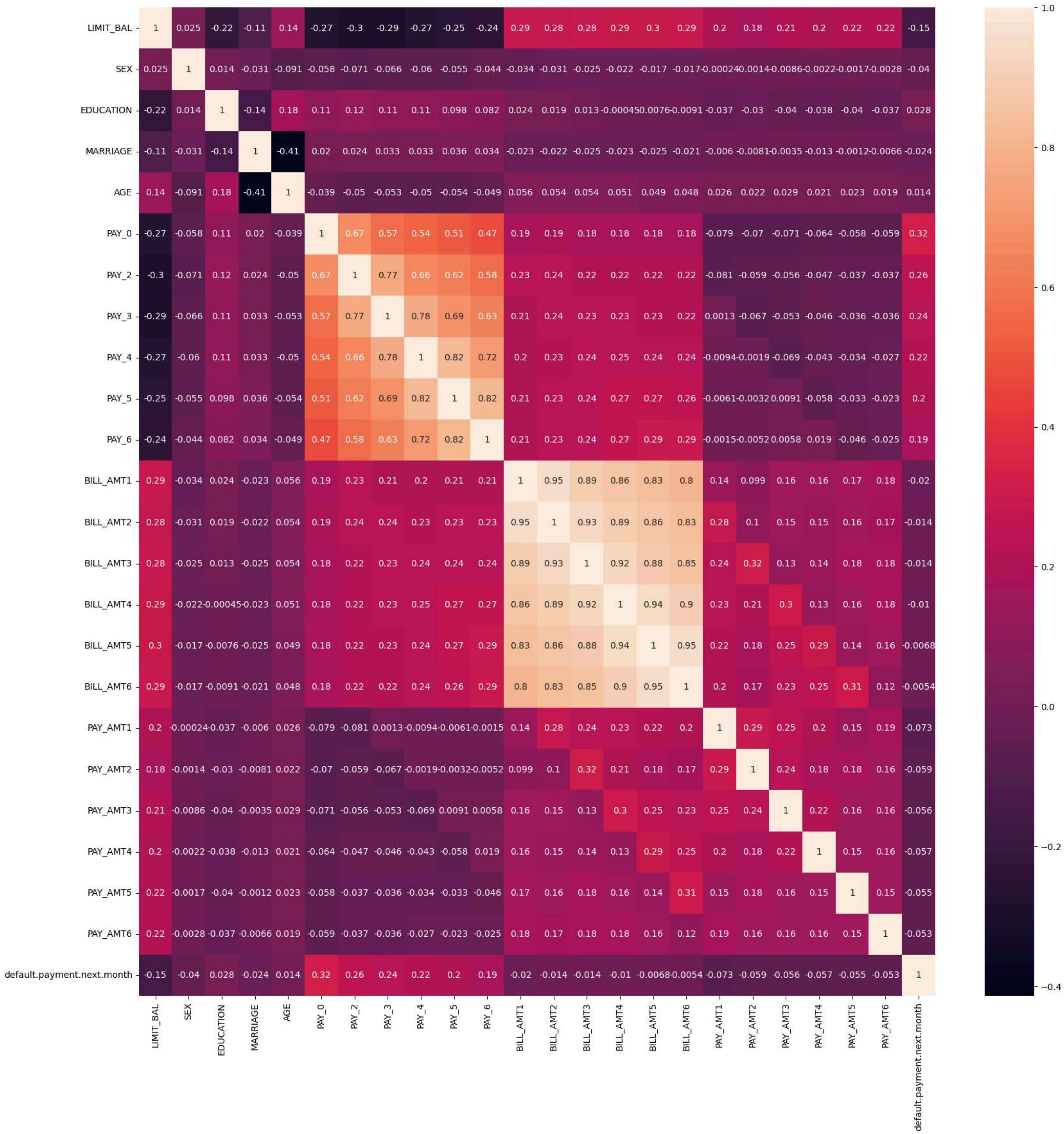
```
ID                            0
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         0
PAY_2                         0
PAY_3                         0
PAY_4                         0
PAY_5                         0
PAY_6                         0
BILL_AMT1                     0
BILL_AMT2                     0
BILL_AMT3                     0
BILL_AMT4                     0
BILL_AMT5                     0
BILL_AMT6                     0
PAY_AMT1                      0
PAY_AMT2                      0
PAY_AMT3                      0
PAY_AMT4                      0
PAY_AMT5                      0
PAY_AMT6                      0
default.payment.next.month    0
dtype: int64
```

```
df_UCI_Credit_Card.hist(bins=30, figsize=(20, 20), color='b');
```


ID
LIMIT_BAL
SEX
EDUCATION
MARRIAGE
AGE
PAY_0
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
PAY_AMT1
PAY_AMT2
PAY_AMT3
PAY_AMT4
PAY_AMT5
PAY_AMT6
default.payment.next.month

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

```
import seaborn as sns
import matplotlib.pyplot as plt

f, ax = plt.subplots(figsize=(20, 20))
sns.heatmap(df_UCI_Credit_Card.corr(), annot=True);
```


LIMIT_BAL
SEX
EDUCATION
MARRIAGE
AGE
PAY_0
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
PAY_AMT1
PAY_AMT2
PAY_AMT3
PAY_AMT4
PAY_AMT5
PAY_AMT6
default.payment.next.month
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

3. Preparar los datos del modelo y dividirlos en entrenamiento y test

Como hay variables categoricas, se debe hacer un one hot encoded

```
from sklearn.preprocessing import OneHotEncoder

onehotencoder = OneHotEncoder()
df_cat = onehotencoder.fit_transform(df_UCI_Credit_Card[['SEX','EDUCATION','MARRIAGE']].copy()).toarray()
df_num = df_UCI_Credit_Card.select_dtypes(include='number').copy()
df_num.drop(['SEX','EDUCATION','MARRIAGE'], axis=1, inplace=True)
```

```
df_cat = pd.DataFrame(df_cat)
```

```
X = pd.concat([df_num, df_cat], axis=1)
X
```

	LIMIT_BAL	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AM
0	20000.0	24	2	2	-1	-1	-2	-2	3913.0	3102.0	689.0	0.0	0.0	0.0	0
1	120000.0	26	-1	2	0	0	0	2	2682.0	1725.0	2682.0	3272.0	3455.0	3261.0	0
2	90000.0	34	0	0	0	0	0	0	29239.0	14027.0	13559.0	14331.0	14948.0	15549.0	1518
3	50000.0	37	0	0	0	0	0	0	46990.0	48233.0	49291.0	28314.0	28959.0	29547.0	2000
4	50000.0	57	-1	0	-1	0	0	0	8617.0	5670.0	35835.0	20940.0	19146.0	19131.0	2000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	220000.0	39	0	0	0	0	0	0	188948.0	192815.0	208365.0	88004.0	31237.0	15980.0	8500
29996	150000.0	43	-1	-1	-1	-1	0	0	1683.0	1828.0	3502.0	8979.0	5190.0	0.0	1837
29997	30000.0	37	4	3	2	-1	0	0	3565.0	3356.0	2758.0	20878.0	20582.0	19357.0	0
29998	80000.0	41	1	-1	0	0	0	-1	-1645.0	78379.0	76304.0	52774.0	11855.0	48944.0	85900
29999	50000.0	46	0	0	0	0	0	0	47929.0	48905.0	49764.0	36535.0	32428.0	15313.0	2078

30000 rows × 34 columns

```
X.drop(['default.payment.next.month'], axis=1, inplace=True)
```

```
y = df_UCI_Credit_Card[['default.payment.next.month']]
```

```
X.columns = X.columns.astype(str)
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X = scaler.fit_transform(X.astype(float))
```

```
from sklearn.model_selection import train_test_split
import numpy as np

#X = np.array(df_UCI_Credit_Card.drop(['default.payment.next.month'], axis=1))
#y = np.array(df_UCI_Credit_Card[['default.payment.next.month']])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
X_train.shape
```

```
(22500, 33)
```

4. Entrenar y evaluar un modelo de clasificador XG-Boost

```
import xgboost as xgb

model_xgb = xgb.XGBClassifier(learning_rate=0.1, max_depth=20, use_label_encoder=False)
model_xgb.fit(X_train, y_train)
```

```
XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=20, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
results = model_xgb.score(X_test, y_test)
print(f'Precision: {results}')
```

```
Precision: 0.8117333333333333
```

```
y_predict = model_xgb.predict(X_test)
y_predict
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

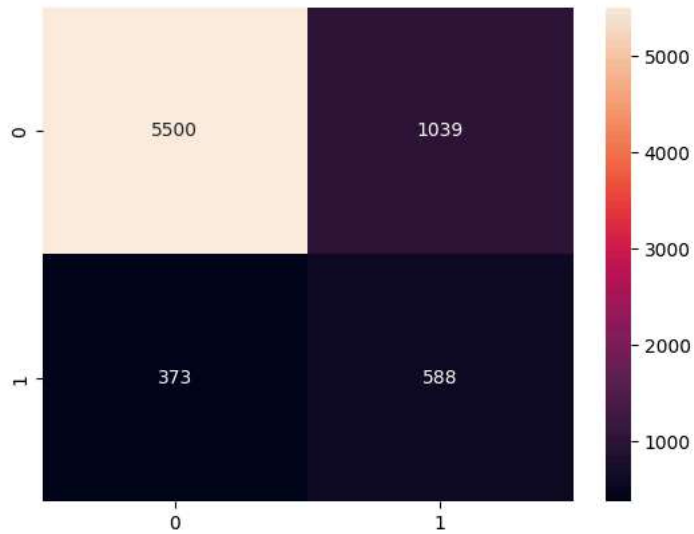
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, confusion_matrix, classification_report, accuracy_score

RMSE = round(float(np.sqrt(mean_squared_error(y_test, y_predict))), 3)
MSE = mean_squared_error(y_test, y_predict)
MAE = mean_absolute_error(y_test, y_predict)
r2 = r2_score(y_test, y_predict)

print(f'RMSE: {RMSE}\nMSE: {MSE}\nMAE: {MAE}\nr2: {r2}')
```

```
RMSE: 0.434
MSE: 0.18826666666666667
MAE: 0.18826666666666667
r2: -0.10827721916815136
```

```
print(f'Accuracy: {100*accuracy_score(y_predict, y_test)} %')
```

```
Accuracy: 81.17333333333333 %
```

```
sns.heatmap(confusion_matrix(y_predict, y_test), annot=True, fmt='d');
```




```
1 print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      5873
           1       0.61      0.36      0.45      1627

    accuracy                           0.81      7500
   macro avg       0.73      0.65      0.67      7500
weighted avg       0.79      0.81      0.79      7500
```

5. Entrenar y evaluar un modelo de regresion logistica

```
1 from sklearn.linear_model import LogisticRegression
2
3 model_LR = LogisticRegression(max_iter=100000)
4 model_LR.fit(X_train, y_train)
```

```
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1310: DataConversionWarning: A colu
  y = column_or_1d(y, warn=True)
```

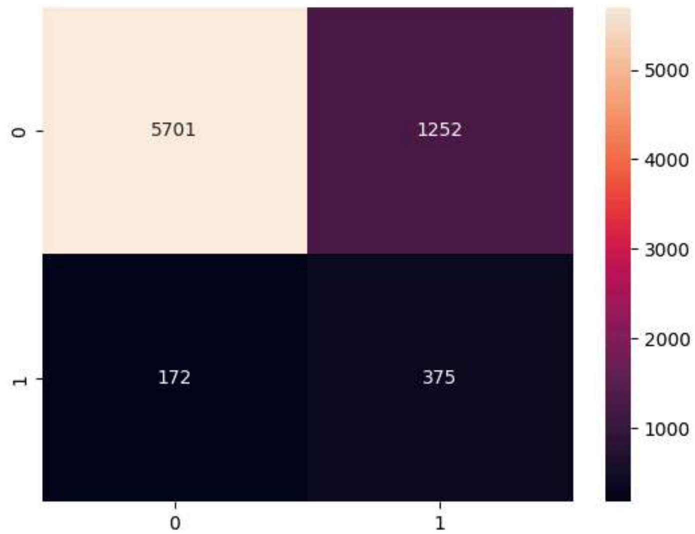
```
▾ LogisticRegression
LogisticRegression(max_iter=100000)
```

```
1 y_predict = model_LR.predict(X_test)
```

```
1 print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.97      0.89      5873
           1       0.69      0.23      0.34      1627

    accuracy                           0.81      7500
   macro avg       0.75      0.60      0.62      7500
weighted avg       0.79      0.81      0.77      7500
```

```
1 sns.heatmap(confusion_matrix(y_predict, y_test), annot=True, fmt='d');
```



6. Entrenar y evaluar un modelo de maquinas de soporte vectorial

```
from sklearn.calibration import CalibratedClassifierCV
from sklearn.svm import LinearSVC

model_SVC = LinearSVC(max_iter=100000)
model_SVM = CalibratedClassifierCV(model_SVC)
model_SVM.fit(X_train, y_train)
```

```
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\preprocessing\_label.py:97: DataConversionWarning: A cc
  y = column_or_1d(y, warn=True)
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1310: DataConversionWarning: A colu
  y = column_or_1d(y, warn=True)
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1310: DataConversionWarning: A colu
  y = column_or_1d(y, warn=True)
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1310: DataConversionWarning: A colu
  y = column_or_1d(y, warn=True)
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1310: DataConversionWarning: A colu
  y = column_or_1d(y, warn=True)
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1310: DataConversionWarning: A colu
  y = column_or_1d(y, warn=True)
```

CalibratedClassifierCV i ?

estimator: LinearSVC

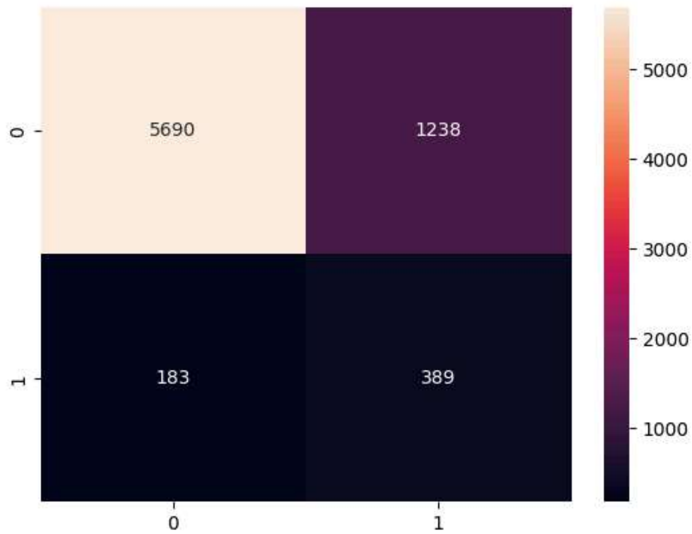
LinearSVC ?

```
y_predict = model_SVM.predict(X_test)
```

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.97      0.89      5873
           1       0.68      0.24      0.35      1627

    accuracy                           0.81      7500
   macro avg       0.75      0.60      0.62      7500
weighted avg       0.79      0.81      0.77      7500
```

```
sns.heatmap(confusion_matrix(y_predict, y_test), annot=True, fmt='d');
```



7. Entrenar y evaluar un bosque aleatorio

```
from sklearn.ensemble import RandomForestClassifier

model_rf = RandomForestClassifier()
model_rf.fit(X_train, y_train)
```

```
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:1473: DataConversionWarning: A column-vector y
  return fit_method(estimator, *args, **kwargs)
```

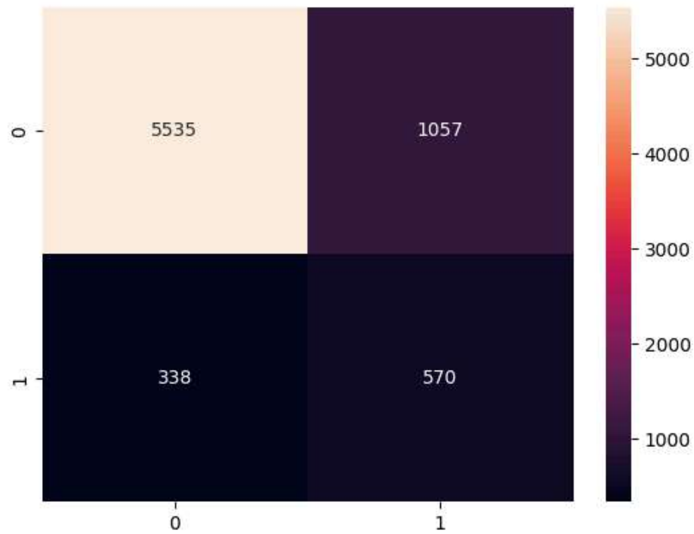
```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y_predict = model_rf.predict(X_test)
```

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      5873
           1       0.63      0.35      0.45      1627

    accuracy                           0.81      7500
   macro avg       0.73      0.65      0.67      7500
weighted avg       0.79      0.81      0.79      7500
```

```
sns.heatmap(confusion_matrix(y_predict, y_test), annot=True, fmt='d');
```



8. Entrenar y evaluar un modelo de KNN

```
from sklearn.neighbors import KNeighborsClassifier

model_knn = KNeighborsClassifier()
model_knn.fit(X_train, y_train)
```

```
c:\Users\sebas\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\neighbors\_classification.py:238: DataConversionWarning
  return self._fit(X, y)
```

▾ KNeighborsClassifier

KNeighborsClassifier()

```
y_predict = model_knn.predict(X_test)
```

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.83      0.92      0.88      5873
           1       0.54      0.34      0.42      1627

    accuracy                           0.80      7500
   macro avg       0.69      0.63      0.65      7500
weighted avg       0.77      0.80      0.78      7500
```

```
sns.heatmap(confusion_matrix(y_predict, y_test), annot=True, fmt='d');
```