Detección de fraudes con tarjetas de crédito

Kevin Zapata

Dataset

El dataset tiene 31 columnas, 28 de las cuales contienen información numérica después de un análisis de componentes principales para mantener la información confidencial. La columna de tiempo contiene la cantidad de segundos transcurridos entre la primera fila del dataset y esa fila; la columna de cantidad es la cantidad de dinero en la transacción. La columna de clase será 1 si es un fraude y 0 si no lo es.

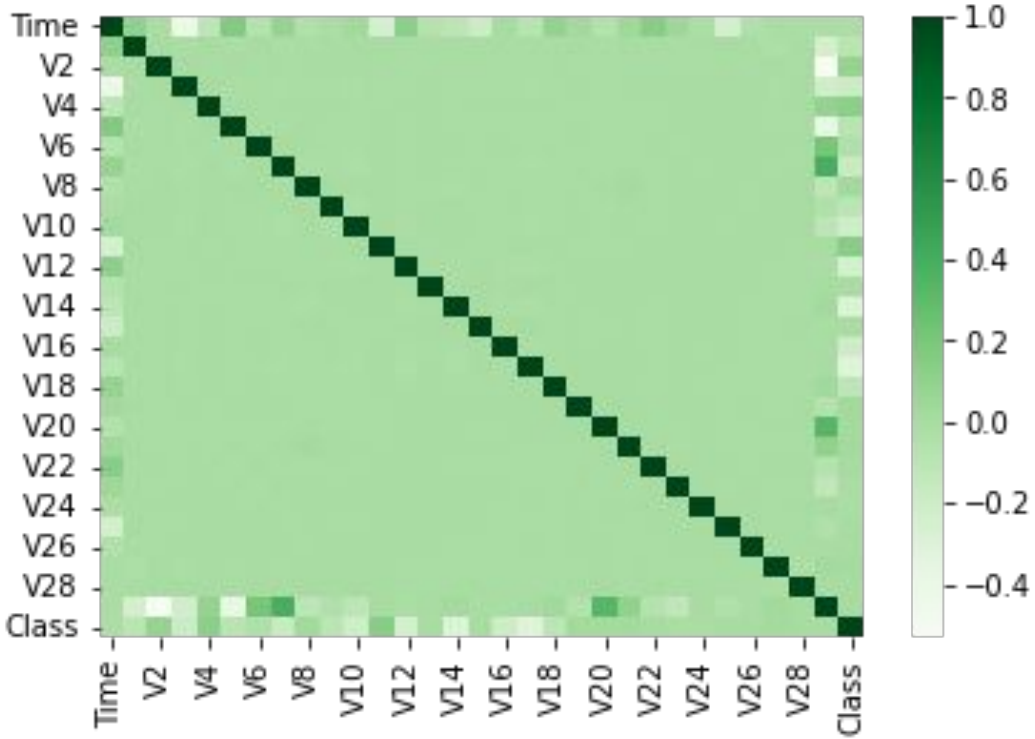
El objetivo es clasificar las transacciones para saber si es un fraude o no. El dataset está muy desbalanceado, donde solo cerca del 2% son fraudes, el reto es construir un modelo de clasificación que este bien a pesar de los pocos fraudes.

Los datos se pueden encontrar en:https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

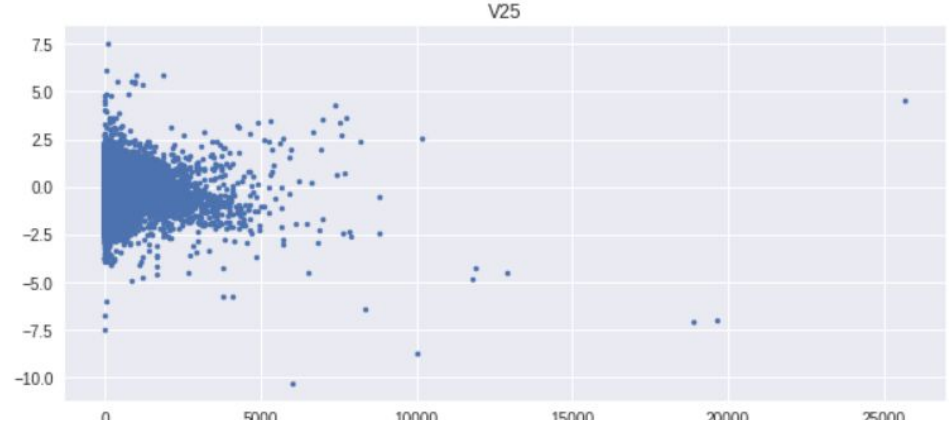
Limpieza

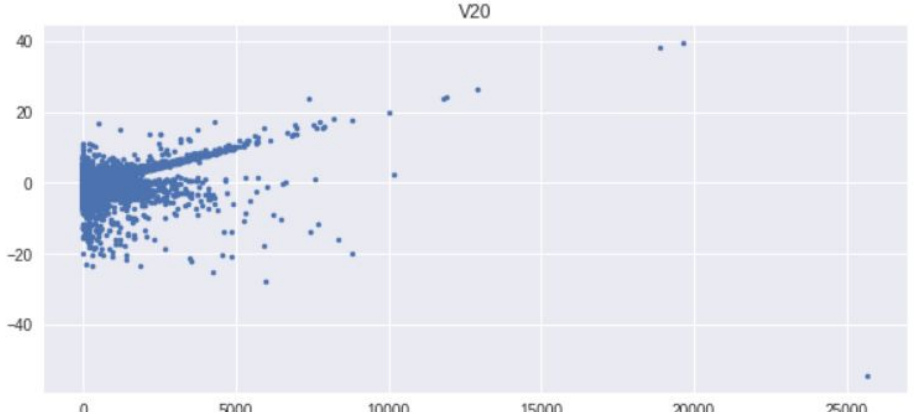
El dataset inicial tiene 284807 filas, de las cuales sólo aproximadamente el 2% son fraudes. Como el dataset ya había pasado por un análisis de componentes principales, todas sus columnas eran numéricas y no tenía ninguna entrada nula; más tenía 1081 filas duplicadas, las cuales decidí eliminar. Las entradas parecían estar bien por lo que la limpieza fue relativamente fácil.

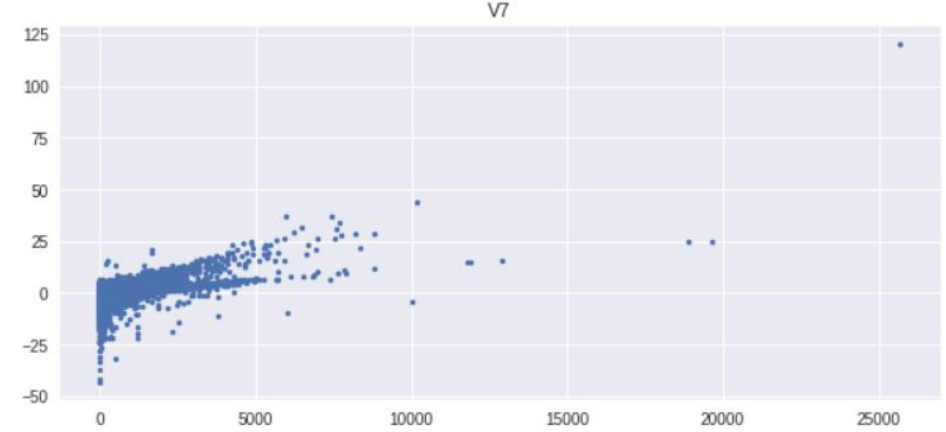
Análisis de correlaciones

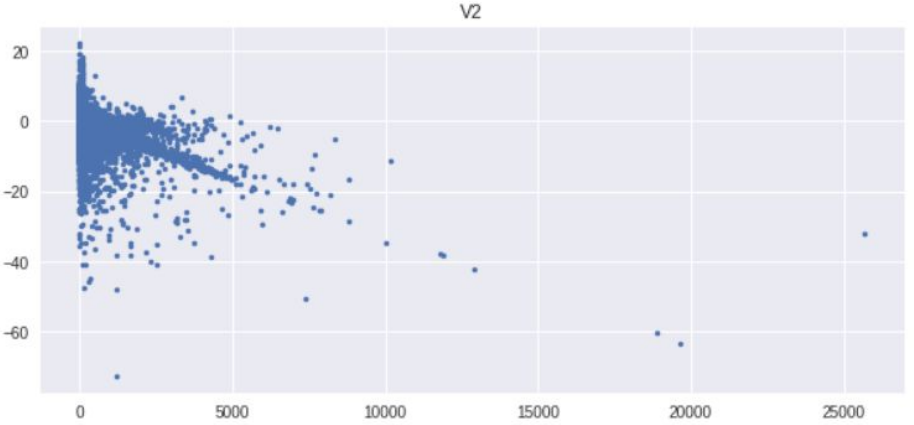


El anterior era el mapa de correlaciones, donde se observa una correlación bastante débil entre las columnas. Las más fuertes serían entre V2 y Amount y entre Amount y V5 (correlación negativa); entre V7 y Amount y entre V20 y Amount. También se ven ciertas correlaciones con la columna de Class. Las gráficas las muestro a continuación, llegando a la conclusión de que Las correlaciones son muy débiles en general, y como se ven en los gráficos, los puntos demasiado dispersos para tener una idea muy clara de que es lo que esta pasando aqui. Como tampoco tenemos mucha información sobre las columnas V1 hasta la V28 estas correlaciones serán muy difíciles de interpretar.


V25
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
0
5000
10000
15000
20000
25000


V20
40
20
0
−20
−40
0
5000
10000
15000
20000
25000


V7
125
100
75
50
25
0
−25
−50
0
5000
10000
15000
20000
25000

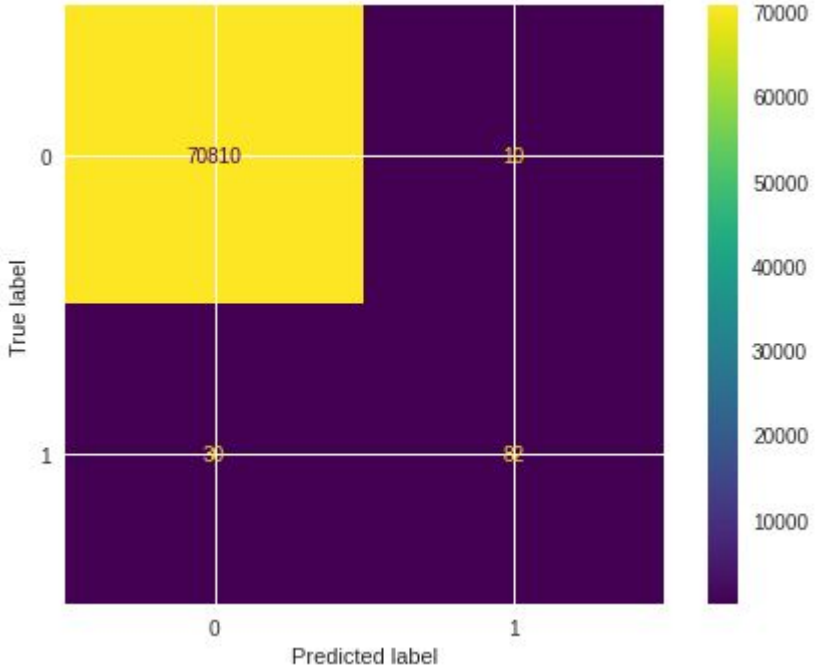

V2
20
0
−20
−40
−60
0
5000
10000
15000
20000
25000

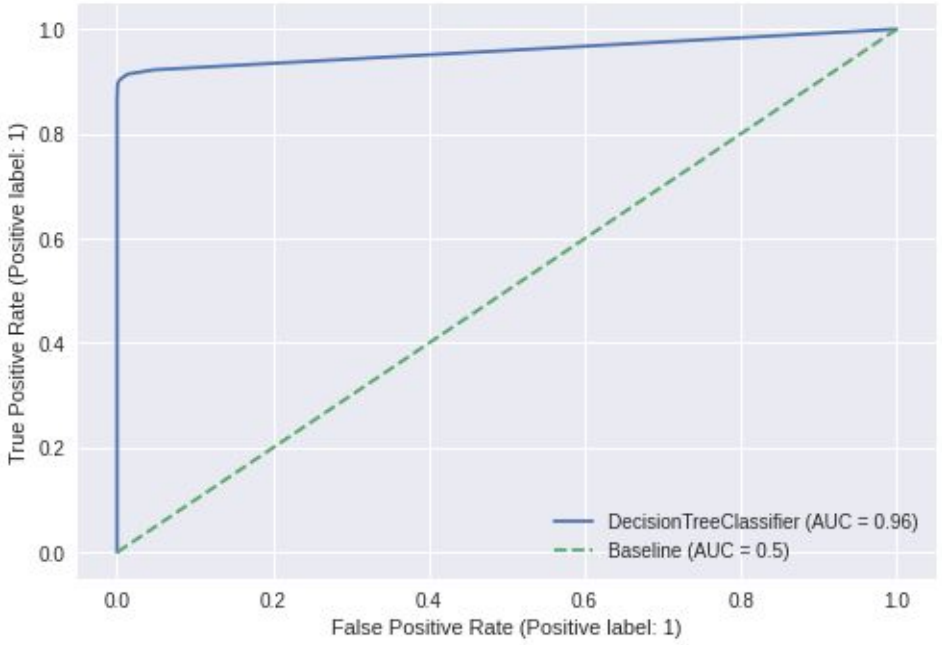
Escalamiento y división

Como todas las columnas son numéricas (debido al PCA) se realiza solo un escalamiento estándar de Pandas. Al realizar la división obtuve:

```
Train data:
Numero total de datos de entrenamiento: 212794
Numero total de fraudes en el entrenamiento: 361
Porcentaje: 0.16964764044098987
Test data:
Numero total de datos de prueba: 70932
Numero total de fraudes en la prueba: 112
Porcentaje: 0.157897704844076
```

Árboles de decisión

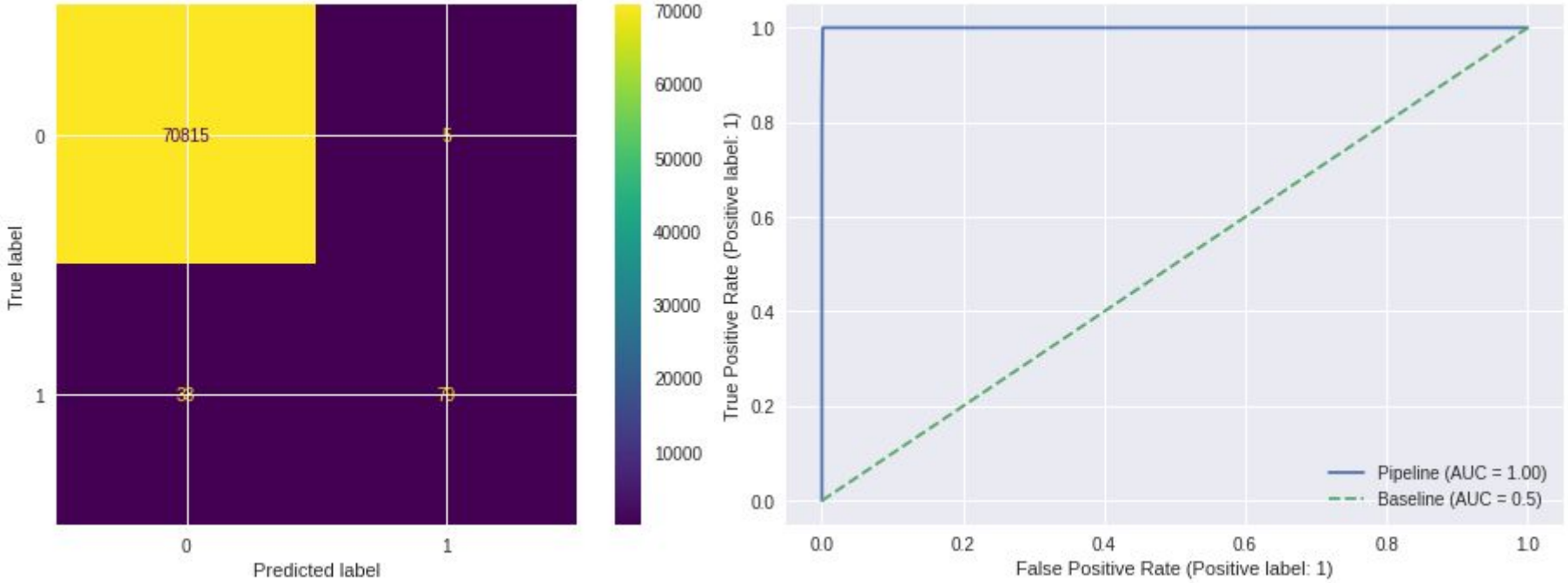




Los resultados de accuracy para este modelo fueron mayores al 99%, pero este resultado hay que analizarlo con cuidado ya que aunque este resultado puede parecer bueno en un comienzo, hay que tener en mente que de nuestros datos de prueba, solo 112 eran fraudes, así que si tenemos más o menos 40 errores (entre falsos positivos y positivos falsos) eso quiere decir que podemos fallar un aproximadamente un 36% de las veces al clasificar una transacción como fraudulenta.

Como era de esperarse los árboles de decisión no son muy buenos en datasets descompensadas.

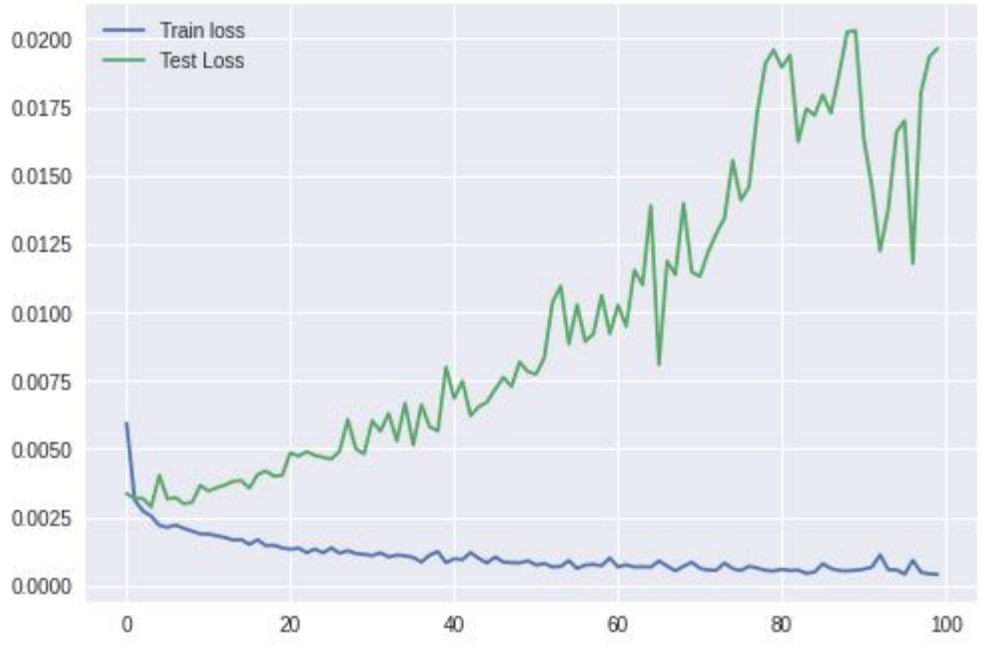
KNN


70815
5
38
79
0
1
True label
0
1
Predicted label
70000
60000
50000
40000
30000
20000
10000
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
Pipeline (AUC = 1.00)
Baseline (AUC = 0.5)

Los resultados de accuracy para este modelo también fueron mayores al 99%, hay que tener en mente que de nuestros datos de prueba, solo 112 eran fraudes, así que si tenemos más o menos 43 errores (entre falsos positivos y positivos falsos) eso quiere decir que podemos fallar un aproximadamente un 37.5% de las veces al clasificar una transacción como fraudulenta.

Redes Neuronales

El primer modelo que probé fue un modelo secuencial con una capa densa de entrada de 30 neuronas; segunda capa densa de 20 neuronas; una tercera capa densa con 30 neuronas; todas con activación tipo relu. También use una salida con bce y adam. En total corrí 100 épocas. Con esto se observa un sobre entrenamiento.

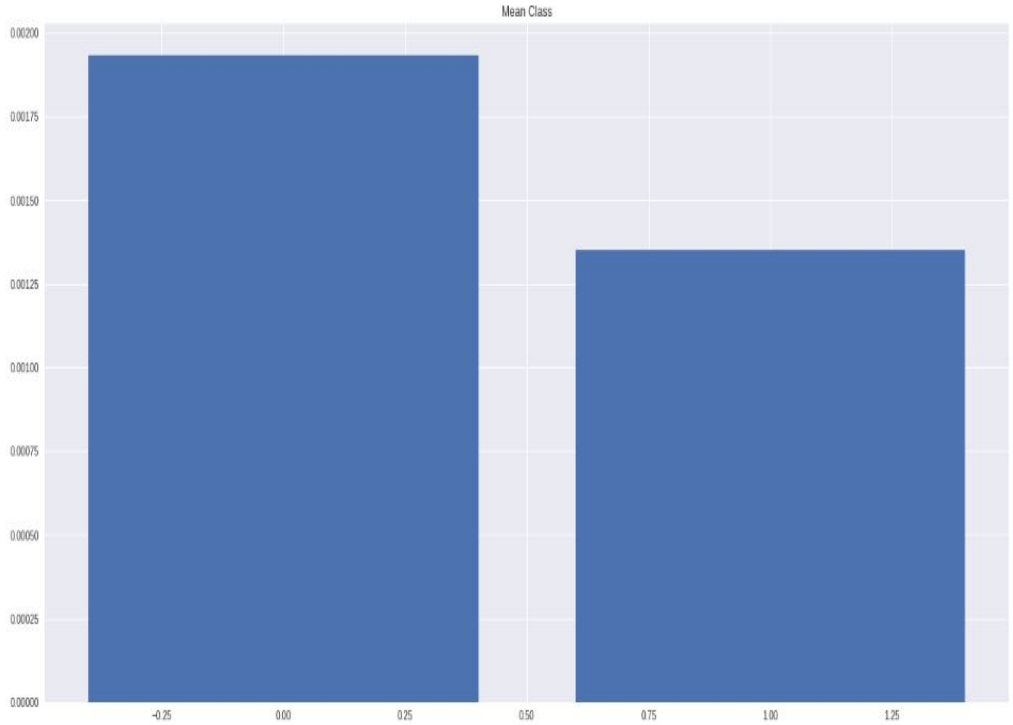


Con un Early Stopping de 10 se mejora el sobreentrenamiento.

K-Means

En este método intente realizar la clasificación con 2 clusters.

Obteniendo un resultado de que solo el 55% de los datos pudieron clasificarse bien.



Conclusión

Los diferentes modelos funcionan bien, pero al ser un dataset tan desbalanceado las interpretaciones pueden llegar a ser confusas y erradas. En el caso del aprendizaje profundo hay que jugar más con los hiperparametros de las redes neuronales y de k-means, en especial este último con un mejor tratamiento (no considerando todas las columnas) debería arrojar un muy buen resultado.