
Santander Customer Transaction Prediction

Breitman, Zarina Madelaine
Navarro Quantín, Denise

Marinella, Santiago
Ramos, Mateo

Fecha de entrega: 27.11.2022



Tabla de contenidos

1. Orígen del dataset y objetivo
2. EDA (Exploratory Data Analysis)
3. Modelos
4. Optimización de modelos
5. Ensamble de modelos
6. Conclusiones

1. Dataset y Objetivo

Origen del dataset: Kaggle

<https://www.kaggle.com/datasets/lakshmi25npathi/santander-customer-transaction-prediction-dataset>

Objetivo:

Predecir si se realizará una transacción determinada. Se eligió este dataset por recomendación del profesor debido a que tenía gran cantidad de entradas y es un dataset real.

2. EDA

Análisis del dataset

```
4 df = pd.read_csv('train.csv', index_col="ID_code")
5 df.head()
```

executed in 5.67s, finished 11:16:31 2022-10-28

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7
ID_code									
train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266
train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338
train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155
train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250
train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514

Variables con nombres desconocidos

```
1 df.shape
```

(200000, 201)

200.000 filas y 200 columnas

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 200000 entries, train_0 to train_199999
Columns: 201 entries, target to var_199
dtypes: float64(200), int64(1)
memory usage: 308.2+ MB
```

Todas variables numéricas

2. EDA

Variable a predecir: **Target**

	Cantidad	Porcentaje
0	179902	90.0%
1	20098	10.0%

Variable target muy desbalanceada

Las otras variables:

Dataset limpio

```
1 J = df["target"].isna().sum()  
2 print("Total null values in target column is " + str(J))
```

executed in 13ms, finished 11:17:12 2022-10-28

Total null values in target column is 0

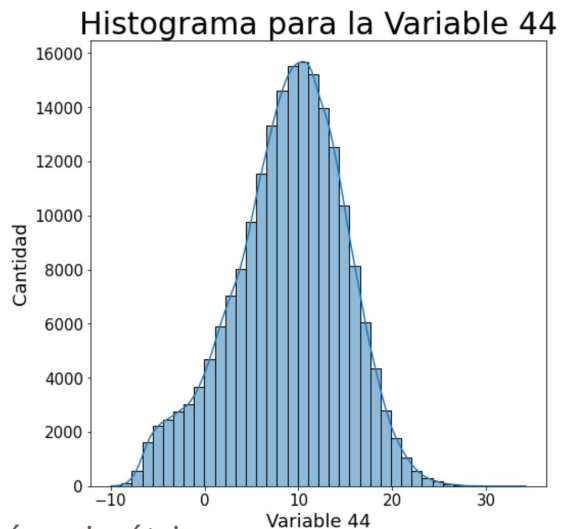
```
1 df.isnull().any().value_counts()
```

False 201
dtype: int64

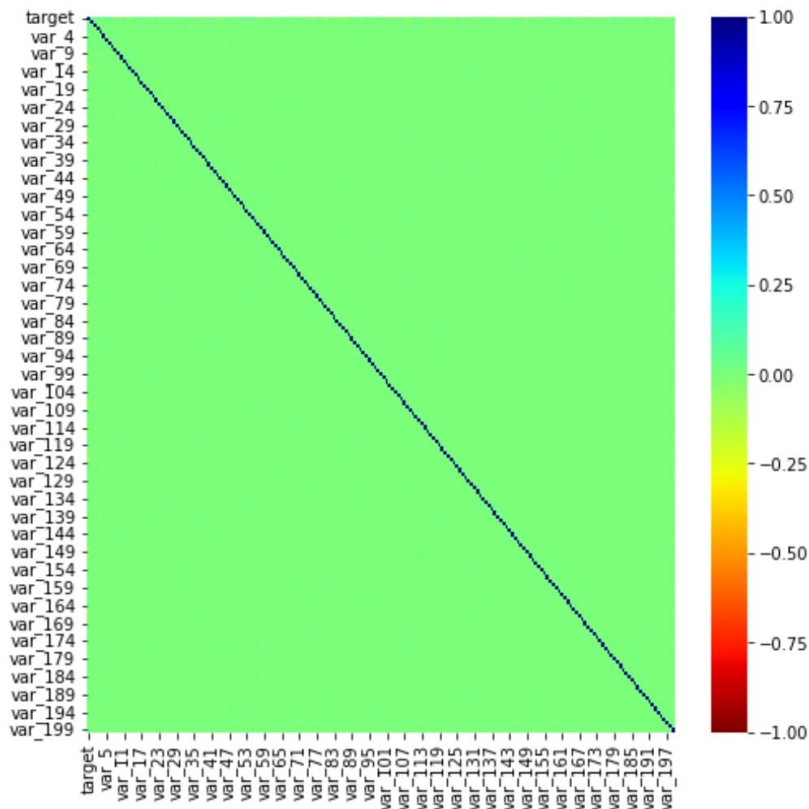
2. EDA

Las otras variables:

- Nula correlación entre variables
- Variables bastante simétricas



Variable más asimétrica



3. Modelos (Clasificación)

No boosting:

- LogisticRegression
- Naïve Bayes Classifier
- MLPClassifier (redes neuronales)
- KNC
- SVC

Boosting:

- Randomforest
- Lightboost
- Adaboost
- Catboost
- XGB

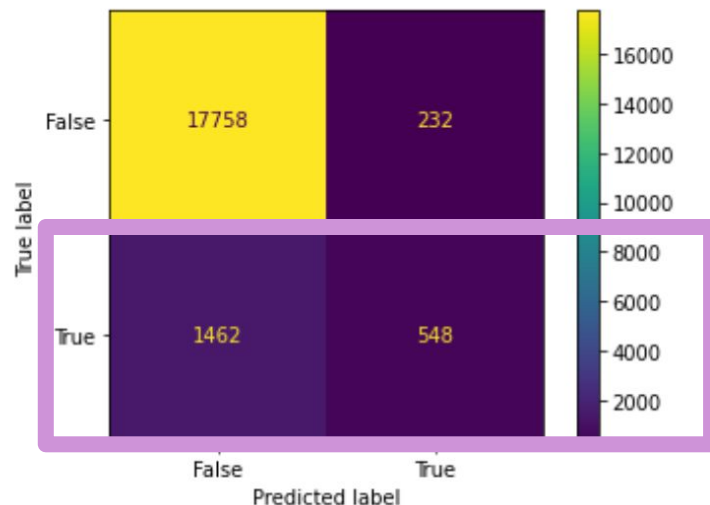
3. Modelos (Métricas)

- Matriz de confusión
- Precision
- **Recall**(dataset desbalanceado)
- F1

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Logistic Regression (all dataset)



Precision score is 0.7025641025641025

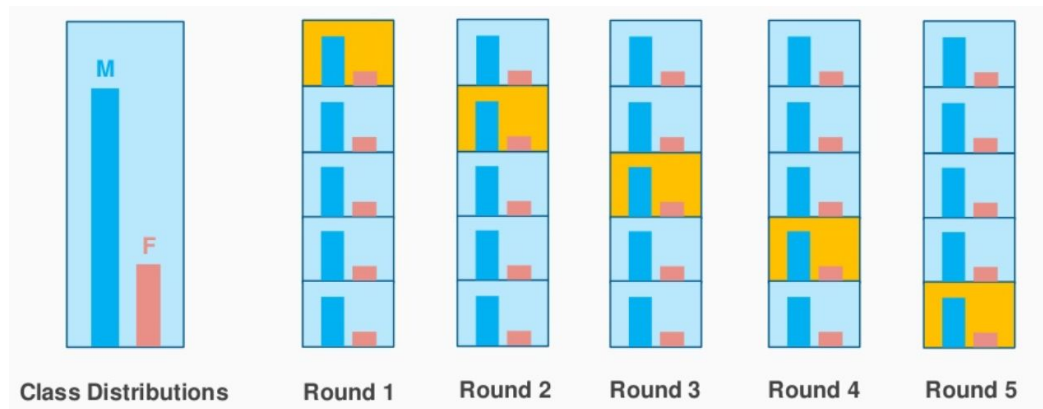
Recall score is 0.272636815920398

F1 score is 0.392831541218638

3. Modelos

Metodología de trabajo

- Stratified K-fold
- PCA
 - 100 componentes
 - 125 componentes
 - 150 componentes
 - 175 componentes



```
skf = StratifiedKFold(n_splits=10, shuffle= True, random_state= 1)
```

3. Modelos

Se obtuvo la media geométrica de cada modelo, promediando los resultados del Kfold.

Se evaluó el costo de cada algoritmo tomando el tiempo de ejecución total.

	RECALL	Tiempo de ejecucion
Fulldata_Catboost	0.334661	501.352978
Fulldata_SVM	0.279937	10509.334164
Fulldata_LogisticRegression	0.268982	802.499365
Fulldata_XGBClassifier	0.249926	1023.306308
Catboost_170	0.242612	440.855039
LogisticRegression_185	0.232312	46.803771
LogisticRegression_170	0.209424	32.181545
XGBClassifier_170	0.189372	1040.673195
Catboost_130	0.187083	381.142504
XGBClassifier_130	0.159021	820.077986
SVM_170	0.149317	9863.222502
LogisticRegression_130	0.143547	29.909196
SVM_130	0.062743	9207.078305

4. Optimización de modelos



O P T U N A

En modelos más costosos se optó por 10 a 20 trials
En modelos más livianos 150 a 400 trials

Se buscó maximizar el recall

Parámetros principales para boosts:

- **n_estimators**: cantidad de árboles en los bosques [300:3000]
- **max_depth**: profundidad de los árboles [3:21]
- **learning_rate**: tamaño de cada paso [0.25:0.85]

Parámetros para LogisticRegression:

- **max_iter**: número máximo de iteraciones para converger [7000:40000]
- **solver**: algoritmos de optimización del modelo ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
- **penalty**: tipo de regularización del modelo ['l2', 'none'] para solvers ['sag', 'saga', 'lbfgs']

4. Optimización de modelos

Best parameters para Logistic Regression

MODEL	RECALL	OPTIMIZED RECALL	PARAMETERS
LogisticRegression_fulldata	0.27119	-	-
LogisticRegression_100	0.21508	0.23772	solver: 'lbfgs', 'max_iter': 25075
LogisticRegression_125	0.23890	0.25194	'solver': 'lbfgs', 'max_iter': 35976
LogisticRegression_150	0.25063	0.26068	'solver': 'lbfgs', 'max_iter': 18956
LogisticRegression_175	0.26778	0.26804	solver: 'sag', 'max_iter': 23980, 'penalty': 'none'

4. Optimización de modelos

Best parameters para otros algoritmos

MODEL	RECALL	OPTIMIZED RECALL	BEST PARAMETERS
Catboost_fulldata	0.334661	0.383969	'iterations': 3600, 'depth': 4, learning_rate': 0.85
Catboost_125	0.18708	0.25029	'iterations': 3600, 'depth': 4, learning_rate': 0.85
MLPClassifier_fulldata	0.35274	-	-
MLPClassifier_175	0.32944	0.36231	'solver': 'lbfgs', 'max_iter': 6250
KNC_fulldata	0.00079	-	-
KNC_175	0.00104	0.04921	'n_neighbors': 2, 'weights': 'distance', 'metric': 'euclidean'
GaussianNB	0.405015	-	No tiene parámetros para optimizar
Lightboost_175	0.05871	0.20875	'max_depth': 13, 'n_estimators': 2430

5. Ensamble de modelos

VotingClassifier:

Se validaron diferentes ensambles de modelos con esta metodología pero el mejor resultado para el recall menor a los modelos sin ensamble y el costo computacional era mayor.

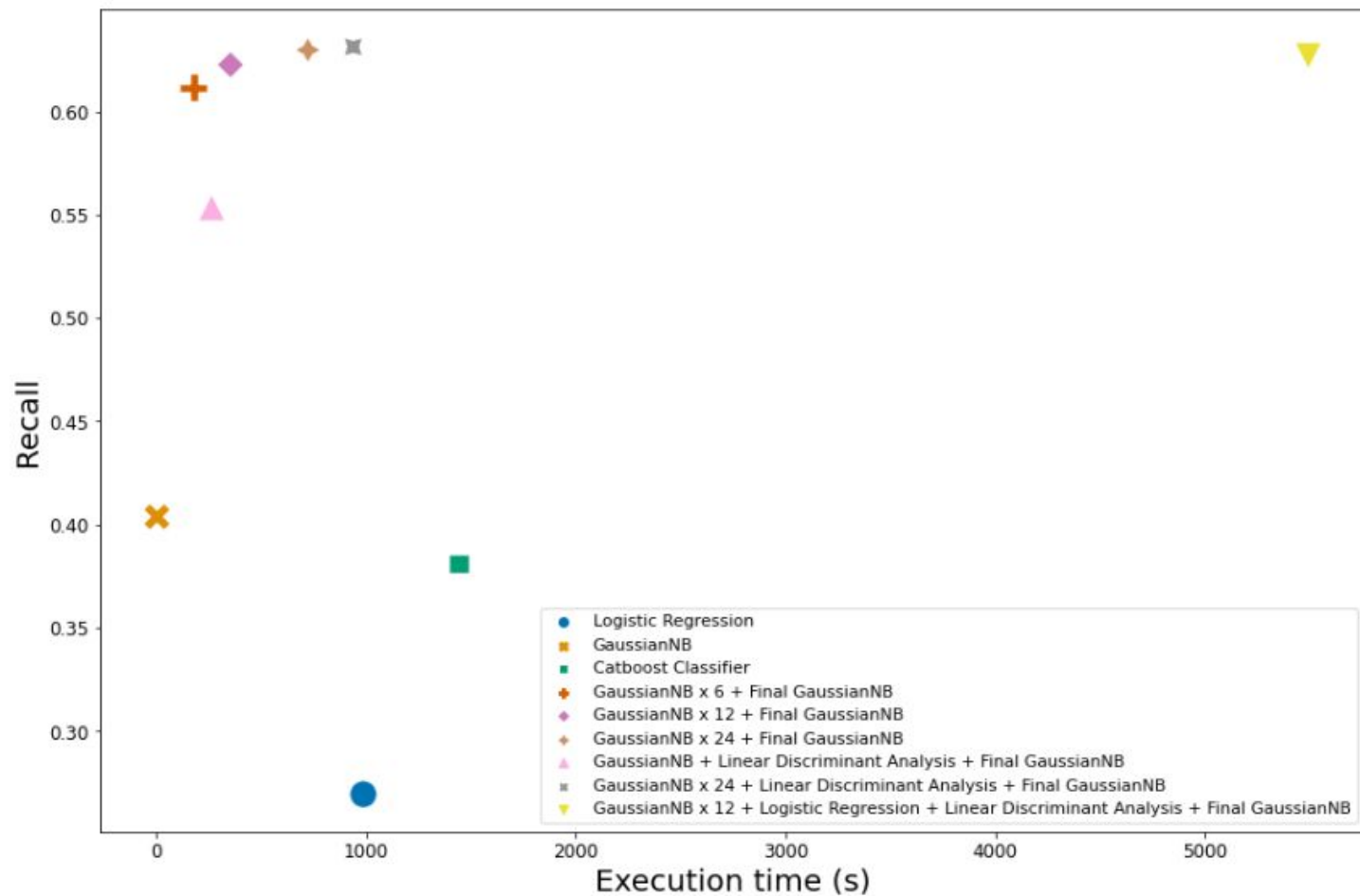


StackingClassifier

Se obtuvieron resultados sustancialmente mejores comparados con los modelos sin ensamblar, para el recall se obtuvo una mejora aproximada de un 23%.



Model Comparison



6. Conclusión

Se puede observar que después de varias interacciones con distintos modelos, la mejor solución es GaussianNB x 24 + Linear Discriminant Analysis + Final Estimator GaussianNB. Con dicho modelo de ensamble se obtuvo un Recall de 0.631196 y un tiempo de ejecución de 940.58 segundos