

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE COMPUTACIÓN Y SISTEMAS



PROYECTO FINAL

MainFrame 1

INTEGRANTES:

Azabache Medina, Jean Pierre

Patiño Hermoza, Gustavo Ze Carlos

DOCENTE:

Cueva Chavez, Walter Manuel

Trujillo, 2020

Descripción del caso de estudio

En este proyecto, analizamos un conjunto de datos de transacciones con tarjetas de crédito realizadas durante un período de dos días, nuestro conjunto de datos contiene 284.807 transacciones, de las cuales 492 la cual sería el 0,17% son fraudulentas.

Cada transacción tiene 30 características, todas ellas numéricas. Las características V1 a V 28 son el resultado de una transformación PCA, para poder proteger la confidencialidad por ello la información básica sobre estas funciones no está disponible. La función Tiempo contiene el tiempo transcurrido desde la primera transacción, y la función Monto contiene el monto de la transacción. La variable de respuesta Clase, es 1 en el caso de fraude y 0 en caso contrario.

El objetivo de este proyecto es construir modelos para predecir si una transacción con tarjeta de crédito es fraudulenta. Intentaremos un enfoque de aprendizaje supervisado. También crearemos visualizaciones para ayudarnos a comprender la estructura de los datos y descubrir patrones interesantes.

Procedimiento

Lograr el procedimiento de identificación de un fraude electrónico, primero se realiza la preparación de los datos (muestreo, normalización e imputación) que luego ingresan en el modelo de aprendizaje automático, la cual analizará los datos etiquetados que previamente fueron muestreados para su entrenamiento, segundo creará un modelo de detección basado en su aprendizaje, tercero ingresan los datos de validación, para la comprobación de sus predicciones con los que ya fueron identificados en el dataset.

1. Infraestructura y Plataforma de Información

2. Ciclo de Vida de Ciencia de los Datos

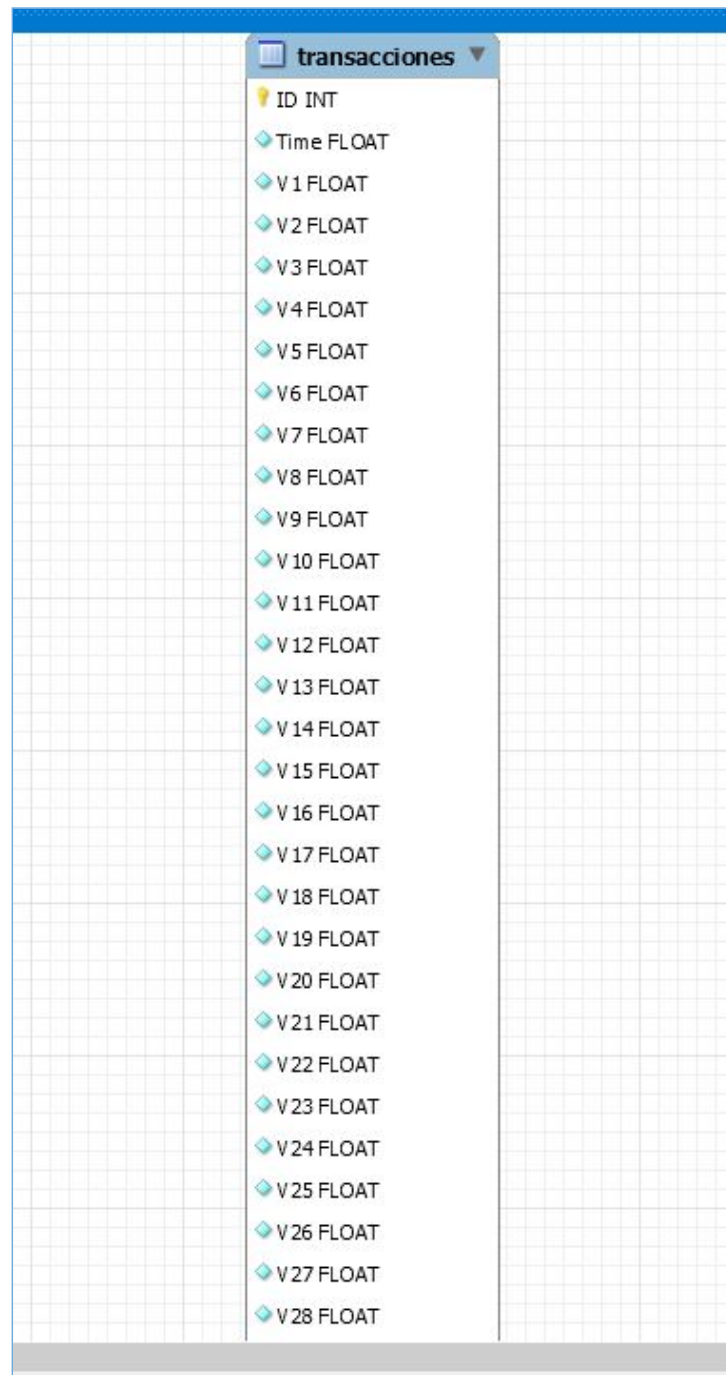
2.1. Recolección de los Datos

Leer los datos usando la librería de Pandas

```
: 1 transacciones = pd.read_csv('creditcard.csv')
```

2.2. Modelo de datos estructurados

2.2.1 Diseño de Tabla en MySql



transacciones	
ID	INT
Time	FLOAT
V 1	FLOAT
V 2	FLOAT
V 3	FLOAT
V 4	FLOAT
V 5	FLOAT
V 6	FLOAT
V 7	FLOAT
V 8	FLOAT
V 9	FLOAT
V 10	FLOAT
V 11	FLOAT
V 12	FLOAT
V 13	FLOAT
V 14	FLOAT
V 15	FLOAT
V 16	FLOAT
V 17	FLOAT
V 18	FLOAT
V 19	FLOAT
V 20	FLOAT
V 21	FLOAT
V 22	FLOAT
V 23	FLOAT
V 24	FLOAT
V 25	FLOAT
V 26	FLOAT
V 27	FLOAT
V 28	FLOAT

2.2.2 Creación de Tabla en MySql

In [12]: %%sql

```
CREATE TABLE `credicart-1`.`transacciones` (  
  `ID` INT NOT NULL AUTO_INCREMENT,  
  `Time` FLOAT NOT NULL,  
  `V1` FLOAT NOT NULL,  
  `V2` FLOAT NOT NULL,  
  `V3` FLOAT NOT NULL,  
  `V4` FLOAT NOT NULL,  
  `V5` FLOAT NOT NULL,  
  `V6` FLOAT NOT NULL,  
  `V7` FLOAT NOT NULL,  
  `V8` FLOAT NOT NULL,  
  `V9` FLOAT NOT NULL,  
  `V10` FLOAT NOT NULL,  
  `V11` FLOAT NOT NULL,  
  `V12` FLOAT NOT NULL,  
  `V13` FLOAT NOT NULL,  
  `V14` FLOAT NOT NULL,  
  `V15` FLOAT NOT NULL,  
  `V16` FLOAT NOT NULL,  
  `V17` FLOAT NOT NULL,  
  `V18` FLOAT NOT NULL,  
  `V19` FLOAT NOT NULL,  
  `V20` FLOAT NOT NULL,  
  `V21` FLOAT NOT NULL,  
  `V22` FLOAT NOT NULL,  
  `V23` FLOAT NOT NULL,  
  `V24` FLOAT NOT NULL,  
  `V25` FLOAT NOT NULL,  
  `V26` FLOAT NOT NULL,  
  `V27` FLOAT NOT NULL,  
  `V28` FLOAT NOT NULL,  
  `Amount` FLOAT NOT NULL,  
  `Class` INT NOT NULL,  
  PRIMARY KEY (`ID`));
```


```
* mysql+mysqlconnector://root:***@localhost:3306/credicart-1  
0 rows affected.
```

Out[12]: []

2.2.3 Población de Tabla en MySQL

Table Data Import

Configure Import Settings

Detected file format: csv 

Encoding:

utf-8

Columns:

<input checked="" type="checkbox"/> Source Column	Dest Column
<input checked="" type="checkbox"/> MyUnknownColumn	ID <div></div>
<input checked="" type="checkbox"/> Time	Time <div></div>
<input checked="" type="checkbox"/> V1	V1 <div></div>
<input checked="" type="checkbox"/> V2	V2 <div></div>
<input checked="" type="checkbox"/> V3	V3 <div></div>
<input checked="" type="checkbox"/> V4	V4 <div></div>
<input checked="" type="checkbox"/> V5	V5 <div></div>
<input checked="" type="checkbox"/> V6	V6 <div></div>

MyUnknown...	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	
0	0.0	-1.3598071...	-0.0727811...	2.53634673...	1.37815522...	-0.3383207...	0.46238777...	0.23959855...	0.09869790...	0.36378696...	0.09079417...	-0.5515995...	-0.6178008...	
1	0.0	1.19185711...	0.26615071...	0.16648011...	0.44815407...	0.06001764...	-0.0823608...	-0.0788029...	0.08510165...	-0.2554251...	-0.1669744...	1.61272666...	1.06523531...	
2	1.0	-1.3583540...	-1.3401630...	1.77320934...	0.37977959...	-0.5031981...	1.80049938...	0.79146095...	0.24767578...	-1.5146543...	0.20764286...	0.62450145...	0.06608368...	
3	1.0	-0.9662717...	-0.1852260...	1.79299333...	-0.8632912...	-0.0103088...	1.24720316...	0.23760893...	0.37743587...	-1.3870240...	-0.0549519...	-0.2264872...	0.17822822...	

< Back

Next >

Cancel

2.3. Transformación y consultas

Estadísticas Descriptivas

```
1 media=transacciones.Time.mean()  
2 media
```

94813.85957508067

```
1 media=transacciones.Amount.mean()  
2 media
```

88.34961925093133

```
1 mediana=transacciones.Time.median()  
2 mediana
```

84692.0

```
1 mediana=transacciones.Amount.median()  
2 mediana
```

22.0

```
1 q4=transacciones.Time.quantile(0.04)  
2 q4
```

19880.72

```
1 q4=transacciones.Amount.quantile(0.04)
2 q4
```

```
0.89
```

```
1 p1=np.percentile(transacciones.Time, 25)
2 p1
```

```
54201.5
```

```
1 p1=np.percentile(transacciones.Amount, 25)
2 p1
```

```
5.6
```

```
1 maximo=transacciones.Time.max()
2 maximo
```

```
172792.0
```

```
1 maximo=transacciones.Amount.max()
2 maximo
```

```
25691.16
```

```
1 minimo=transacciones.Amount.min()
2 minimo
```

```
0.0
```

```
1 minimo=transacciones.Time.min()
2 minimo
```

```
0.0
```

Transformación

```
1 #transacciones.drop(['Resultados'], axis=1, inplace=True)
2 transacciones.insert(1, 'Resultados', 0)
```

```
1 transacciones.Resultados=transacciones.Class
```

```
1 transacciones.head(5)
```

	Time	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23	V24
0	0.0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	...	-0.018307	0.277838	-0.110474	0.066928
1	0.0	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	...	-0.225775	-0.638672	0.101288	-0.339846
2	1.0	0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	...	0.247998	0.771679	0.909412	-0.689281
3	1.0	0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	...	-0.108300	0.005274	-0.190321	-1.175575
4	2.0	0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	...	-0.009431	0.798278	-0.137458	0.141267

5 rows x 32 columns

```
1 transacciones[['Time','Amount','Resultados']].head(20)
```

	Time	Amount	Resultados
0	0.0	149.62	0
1	0.0	2.69	0
2	1.0	378.66	0
3	1.0	123.50	0
4	2.0	69.99	0
5	2.0	3.67	0

```
1 transacciones['Class'].value_counts()
```

```
0    284315
1      492
Name: Class, dtype: int64
```

```
1 transacciones.sort_values(by=['Amount'], ascending=True)
```

	Time	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22
15816	27255.0	0	1.248804	0.047208	0.423388	-0.139515	-0.592217	-0.980654	-0.042416	-0.123044	...	-0.186215	-0.501598
77470	57062.0	0	-1.188664	-0.612034	2.422204	-0.812786	0.318493	-0.671637	-0.432053	0.068237	...	0.002347	0.164823
190885	129019.0	0	1.868263	0.273764	-0.288023	3.835852	0.268329	0.817380	-0.287993	0.203258	...	0.115927	0.610472
87335	61640.0	0	-0.848470	1.426562	2.137094	2.852036	-0.366945	1.158146	-0.416142	0.812490	...	-0.210710	-0.369433
174481	121931.0	0	-1.184195	0.804518	2.240498	2.853175	1.038068	0.171728	0.457665	0.290123	...	0.110209	0.254591
...
54018	46253.0	0	-21.780665	-38.305310	-12.122409	9.752791	-12.880794	4.256017	14.785051	-2.818253	...	7.437478	-5.619439
46841	42951.0	0	-23.712839	-42.172688	-13.320825	9.925019	-13.945538	5.564891	15.710644	-2.844253	...	7.921600	-6.320710
151296	95288.0	0	-34.549296	-60.464618	-21.340854	16.875344	-19.229075	6.335259	24.422716	-4.964566	...	11.502580	-9.499423
58465	48401.0	0	-36.802320	-63.344698	-20.645794	16.715537	-20.672064	7.694002	24.956587	-4.730111	...	11.455313	-10.933144
274771	166198.0	0	-35.548539	-31.850484	-48.325589	15.304184	-113.743307	73.301626	120.589494	-27.347360	...	-21.620120	5.712303


```
1 transacciones.select_dtypes(include=['int64'])
```

	Resultados	Class
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...
284802	0	0
284803	0	0
284804	0	0
284805	0	0
284806	0	0

284807 rows × 2 columns

```
1 transacciones.sort_values(by=['Time'], ascending=False)
```

	Time	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23
284806	172792.0	0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414850	...	0.261057	0.643078	0.376777
284805	172788.0	0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	...	0.265245	0.800049	-0.163298
284804	172788.0	0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	...	0.232045	0.578229	-0.037501
284803	172787.0	0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	...	0.214205	0.924384	0.012463
284802	172786.0	0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	...	0.213454	0.111864	1.014480
...
5	2.0	0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	...	-0.208254	-0.559825	-0.026398
2	1.0	0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	...	0.247998	0.771679	0.909412
3	1.0	0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237669	0.377436	...	-0.108300	0.005274	-0.190321
1	0.0	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	...	-0.225775	-0.638672	0.101288
0	0.0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	...	-0.018307	0.277838	-0.110474

284807 rows × 32 columns

```
1 transacciones.Amount.describe()
```

```
count      284807.000000
mean         88.349619
std         250.120109
min           0.000000
25%          5.600000
50%         22.000000
75%         77.165000
max        25691.160000
Name: Amount, dtype: float64
```

```
1 normal.Amount.describe()
```

```
count      284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%          5.650000
50%         22.000000
75%         77.050000
max        25691.160000
Name: Amount, dtype: float64
```

Porcentaje de clasificacion sobre el total del DataSet

Porcentaje de clasificacion sobre el total del DataSet

```
: 1 transacciones['Class'].value_counts(normalize=True)
: 0    0.998273
: 1    0.001727
Name: Class, dtype: float64
```

2.3. Preparación de Datos

```
1 transacciones.sample(5)
```

	Time	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	—	V21	V22	V23	
140046	83509.0	0	1.257065	0.674848	-0.303137	0.902041	-0.023612	-1.577300	0.464704	-0.389980	—	-0.106196	-0.207451	-0.049645	0.6411
48973	43858.0	0	-3.312835	2.181919	-0.762963	0.311522	2.215393	4.913325	-1.931987	-3.412496	—	4.616781	-1.752986	0.312910	0.9211
189966	128632.0	0	-0.187254	1.017291	-0.974222	-0.015757	0.170746	-1.111031	1.163252	-0.001150	—	0.440261	1.148669	0.135092	-0.0411
637115	50843.0	0	-1.392478	1.192584	0.286848	-2.959640	2.775996	3.246714	0.974131	-0.072125	—	-0.568914	-0.496575	-0.237096	-1.0111
207144	1366538.0	0	2.026829	-0.235613	-3.122868	0.008242	1.279637	-0.012390	1.048054	-0.535275	—	0.240236	0.579650	-0.361863	-0.2011

5 rows x 32 columns

```
1 transacciones.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
...
284802  False
284803  False
284804  False
284805  False
284806  False
Length: 284807, dtype: bool
```

```
1 eliminar=transacciones.drop_duplicates()
2 eliminar
```

	Time	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23	
0	0.0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462368	0.239599	0.098698	—	-0.018307	0.277838	-0.110474	0.04
1	0.0	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	—	-0.225775	-0.638672	0.101288	-0.3
2	1.0	0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	—	0.247998	0.771679	0.909412	-0.6
3	1.0	0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	—	-0.108300	0.005274	-0.190321	-1.1
4	2.0	0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	—	-0.009431	0.798278	-0.137458	0.1
...
284802	172786.0	0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	—	0.213454	0.111864	1.014480	-0.5
284803	172787.0	0	-0.732789	-0.056080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	—	0.212405	0.924384	0.012463	-1.0
284804	172788.0	0	1.191955	-0.301254	-3.249840	-0.557828	2.630515	3.031260	-0.296827	0.708417	—	0.232045	0.578229	-0.037501	0.6
284805	172788.0	0	-2.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	—	0.265245	0.800049	-0.163298	0.1
284806	172792.0	0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	—	0.261057	0.643078	0.376777	0.0

283726 rows x 32 columns

```
1 set(transacciones["Resultados"])
```

$$I = (0, 1)$$

```
1 transacciones['Resultados'] = transacciones['Resultados'].replace(0, "Normal")
2 transacciones['Resultados'] = transacciones['Resultados'].replace(1, "Fraude")
```

```
1 transacciones.sample(5)
```

	Time	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23	V24
876	664.0	Normal	-2.801435	-0.191528	2.406736	1.221994	0.615015	0.605587	0.217909	-2.551862	...	0.804909	-0.556608	0.057194	-0.117751
25065	33503.0	Normal	-0.365785	0.043265	0.701371	-2.606583	-0.518856	-0.463404	0.305894	-1.319847	...	0.331104	-1.029224	0.166928	-0.047767
20951	31373.0	Normal	1.165221	0.093792	0.112002	0.475063	-0.110208	-0.116998	-0.182743	0.183729	...	-0.238663	-0.793364	0.110363	-0.393744
44771	42077.0	Normal	-0.826625	1.255262	1.306638	0.037673	-0.360009	-0.657371	0.303772	0.468929	...	-0.131625	-0.19818	0.114568	0.383326
90717	63139.0	Normal	-0.492877	-0.741865	2.330206	-2.291968	-1.291543	0.135619	-0.642833	0.174217	...	-0.073501	0.266854	0.054104	0.080202

5 rows x 32 columns

```

1 # Separamos las variables dependientes e independientes
2 x = transacciones.iloc[:, :-1]
3 y = transacciones.iloc[:, 3]

1 from sklearn import preprocessing

1 fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(6,5))
2 ax1.set_title('Antes de escalar')
3 sns.kdeplot(transacciones['V1'],ax=ax1)
4 sns.kdeplot(transacciones['V2'],ax=ax1)
5 sns.kdeplot(transacciones['V3'],ax=ax1)
6 sns.kdeplot(transacciones['V4'],ax=ax1)
7
8 scaler=preprocessing.StandardScaler()
9 transacciones[['Time','Amount']]=scaler.fit_transform(transacciones[['Time','Amount']])
10
11
12 ax1.set_title('Despues de escalar')
13 sns.kdeplot(transacciones['V1'],ax=ax2)
14 sns.kdeplot(transacciones['V2'],ax=ax2)
15 sns.kdeplot(transacciones['V3'],ax=ax2)
16 sns.kdeplot(transacciones['V4'],ax=ax2)
17
18

```

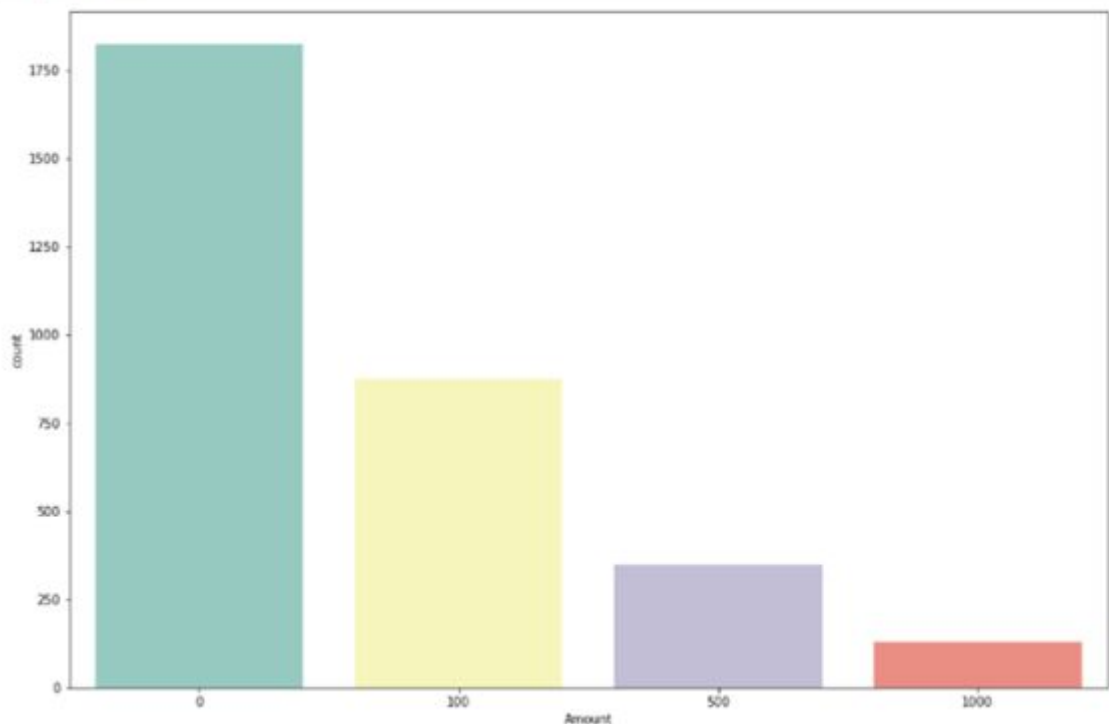
4 AxesSubplot:~

2.4. Exploración Visual de datos

```

1 plt.figure(figsize=(15,10))#canvlas
2 sns.countplot(x="Amount",data=transacciones,palette="Set3",order=[0,100,500, 1000])
3 plt.show()

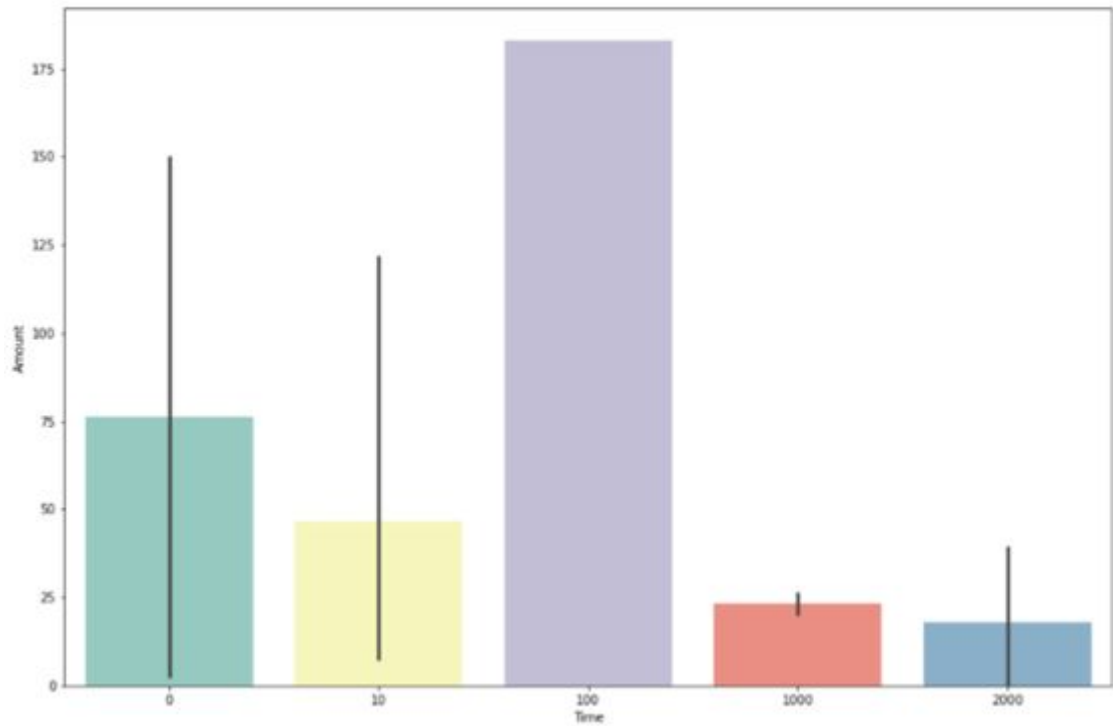
```



```

1 plt.figure(figsize=(15,10))#canvas
2 sns.barplot(x="Time",y="Amount",data=transacciones,palette="Set3",order=[0,10,100,1000,2000])
3 plt.show()

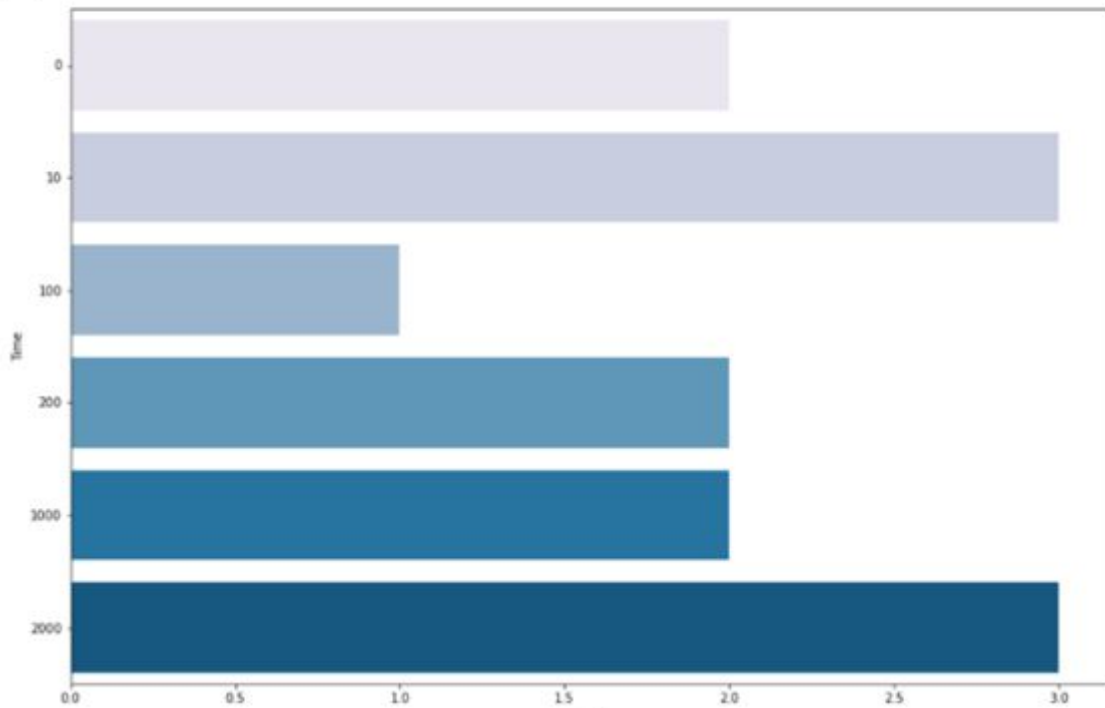
```



```

2 plt.figure(figsize=(15,10))#canvas
3 sns.countplot(y="Time",data=transacciones,palette="PuBu",order=[0,10,100,200,1000,2000])
4 plt.show()

```

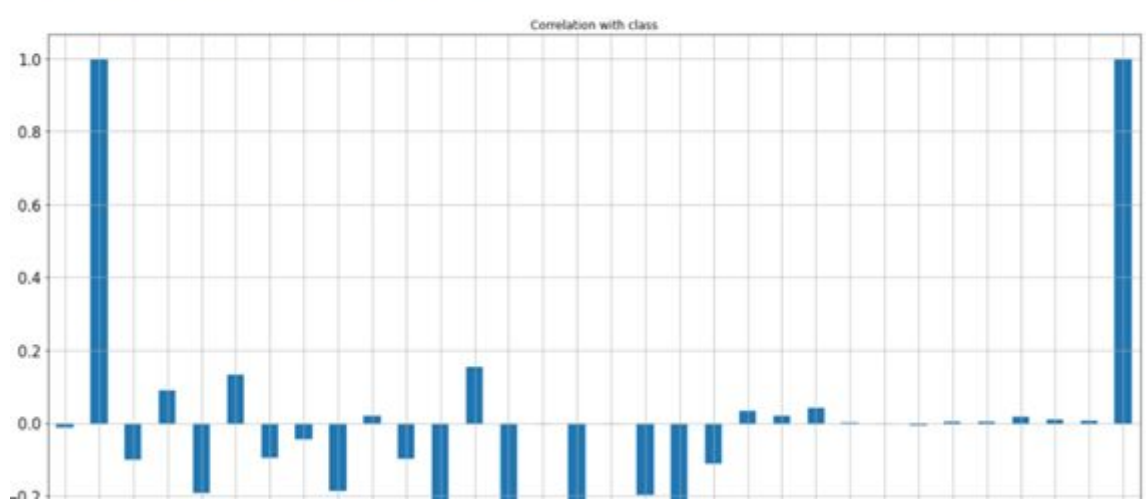


```
1 count_classes = pd.value_counts(transacciones['Class'], sort = True)
2 count_classes.plot(kind = 'bar', rot=0)
3 plt.title("Transaction class distribution")
4 plt.xticks(range(2), LABELS)
5 plt.xlabel("Class")
6 plt.ylabel("Frequency");
```



```
1 transacciones.corrwith(transacciones.Class).plot.bar(
2     figsize = (20, 10), title = "Correlation with class", fontsize = 15,
3     rot = 45, grid = True)
```

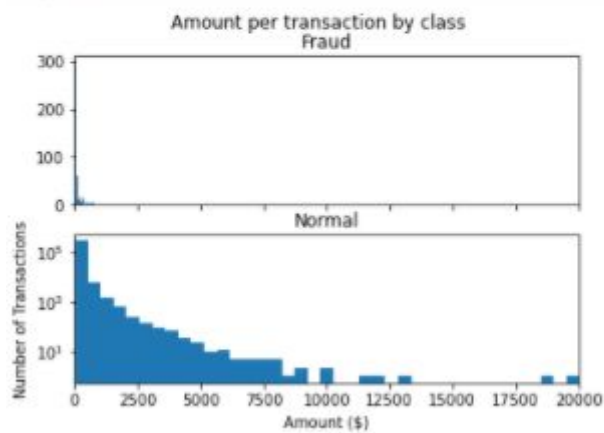
axesSubplot:title={'center':'Correlation with class'}>




```

1 f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
2 f.suptitle('Amount per transaction by class')
3 bins = 50
4 ax1.hist(frauds.Amount, bins = bins)
5 ax1.set_title('Fraud')
6 ax2.hist(normal.Amount, bins = bins)
7 ax2.set_title('Normal')
8 plt.xlabel('Amount ($)')
9 plt.ylabel('Number of Transactions')
10 plt.xlim((0, 20000))
11 plt.yscale('log')
12 plt.show();

```



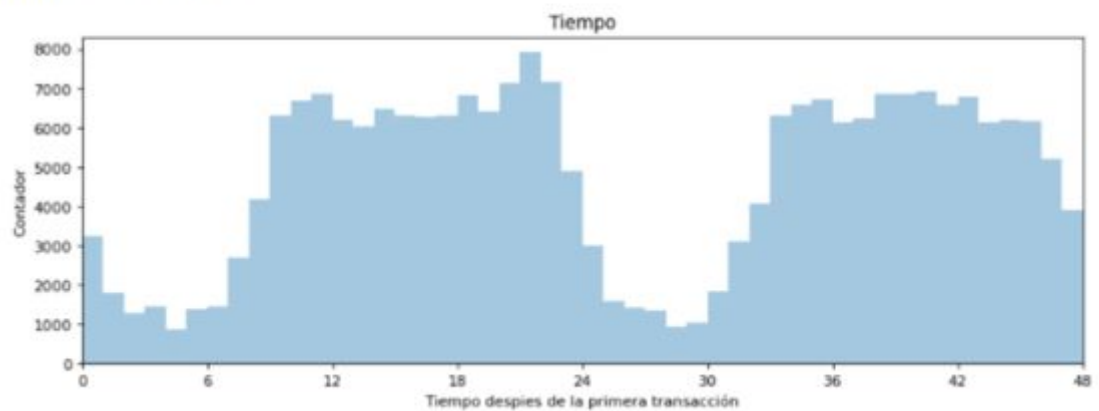
Solo el 0,17% (492 de 284.807) transacciones son fraudulentas

```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.distplot(X_train['Time'], bins=48, kde=False)
3 plt.xlim([0,48])
4 plt.xticks(np.arange(0,54,6))
5 plt.xlabel('Tiempo despies de la primera transacción')
6 plt.ylabel('Contador')
7 plt.title('Tiempo')

```

Text(0.5, 1.0, 'Tiempo')

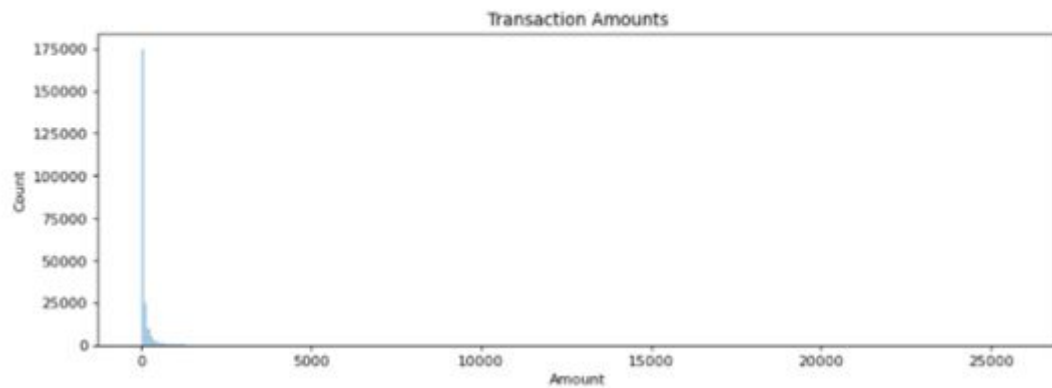


```

]: 1 plt.figure(figsize=(12,4), dpi=80)
   2 sns.distplot(X_train['Amount'], bins=300, kde=False)
   3 plt.ylabel('Count')
   4 plt.title('Transaction Amounts')

```

```
]: Text(0.5, 1.0, 'Transaction Amounts')
```



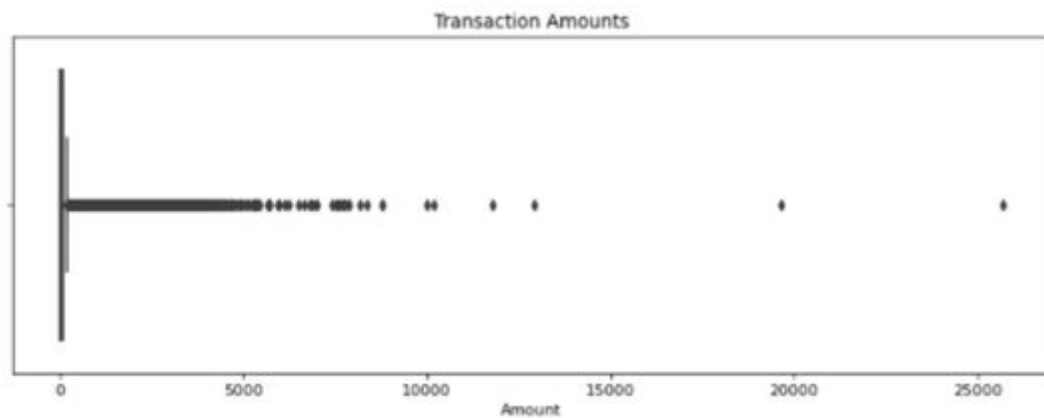
- Diagrama de caja para poder observar los valores atípicos que no se pueden diferenciar en histograma realizado anteriormente

```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.boxplot(X_train['Amount'])
3 plt.title('Transaction Amounts')

```

```
Text(0.5, 1.0, 'Transaction Amounts')
```




```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.distplot(X_train['Amount'], kde=False)
3 plt.xlabel('Transformed Amount')
4 plt.ylabel('Count')
5 plt.title('Monto de Transacciones')

```

Text(0.5, 1.0, 'Monto de Transacciones')

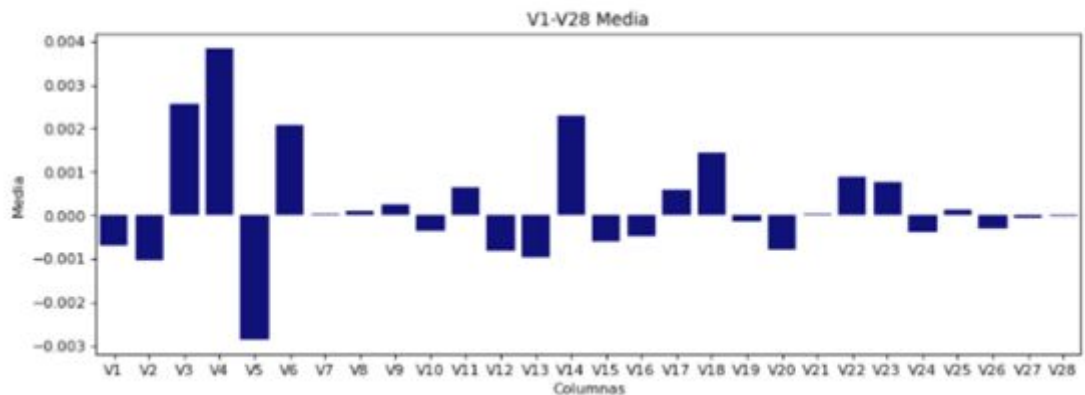


```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.barplot(x=pca_vars, y=X_train[pca_vars].mean(), color='darkblue')
3 plt.xlabel('Columnas')
4 plt.ylabel('Media')
5 plt.title('V1-V28 Media')

```

Text(0.5, 1.0, 'V1-V28 Media')



```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.barplot(x=pca_vars, y=X_train[pca_vars].std(), color='darkred')
3 plt.xlabel('Columnas')
4 plt.ylabel('Desviacion Estandar')
5 plt.title('V1-V28 Desviacion Estandar')

```

Text(0.5, 1.0, 'V1-V28 Desviacion Estandar')



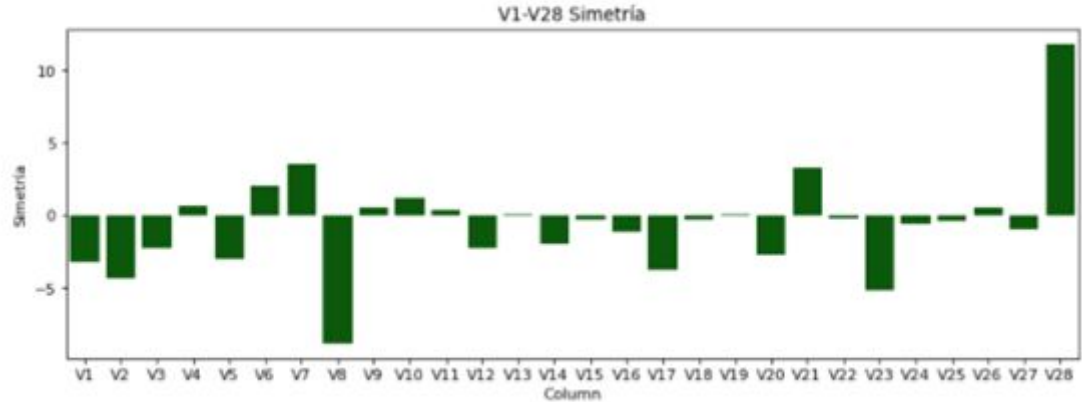
The PCA variables have roughly unit variance, but as low as ~0.3 and as high as ~1.9. Plot the skewnesses next:

```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.barplot(x=pca_vars, y=X_train[pca_vars].skew(), color='darkgreen')
3 plt.xlabel('Column')
4 plt.ylabel('Simetría')
5 plt.title('V1-V28 Simetría')

```

Text(0.5, 1.0, 'V1-V28 Simetría')

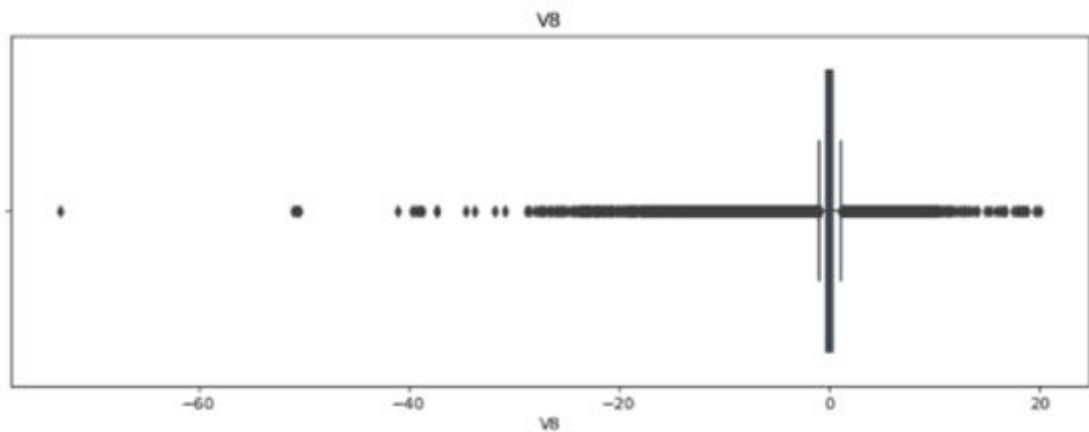


```

1 plt.figure(figsize=(12,4), dpi=80)
2 sns.boxplot(X_train['V8'])
3 plt.title('V8')

```

Text(0.5, 1.0, 'V8')

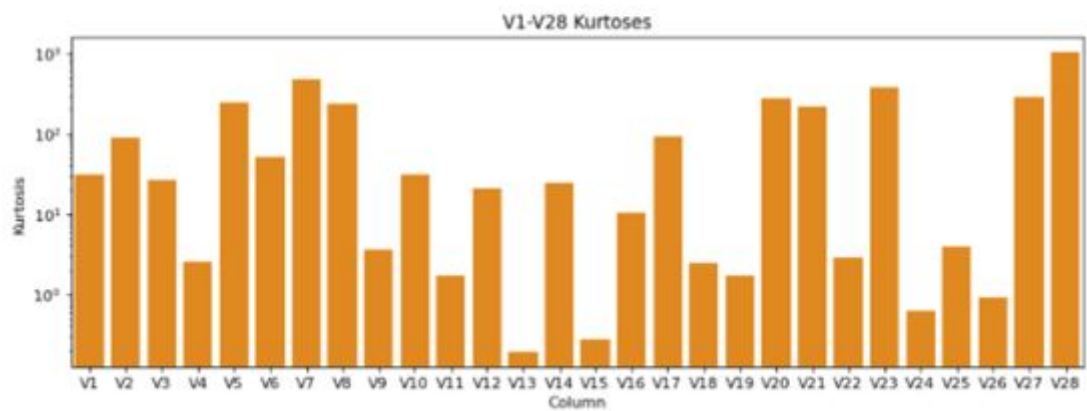


```

1 plt.figure(figsize=(12,4), dpi=80)
2 plt.yscale('log')
3 sns.barplot(x=pca_vars, y=X_train[pca_vars].kurtosis(), color='darkorange')
4 plt.xlabel('Column')
5 plt.ylabel('Kurtosis')
6 plt.title('V1-V28 Kurtoses')

```

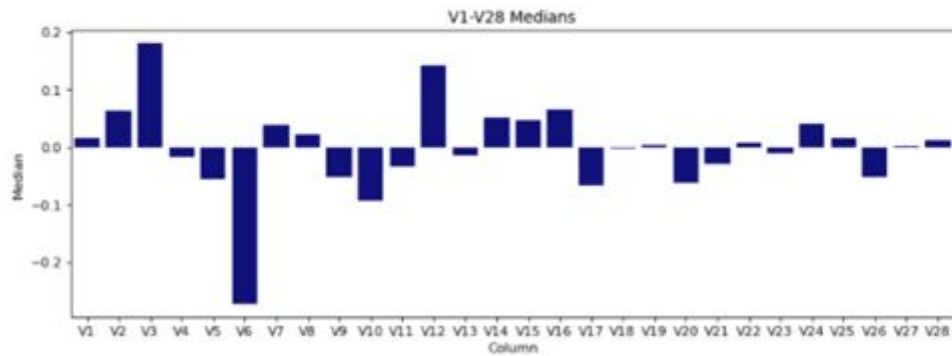
Text(0.5, 1.0, 'V1-V28 Kurtoses')



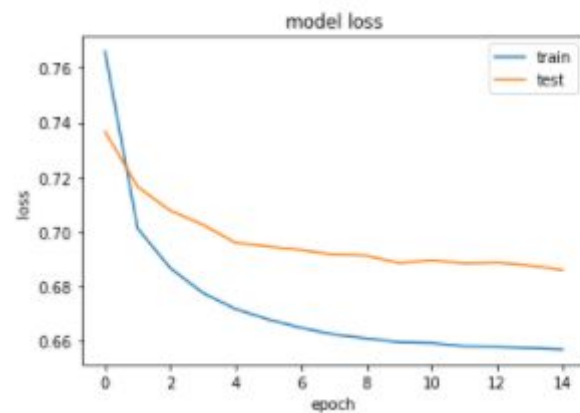
Hemos aprendido que muchas de las variables de PCA tienen colas pesadas. La gran cantidad de valores atípicos en "V1-V28" nos motiva a considerar estadísticas descriptivas sólidas. Grafiquemos las medianas:

```
1 plt.figure(figsize=(12,4), dpi=80)
2 sns.barplot(x=pca_vars, y=X_train[pca_vars].median(), color='darkblue')
3 plt.xlabel('Column')
4 plt.ylabel('Median')
5 plt.title('V1-V28 Medians')
```

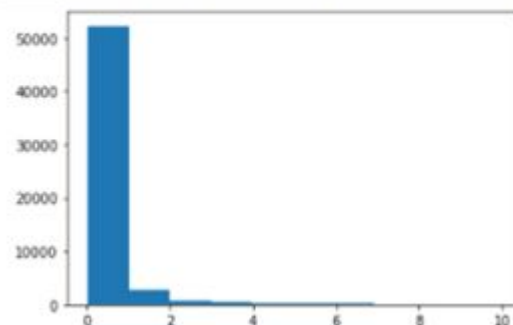
Text(0.5, 1.0, 'V1-V28 Medians')



```
1 plt.plot(history['loss'])
2 plt.plot(history['val_loss'])
3 plt.title('model loss')
4 plt.ylabel('loss')
5 plt.xlabel('epoch')
6 plt.legend(['train', 'test'], loc='upper right');
```



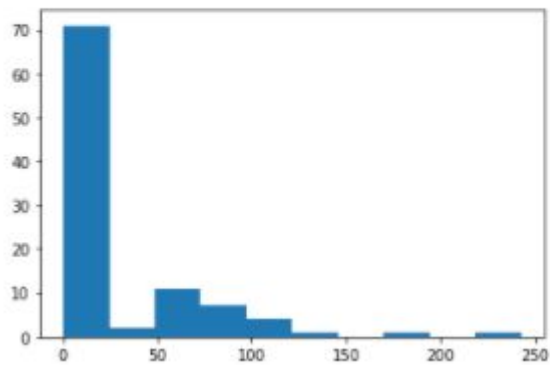
```
1 fig = plt.figure()
2 ax = fig.add_subplot(111)
3 normal_error_df = error_df[(error_df['true_class']== 0) & (error_df['reconstruction_error'] < 10)]
4 _ = ax.hist(normal_error_df.reconstruction_error.values, bins=10)
```



```

1 fig = plt.figure()
2 ax = fig.add_subplot(111)
3 fraud_error_df = error_df[error_df['true_class'] == 1]
4 _ = ax.hist(fraud_error_df.reconstruction_error.values, bins=10)

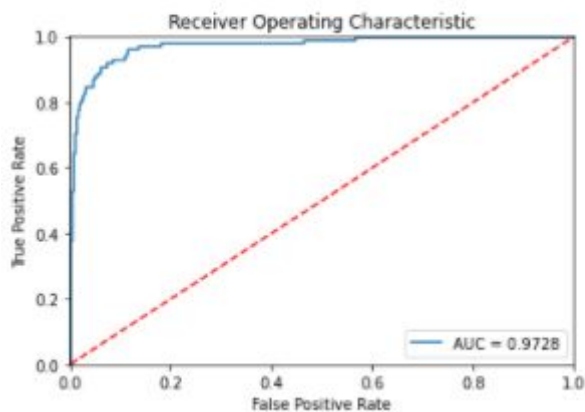
```



```

3 plt.title('Receiver Operating Characteristic')
4 plt.plot(fpr, tpr, label='AUC = %0.4f'% roc_auc)
5 plt.legend(loc='lower right')
6 plt.plot([0,1],[0,1], 'r--')
7 plt.xlim([-0.001, 1])
8 plt.ylim([0, 1.001])
9 plt.ylabel('True Positive Rate')
10 plt.xlabel('False Positive Rate')
11 plt.show();

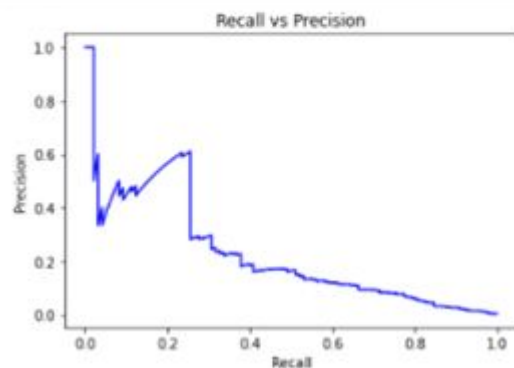
```



```

1 precision, recall, th = precision_recall_curve(error_df.true_class, error_df.reconstruction_error)
2 plt.plot(recall, precision, 'b', label='Precision-Recall curve')
3 plt.title('Recall vs Precision')
4 plt.xlabel('Recall')
5 plt.ylabel('Precision')
6 plt.show()

```



2.5. Modelos

2.5.1. Regresión Logística

```
1 from sklearn.pipeline import Pipeline
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.linear_model import SGDClassifier
```

```
1 pipeline_sgd = Pipeline([
2     ('scaler', StandardScaler(copy=False)),
3     ('model', SGDClassifier(max_iter=1000, tol=1e-3, random_state=1, warm_start=True))
4 ])
```

```
1 MCC_scorer = make_scorer(matthews_corrcoeff)
2 grid_sgd = GridSearchCV(estimator=pipeline_sgd, param_grid=param_grid_sgd, scoring=MCC_scorer, n_jobs=-1, pre_dispatch='2*n_
3
4
```

```
1 import warnings
2 with warnings.catch_warnings(): # Suppress warnings from the matthews_corrcoeff function
3     warnings.simplefilter("ignore")
4     grid_sgd.fit(X_train, y_train)
```

Fitting 5 folds for each of 80 candidates, totalling 400 fits

```
1 grid_sgd.best_score_
```

```
1 grid_sgd.best_params_
```

2.5.2 Random Forest

```
1 from sklearn.ensemble import RandomForestClassifier
```

```
1 pipeline_rf = Pipeline([
2     ('model', RandomForestClassifier(n_jobs=-1, random_state=1))
3 ])
```

```
1 param_grid_rf = {'model__n_estimators': [75]}
```

```
1 grid_rf = GridSearchCV(estimator=pipeline_rf, param_grid=param_grid_rf, scoring=MCC_scorer, n_jobs=-1, pre_dispatch='2*n_job
2
3
```



```
1 grid_rf.fit(X_train, y_train)
```

Fitting 5 folds for each of 1 candidates, totalling 5 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.  
[Parallel(n_jobs=-1)]: Done 5 out of 5 | elapsed: 2.5min finished
```

```
GridSearchCV(cv=5,  
             estimator=Pipeline(steps=[('model',  
                                       RandomForestClassifier(n_jobs=-1,  
                                                             random_state=1))]),  
             n_jobs=-1, param_grid={'model__n_estimators': [75]},  
             scoring=make_scorer(matthews_corrcoef), verbose=1)
```

```
1 grid_rf.best_score_
```

1.0

The random forest performed much better

```
1 grid_rf.best_params_
```

```
{'model__n_estimators': 75}
```

```
1 from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score, a
```

```
1 def classification_eval(estimator, X_test, y_test):  
2     y_pred = estimator.predict(X_test)  
3  
4     # Number of decimal places based on number of samples  
5     dec = np.int64(np.ceil(np.log10(len(y_test))))  
6  
7     print('CONFUSION MATRIX')  
8     print(confusion_matrix(y_test, y_pred), '\n')  
9  
10    print('CLASSIFICATION REPORT')  
11    print(classification_report(y_test, y_pred, digits=dec))  
12
```

```
: 1 classification_eval(grid_rf, X_test, y_test)
```

```
CONFUSION MATRIX  
[[56864  0]  
 [  0  98]]
```

```
CLASSIFICATION REPORT
```

	precision	recall	f1-score	support
0	1.00000	1.00000	1.00000	56864
1	1.00000	1.00000	1.00000	98
accuracy			1.00000	56962
macro avg	1.00000	1.00000	1.00000	56962
weighted avg	1.00000	1.00000	1.00000	56962

2.5.3. Red neuronal ANN

```
1 import tensorflow as tf
2 from tensorflow import keras
```

```
1 from sklearn.preprocessing import StandardScaler
2 transacciones['normalizedAmount'] = StandardScaler().fit_transform(transacciones['Amount'].values)
3 data = transacciones.drop(['Amount'],axis=1)
4 data = transacciones.drop(['Time'],axis=1)
5 data.head()
```

	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V10
0	0	-1.359807	-0.072781	2.538347	1.378155	-0.338321	0.482388	0.239599	0.098898	0.383787	...	0.2778
1	0	1.191857	0.286151	0.186480	0.448154	0.080018	-0.082381	-0.078803	0.085102	-0.255425	...	-0.6386
2	0	-1.358354	-1.340183	1.773209	0.379780	-0.503198	1.800499	0.791481	0.247876	-1.514854	...	0.7716
3	0	-0.988272	-0.185228	1.792993	-0.883291	-0.010309	1.247203	0.237809	0.377438	-1.387024	...	0.0052
4	0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	0.7982

5 rows x 32 columns

```
1 X = data.iloc[:, data.columns != 'Class']
2 y = data.iloc[:, data.columns == 'Class']
3 from sklearn.model_selection import train_test_split
4 X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state=0)
```

Preparar Datos ¶

```
from sklearn.preprocessing import StandardScaler
data = transacciones.drop(['Time'], axis=1)
data['Amount'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1, 1))
```

RANDOM_SEED=42

```
X_train, X_test = train_test_split(data, test_size=0.2, random_state=RANDOM_SEED)
X_train = X_train[X_train.Class == 0]
X_train = X_train.drop(['Class'], axis=1)
y_test = X_test['Class']
X_test = X_test.drop(['Class'], axis=1)
X_train = X_train.values
X_test = X_test.values
```

X_train.shape

451, 31)

2. Construir Modelo

```
input_dim = X_train.shape[1]
encoding_dim = 14
```

```
from sklearn.model_selection import train_test_split
from keras.models import Model, load_model
from keras.layers import Input, Dense
from keras.callbacks import ModelCheckpoint, TensorBoard
from keras import regularizers
input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="tanh",
                activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoder = Dense(int(encoding_dim / 2), activation="relu")(encoder)
decoder = Dense(int(encoding_dim / 2), activation='tanh')(encoder)
decoder = Dense(input_dim, activation='relu')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
```



```

1 nb_epoch = 15
2 batch_size = 32
3 autoencoder.compile(optimizer='adam',
4                     loss='mean_squared_error',
5                     metrics=['accuracy'])
6 checkpointer = ModelCheckpoint(filepath="model.h5",
7                                verbose=0,
8                                save_best_only=True)
9 tensorboard = TensorBoard(log_dir='/media/old-tf-hackers-7/logs',
10                           histogram_freq=0,
11                           write_graph=True,
12                           write_images=True)
13 history = autoencoder.fit(X_train, X_train,
14                          epochs=nb_epoch,
15                          batch_size=batch_size,
16                          shuffle=True,
17                          validation_data=(X_test, X_test),
18                          verbose=1,
19                          callbacks=[checkpointer, tensorboard]).history

```

Epoch 1/15

1/7108 [.....] - ETA: 0s - loss: 1.3746 - accuracy: 0.0625WARNING:tensorflow:From d:\miniconda3\lib\site-packages\tensorflow\python\ops\summary_ops_v2.py:1277: stop (from tensorflow.python.eager.profiler) is deprecated and will be removed after 2020-07-01.

Instructions for updating:

use 'tf.profiler.experimental.stop' instead.

2/7108 [.....] - ETA: 46:08 - loss: 1.0535 - accuracy: 0.0781WARNING:tensorflow:Callbacks method 'on_train_batch_end' is slow compared to the batch time (batch time: 0.0400s vs 'on_train_batch_end' time: 0.7258s). Check your callbacks.

7108/7108 [.....] - 16s 2ms/step - loss: 0.7662 - accuracy: 0.5872 - val_loss: 0.7368 - val_accuracy: 0.6581

Epoch 2/15

7108/7108 [.....] - 17s 2ms/step - loss: 0.7012 - accuracy: 0.6687 - val_loss: 0.7165 - val_accuracy: 0.6776

Epoch 3/15

7108/7108 [.....] - 20s 3ms/step - loss: 0.6867 - accuracy: 0.6848 - val_loss: 0.7076 - val_accuracy:

```

1 predictions = autoencoder.predict(X_test)
2 mse = np.mean(np.power(X_test - predictions, 2), axis=1)
3 error_df = pd.DataFrame({'reconstruction_error': mse,
4                          'true_class': y_test})
5 error_df.describe()

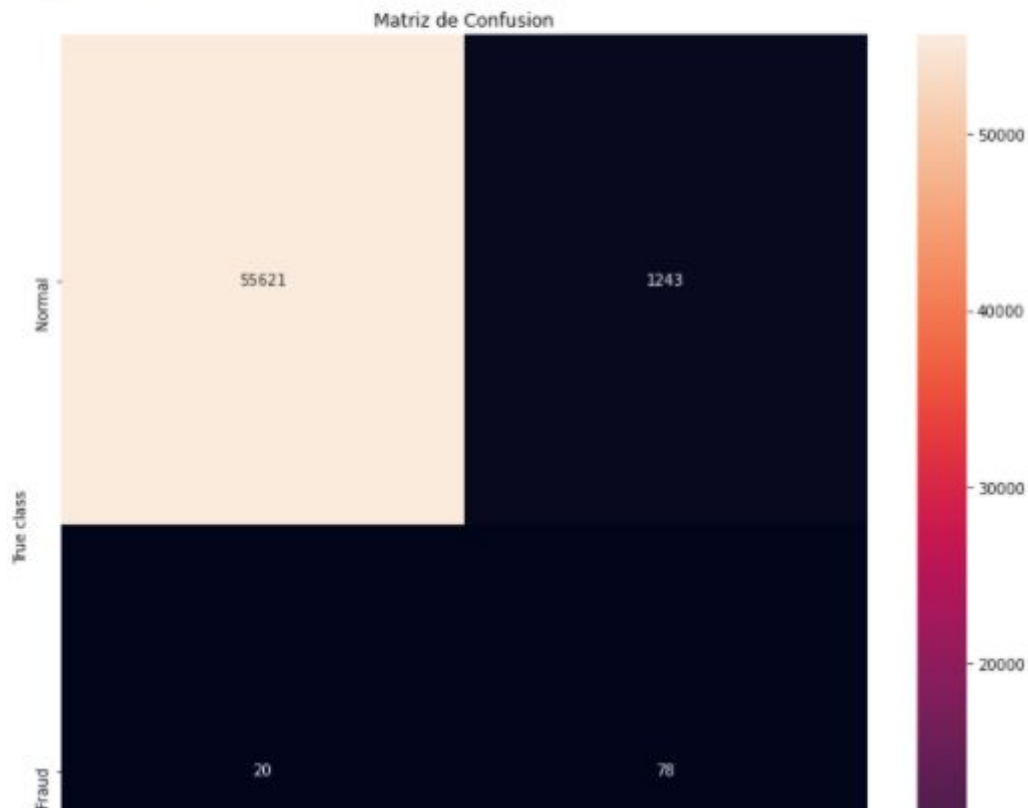
```

	reconstruction_error	true_class
count	58982.000000	58982.000000
mean	0.685082	0.001720
std	3.212940	0.041443
min	0.035913	0.000000
25%	0.223189	0.000000
50%	0.357303	0.000000
75%	0.567823	0.000000
max	242.401172	1.000000

```

1 threshold = 2.9
2 y_pred = [1 if e > threshold else 0 for e in error_df.reconstruction_error.values]
3 conf_matrix = confusion_matrix(error_df.true_class, y_pred)
4 plt.figure(figsize=(12, 12))
5 sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d");
6 plt.title("Matriz de Confusion")
7 plt.ylabel('True class')
8 plt.xlabel('Clase predictora')
9 plt.show()

```



```

1 #Ver la actuación del modelo
2 from sklearn.metrics import classification_report
3 print(classification_report(y_test, y_pred))

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	85296
1	0.95	0.99	0.97	147
accuracy			1.00	85443
macro avg	0.97	0.99	0.98	85443
weighted avg	1.00	1.00	1.00	85443

2.6. Exportación y comunicación.

```
transacciones.to_csv('backupSuper.csv')
```

```
1 mysqldump -u root -p creditcart-1 > BackupBD.sql
```

mysqldump -u root -p creditcart-1 > BackupBD.sql

3. Funcionalidades Adicionales

3.1 Envío de Correo Electrónico

```
: 1 !pip install python-sendmail
Collecting python-sendmail
  Downloading python-sendmail-0.3.0.tar.gz
Requirement already satisfied: click in d:
Building wheels for collected packages: py
  Building wheel for python-sendmail (setu
  Building wheel for python-sendmail (setu
  Created wheel for python-sendmail: file
e208e96c877eaef2f6c1e7ab55768f744e8230e
  Stored in directory: c:\users\gustavo\ap
26f
Successfully built python-sendmail
Installing collected packages: python-send
Successfully installed python-sendmail-0.3
```

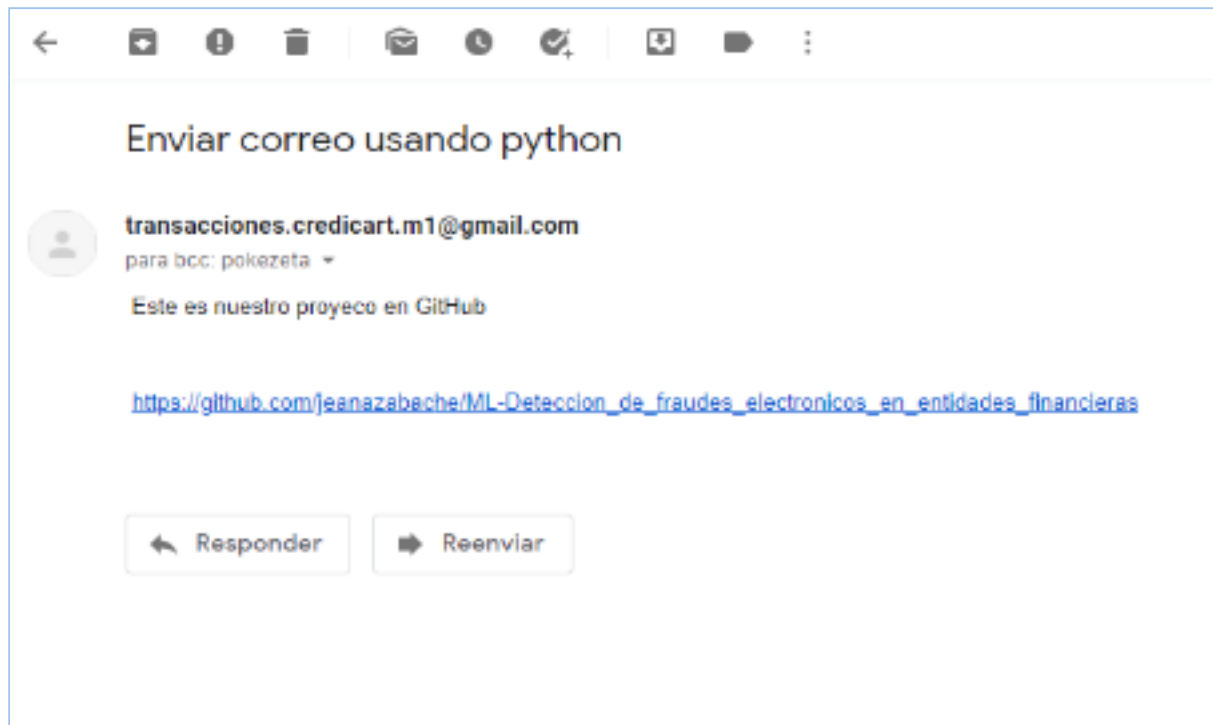
```
: 1 !Pip install secure-smtplib
Collecting secure-smtplib
  Downloading secure_smtplib-0.1.1-py2.py3
Installing collected packages: secure-smtp
Successfully installed secure-smtplib-0.1.
```

```
: 1 import smtplib
```

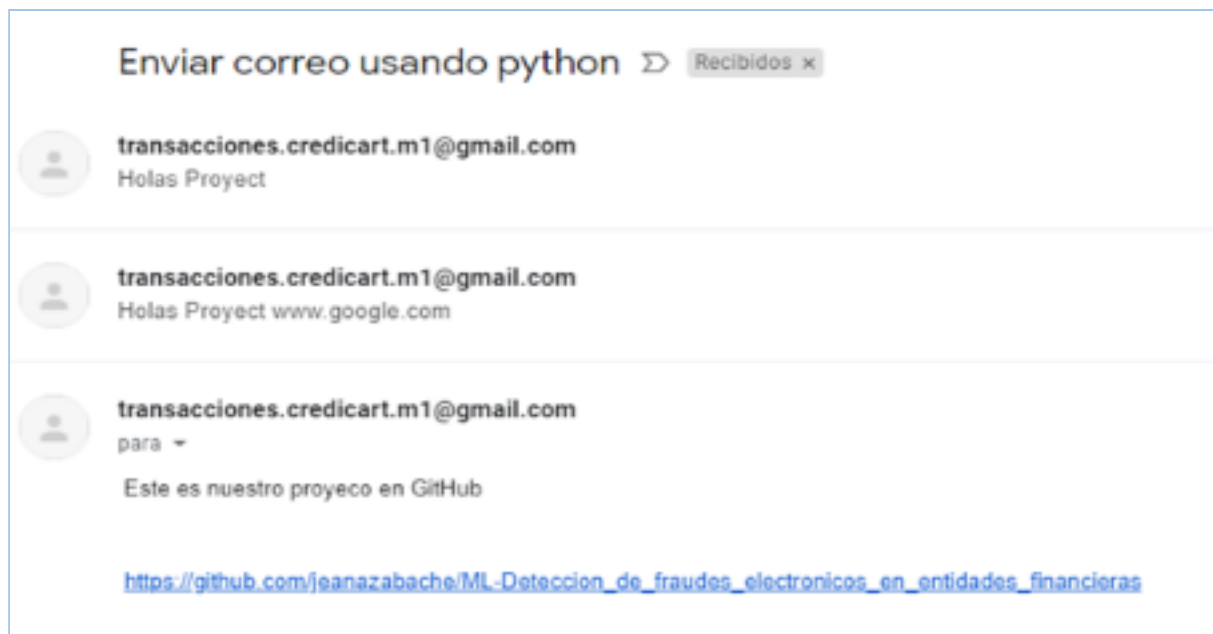
```
In [3]: import smtplib
conn=smtplib.SMTP('smtp.gmail.com',587)
type(conn)
conn.ehlo()
conn.starttls()
conn.login('transacciones.credicart.m1@gmail.com','Credicart1')
conn.sendmail('transacciones.credicart.m1@gmail.com','pokezeta@gmail.com','Subject: Enviar correo
```

```
Out[3]: {}
```

Registro de Mensaje de Cuenta Remitente



Registro de Mensaje de Cuenta Destinatario



4. Aplicación Web

4.1 Gráficos realizados con Plotly y transformados a Json

```
from flask import Flask
import json
import plotly
import pandas as pd
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/')
def histograma_time():
    fig = px.histogram(X_train, x="Time")
    graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return graphJSON

@app.route('/amount')
def histograma_amount():
    fig = px.histogram(X_train, x="Amount")
    graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return graphJSON

@app.route('/box')
def box_amount():
    fig = px.box(X_train, y="Amount")
    graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return graphJSON

@app.route('/train_amount')
def histograma_amount_train():
    fig = px.histogram(X_train, x="Amount")
    graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return graphJSON

@app.route('/bar_mean')
def bar_mean():
    fig = px.bar(x=['V%i' % k for k in range(1,29)], y=X_train[pca_vars].mean())
    graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return graphJSON

@app.route('/bar_De')
def bar_de():
    fig = px.bar(x=['V%i' % k for k in range(1,29)], y=X_train[pca_vars].std())
    graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return graphJSON

@app.route('/bar_skew')
```



```

{[id]: "[signgroup": "Time", "ringroup": "X", "timesamples": "Time(s)",
name="h[j]", "legengroup": "", "major": "[boku": "h5564", "name": "", "aliasgroup": "", "comment": "", "showgroup": "file", "c": "[133330, 133612, 44897, 785730, 155618, 119146, 35492, 89638, 114005,
134000, 137540, 90305, 147731, 148701, 167020, 153271, 145330, 123360, 67329, 59572, 81361, 75667, 78510, 13541, 41163, 90340, 118019, 41492, 147896, 13471, 130103, 45700, 41
147896, 151803, 147394, 157550, 135521, 151210, 149403, 113732, 43519, 533230, 161010, 80697, 148353, 40631, 7588, 81235, 147553, 1217030, 155240, 155794, 52865, 149121, 1653
49451, 144083, 155451, 84710, 121471, 37889, 84970, 117450, 129740, 67031, 138550, 16310, 120629, 135771, 138800, 78571, 42178, 135774, 13303, 112940, 198701, 10781, 10873, 10873
5913, 4106, 115671, 59543, 5786, 145400, 4060, 146810, 14040, 134001, 118733, 6780, 74541, 148550, 74140, 59470, 125060, 126601, 113060, 83000, 55075, 119700, 149450,
145770, 151057, 156707, 154290, 14430, 19180, 11791, 117941, 23470, 162400, 60693, 152070, 125451, 139951, 41781, 149750, 69250, 7118, 5410, 43368, 15170, 7040, 45913, 1
129510, 4633, 141810, 176540, 134601, 155623, 62570, 3748, 156550, 43450, 11225, 78429, 162035, 47070, 55785, 158250, 50060, 157616, 86740, 111531, 8558, 160050, 15279, 1
138470, 121842, 111240, 136120, 31170, 151848, 25810, 167848, 3357, 57733, 40214, 157000, 5157, 147850, 68031, 147450, 15671, 146770, 14592, 139100, 15217, 13672, 1334
150210, 108550, 152783, 36570, 14067, 115490, 5018, 5085, 82340, 400, 152840, 121278, 118390, 16341, 41598, 85615, 37180, 34420, 166574, 86270, 117380, 7816, 120805, 140
17030, 151012, 160980, 88807, 81079, 7800, 169793, 122920, 113210, 163341, 136880, 18010, 357520, 139061, 134730, 113360, 168298, 5103, 117557, 137410, 150490, 31641, 106
158150, 140128, 90303, 43840, 112896, 36097, 137540, 52704, 80415, 8200, 34523, 151570, 143278, 170380, 31406, 146950, 168001, 115230, 69045, 55177, 50523, 127274, 13940, 1
72512, 168089, 57300, 50157, 50141, 4510, 14432, 46531, 90240, 43540, 123360, 113710, 61597, 118730, 83073, 120010, 68540, 7834, 67112, 146820, 16499, 112100, 13078, 531
100350, 48910, 14640, 73420, 35840, 127218, 140405, 58651, 150720, 15485, 70800, 57741, 45451, 140500, 64791, 151730, 135770, 125094, 14711, 5064, 113114, 7506, 160070, 1
14710, 4642, 80780, 15145, 144861, 140890, 43457, 125190, 95700, 89560, 157450, 133236, 50814, 41979, 4060, 82930, 67300, 100871, 145228, 75794, 52867, 129690, 148210, 5
146310, 124410, 17038, 80241, 71802, 410110, 152815, 48080, 45750, 77050, 17070, 35320, 40600, 72800, 68260, 126510, 52710, 151410, 71460, 48390, 112260, 125450, 75400, 51
157004, 133312, 4545, 137878, 49723, 80120, 70150, 91135, 124844, 142340, 78840, 53810, 421, 4723, 70479, 41810, 166800, 4784, 84483, 111789, 13657, 3448, 36871, 86540,
69115, 2851, 71011, 82750, 44610, 150750, 36831, 140650, 50100, 59210, 120920, 137703, 147287, 45140, 151420, 10456, 154750, 123550, 88470, 51810, 82000, 70810, 5907, 11
148350, 144850, 127120, 84160, 14023, 34410, 134700, 151000, 113278, 13903, 118750, 31157, 35347, 82380, 130402, 52823, 41613, 87415, 147577, 137540, 138670, 11932, 71673,
7437, 128750, 45150, 165370, 36820, 68780, 59421, 118250, 121497, 89300, 135770, 81295, 100851, 146330, 16738, 115360, 125450, 51570, 71650, 121830, 112657, 90035, 56771,
125430, 82840, 11585, 48825, 117030, 131914, 36719, 70790, 130876, 146780, 137170, 15888, 112001, 115177, 40680, 73540, 31357, 155901, 16308, 80498, 54420, 52250, 16678,
40113, 67422, 124657, 13811, 121630, 150689, 71820, 457120, 110144, 11088, 74365, 59770, 85800, 121141, 15560, 13725, 55400, 32135, 57805, 38430, 36300, 60410, 196
12000, 82800, 72540, 19000, 133240, 157700, 51900, 70000, 82575, 136370, 158401, 151800, 85410, 71810, 168001, 31390, 147896, 85854, 126850, 15685, 118925, 63713, 7
7024, 168410, 178780, 40980, 217320, 62550, 40410, 55180, 61797, 78623, 120030, 51254, 36538, 44908, 119757, 36519, 110190, 167802, 28150, 160050, 16562, 119700, 156840, 57
4485, 1471290, 12907, 147737, 8108, 71589, 168410, 136750, 133456, 2700, 168420, 5278, 128750, 13108, 82410, 168040, 17991, 31430, 128450, 16112, 7158, 168078, 1681
153840, 71810, 14221, 68006
```


4.3 Código en Visual Studio Code

```
Ejecutar Terminal Ayuda Introduccion.vue - proyecto-1 - Visual Studio Code
▼ App.vue ▼ Introduccion.vue X
src > components > ▼ Introduccion.vue > {} "Introduccion.vue" > template > div.Introduccion
1 <template>
2   <div class="Introduccion">
3     <h1 :style="{font-size: ' + range + 'px'}">{{msg}}</h1>
4     <h1>Identificación de fraude electrónico</h1>
5     <h2>Curso de MainFrame 1</h2>
6     <h2>Autores : <br>
7       - Azabache Medina, Jean Pierre<br>
8       - Patiño Hermoza, Ze Carlos<br></h2>
9     <p></p>
10    <p></p>
11    <button type="button" @click="tab=1">Introducción </button>
12    <button type="button" @click="tab=2">Exploración de Datos </button>
13    <button type="button" @click="tab=3">Modelado de Datos </button>
14    <button type="button" @click="tab=4">Preparación de Datos </button>
15    <button type="button" @click="tab=5">Modelos de Predicción</button>
16    <button type="button" @click="tab=6">Visualizacion de Datos </button>
17    <p>Mi DashBoard Personalizado</p>
18    <input type="range" min="8" max="80" v-model="range">
19
20
21  <div>
22
23    <h2>Histograma de Monto de transacciones</h2>
24    <Plotly v-if="tab==6" :data="grafico1.data" :layout="grafico1.layout" :display-mode-bar="true"></Plotly>
25    <h2>Diagrama de caja de Monto de transacciones</h2>
26    <Plotly v-if="tab==6" :data="grafico2.data" :layout="grafico2.layout" :display-mode-bar="true"></Plotly>
27    <h2>Diagrama de barras de la Media entre V1 - V28</h2>
28    <Plotly v-if="tab==6" :data="grafico3.data" :layout="grafico3.layout" :display-mode-bar="true"></Plotly>
29    <h2>Diagrama de Barras de la desviación estandar entre V1 - V28</h2>
30    <Plotly v-if="tab==6" :data="grafico5.data" :layout="grafico5.layout" :display-mode-bar="true"></Plotly>
31    <h2>Diagrama de Barras de la simetría entre V1 - V28</h2>
32    <Plotly v-if="tab==6" :data="grafico6.data" :layout="grafico6.layout" :display-mode-bar="true"></Plotly>
33    <h2>Diagrama de Barras de la varialbe V8</h2>
34    <Plotly v-if="tab==6" :data="grafico7.data" :layout="grafico7.layout" :display-mode-bar="true"></Plotly>
35    <h2>Diagrama de Caja de la variable entre V8</h2>
36    <Plotly v-if="tab==6" :data="grafico8.data" :layout="grafico8.layout" :display-mode-bar="true"></Plotly>
37    <h2>Diagrama de Barras de la curtosis entre V1 - V28</h2>
38    <Plotly v-if="tab==6" :data="grafico9.data" :layout="grafico9.layout" :display-mode-bar="true"></Plotly>
39    <h2>Diagrama de Barras del rango intercuartil entre V1 - V28</h2>
40    <Plotly v-if="tab==6" :data="grafico10.data" :layout="grafico10.layout" :display-mode-bar="true"></Plotly>
```

4.4 Web Service



Identificación de fraude electrónico

Curso de MainFrame 1

Autores :

- Azabache Medina, Jean Pierre
- Patiño Hermoza, Ze Carlos

Introducción

Exploración de Datos

Modelado de Datos

Preparación de Datos

Modelos de Predicción

Visualización de Datos

Mi DashBoard Personalizado



Histograma de Monto de transacciones



Finalmente, el modelo está apto para cumplir su objetivo en el negocio, no obstante, el modelo deberá tener una constante mejora y actualización para aumentar su precisión en las nuevas modalidades de evasión hacia el sistema.

Conclusiones

Logramos identificar el modelo adecuado de aprendizaje automático para la predicción de transacciones fraudulentas que fueron realizadas por usuarios con sus tarjetas de crédito, en esta evaluación se tomaron en cuenta los modelos de bosque aleatorio, regresión logística, redes neuronales, obteniendo como resultado el modelo con el mayor puntaje de precisión en predecir las operaciones fraudulentas que ocurrirán en los bancos y estos podrán tomar medidas para evitar esto (planes de contingencia).