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 libreria de `Pandas`

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

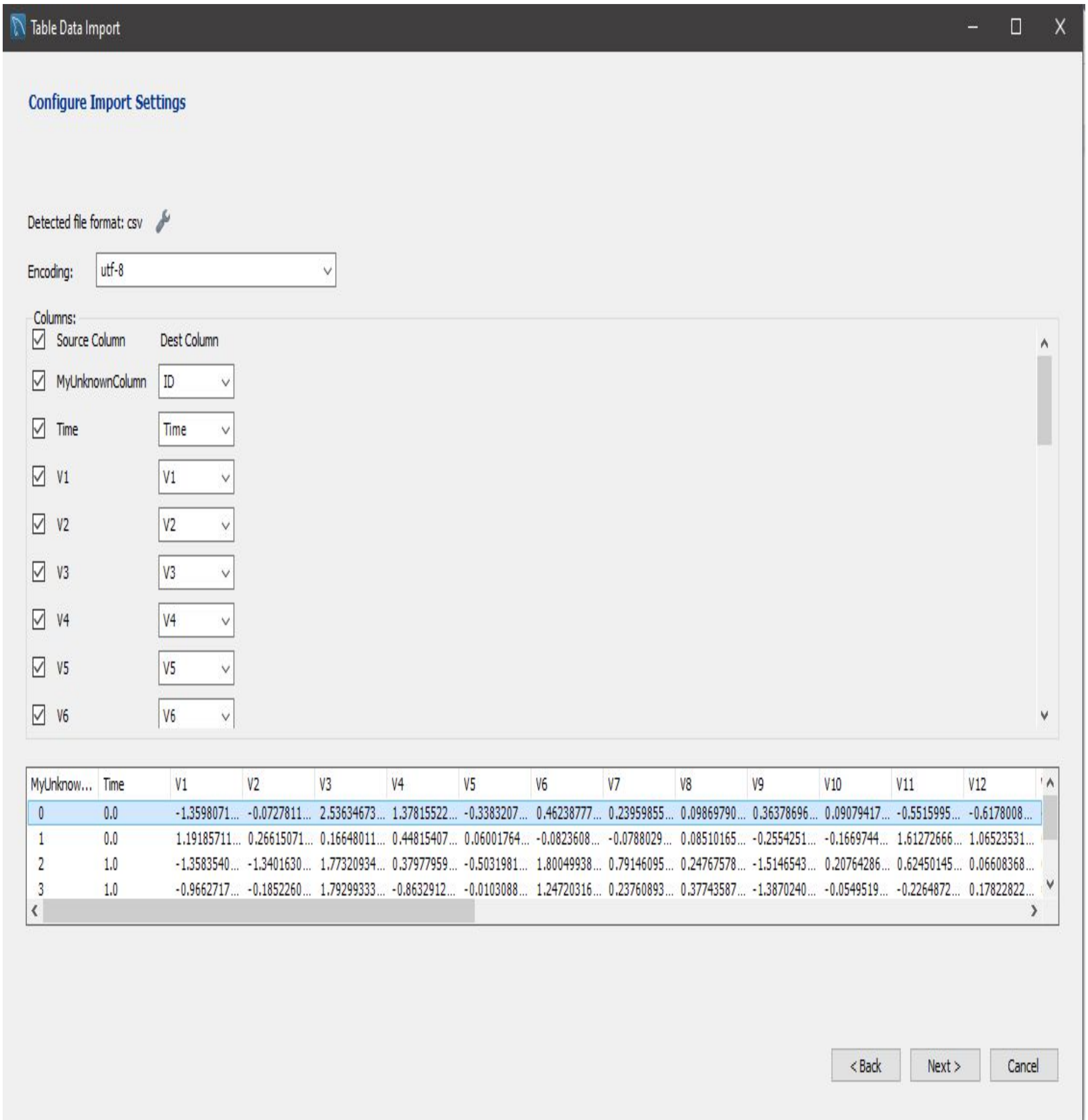
2.2. Modelo de datos estructurados

2.2.1 Diseño de Tabla en MySql

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



2.3. Transformación y consultas

Estadisticas Descriptivas

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

```
94813.85957508067
```

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

```
88.34961925093133
```

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

```
84692.0
```

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

```
22.0
```

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

```
19880.72
```

```python
q4=transacciones.Amount.quantile(0.04)
q4
```

```
0.89
```

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

```
54201.5
```

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

```
5.6
```

```python
maximo=transacciones.Time.max()
maximo
```

```
172792.0
```

```python
maximo=transacciones.Amount.max()
maximo
```

```
25691.16
```

```python
minimo=transacciones.Amount.min()
minimo
```

```
0.0
```

```python
minimo=transacciones.Time.min()
minimo
```

```
0.0
```

Transformación

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

```python
transacciones.Resultados=transacciones.Class
```

```python
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	0.1
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	0.1
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	-0.3
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	0.6
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	-0.2

5 rows × 32 columns

```python
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

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

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

```python
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	...	-0.166215	-0.501598	0.12
77470	57062.0	0	-1.188664	-0.612034	2.422204	-0.812786	0.318493	-0.671637	-0.432053	0.068237	...	0.002347	0.164823	-0.09
190885	129019.0	0	1.868263	0.273764	-0.288023	3.835852	0.268329	0.817380	-0.287993	0.203258	...	0.115927	0.610472	0.02
87335	61640.0	0	-0.848470	1.426562	2.137094	2.852036	-0.366945	1.158146	-0.416142	0.812490	...	-0.210710	-0.369433	-0.27
174481	121931.0	0	-1.184195	0.804518	2.240498	2.853175	1.038068	0.171728	0.457665	0.290123	...	0.110209	0.254591	-0.47
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
54018	46253.0	0	-21.780665	-38.305310	-12.122469	9.752791	-12.880794	4.256017	14.785051	-2.818253	...	7.437478	-5.619439	-10.54
46841	42951.0	0	-23.712839	-42.172688	-13.320825	9.925019	-13.945538	5.564891	15.710644	-2.844253	...	7.921600	-6.320710	-11.31
151296	95286.0	0	-34.549296	-60.464618	-21.340854	16.875344	-19.229075	6.335259	24.422716	-4.964566	...	11.502580	-9.499423	-16.51
58465	48401.0	0	-36.802320	-63.344698	-20.645794	16.715537	-20.672064	7.694002	24.956587	-4.730111	...	11.455313	-10.933144	-17.17
274771	166198.0	0	-35.548539	-31.850484	-48.325589	15.304184	-113.743307	73.301626	120.589494	-27.347360	...	-21.620120	5.712303	-1.58

```python
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

```python
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	-0.414650	...	0.261057	0.643078	0.376777	0.0
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	0.1
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	0.6
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	-1.0
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	-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
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
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	0.0

284807 rows × 32 columns

```python
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
```

```python
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`

```python
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	0.64
48973	43858.0	0	-3.312835	2.181915	-0.762963	0.311522	2.215393	4.913325	-1.931967	-3.412496	...	4.616781	-1.752986	0.312910	0.92
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	-0.04
63715	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	1.01
207144	136538.0	0	2.026829	-0.235613	-3.122868	0.008242	1.279637	-0.812390	1.048054	-0.535275	...	0.240236	0.579850	-0.361863	-0.20

5 rows × 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	0.462388	0.239599	0.098698	...	-0.018307	0.277838	-0.110474	0.0
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	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	...	0.214205	0.924384	0.012463	-1.0
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	0.6
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	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 × 32 columns

```
]: 1 set(transacciones['Resultados'])
]: {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	-0.419818	0.114568	0.383328
90717	63139.0	Normal	-0.492877	-0.741865	2.339206	-2.291968	-1.291543	0.135619	-0.642833	0.174217	...	-0.073501	0.266854	0.054104	0.080246

5 rows × 32 columns

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

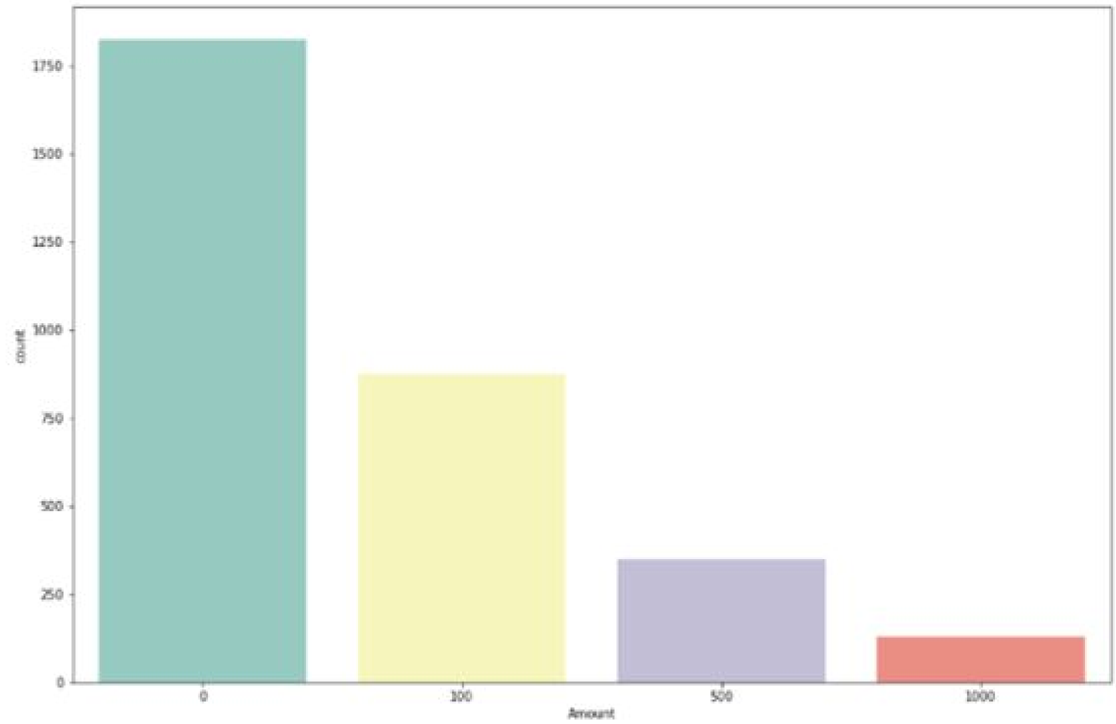
```python
from sklearn import preprocessing
```

```python
fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(6,5))
ax1.set_title('Antes de escalar')
sns.kdeplot(transacciones['V1'],ax=ax1)
sns.kdeplot(transacciones['V2'],ax=ax1)
sns.kdeplot(transacciones['V3'],ax=ax1)
sns.kdeplot(transacciones['V4'],ax=ax1)

scaler=preprocessing.StandardScaler()
transacciones[['Time','Amount']]=scaler.fit_transform(transacciones[['Time','Amount']])


ax1.set_title('Despues de escalar')
sns.kdeplot(transacciones['V1'],ax=ax2)
sns.kdeplot(transacciones['V2'],ax=ax2)
sns.kdeplot(transacciones['V3'],ax=ax2)
sns.kdeplot(transacciones['V4'],ax=ax2)
```

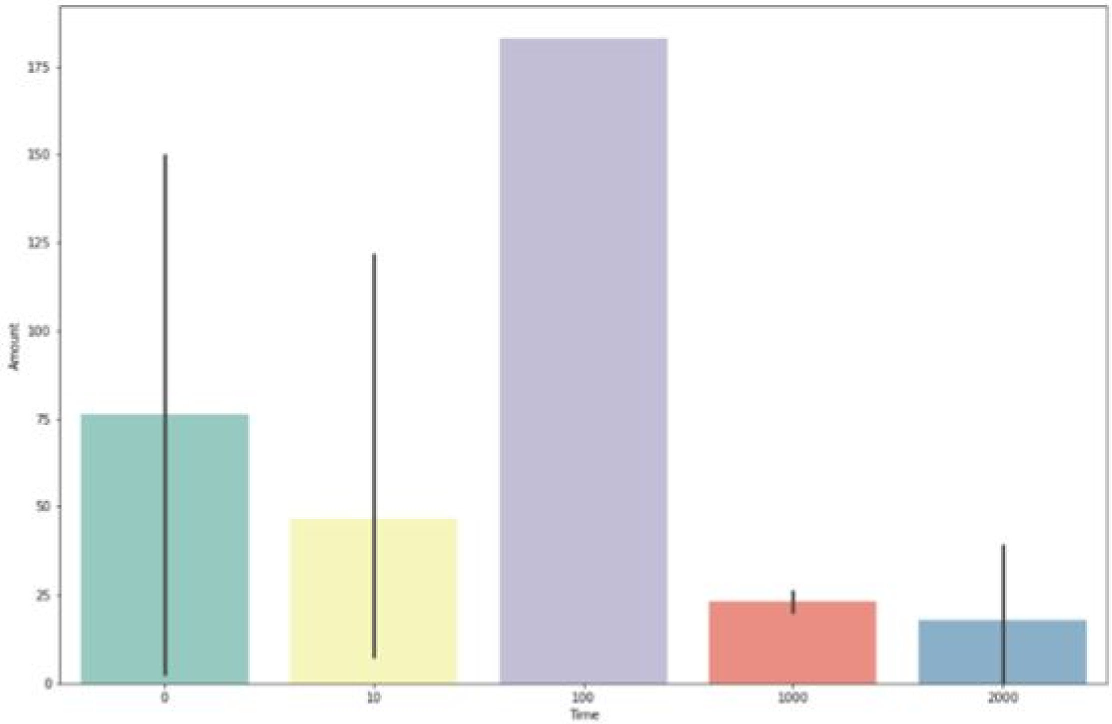
```
<AxesSubplot:>
```

2.4. Exploración Visual de datos

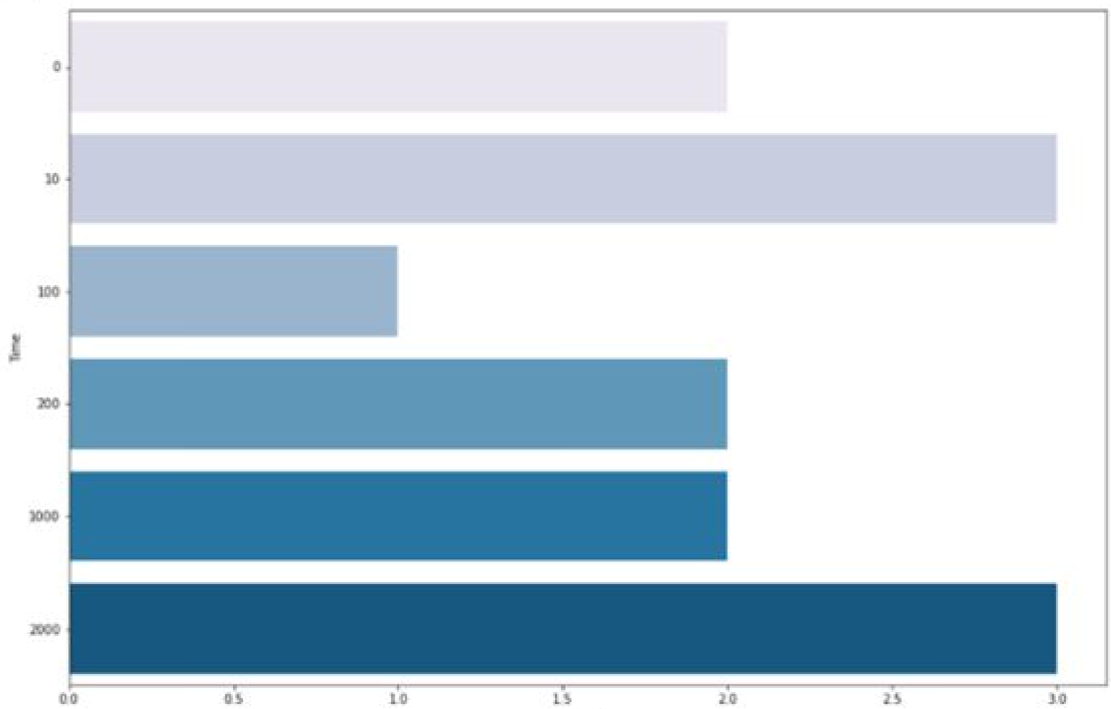
```python
plt.figure(figsize=(15,10))#canvas
sns.countplot(x="Amount",data=transacciones,palette="Set3",order=[0,100,500, 1000])
plt.show()
```



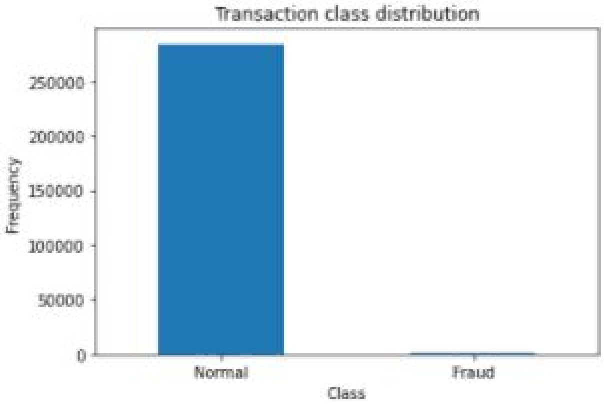
```
plt.figure(figsize=(15,10))#canvas
sns.barplot(x="Time",y="Amount",data=transacciones,palette="Set3",order=[0,10,100,1000,2000])
plt.show()
```




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

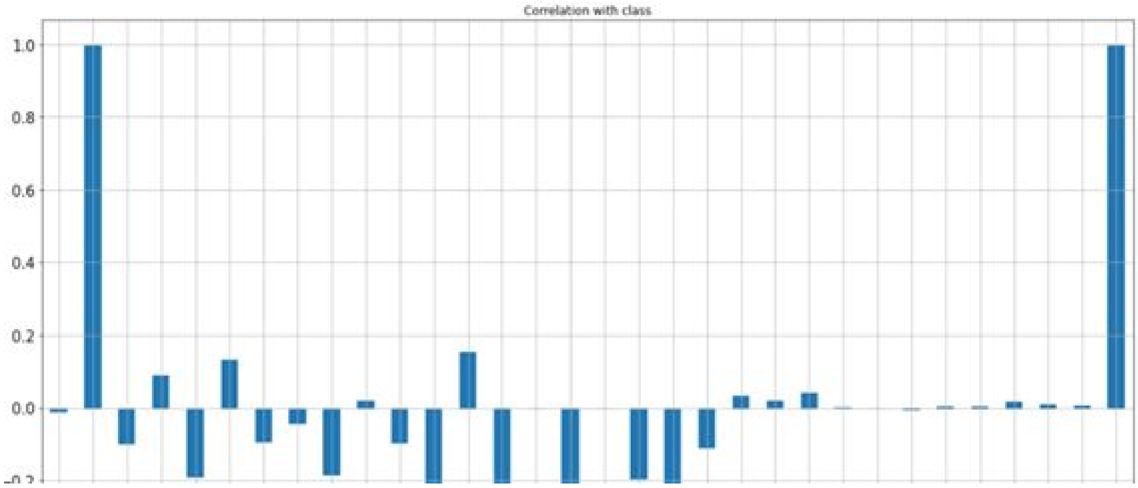


```python
count_classes = pd.value_counts(transacciones['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0)
plt.title("Transaction class distribution")
plt.xticks(range(2), LABELS)
plt.xlabel("Class")
plt.ylabel("Frequency");
```

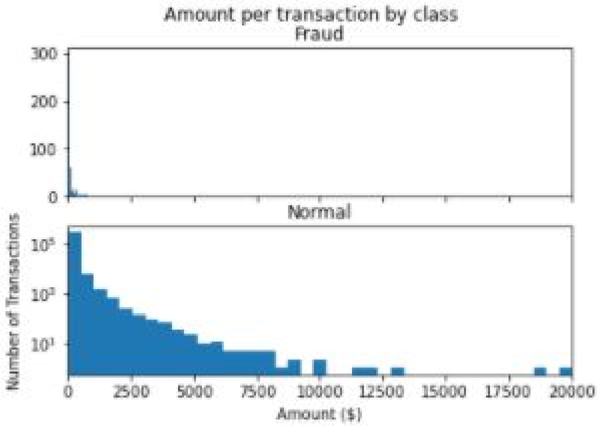



```python
transacciones.corrwith(transacciones.Class).plot.bar(
        figsize = (20, 10), title = "Correlation with class", fontsize = 15,
        rot = 45, grid = True)
```

<AxesSubplot:title={'center':'Correlation with class'}>



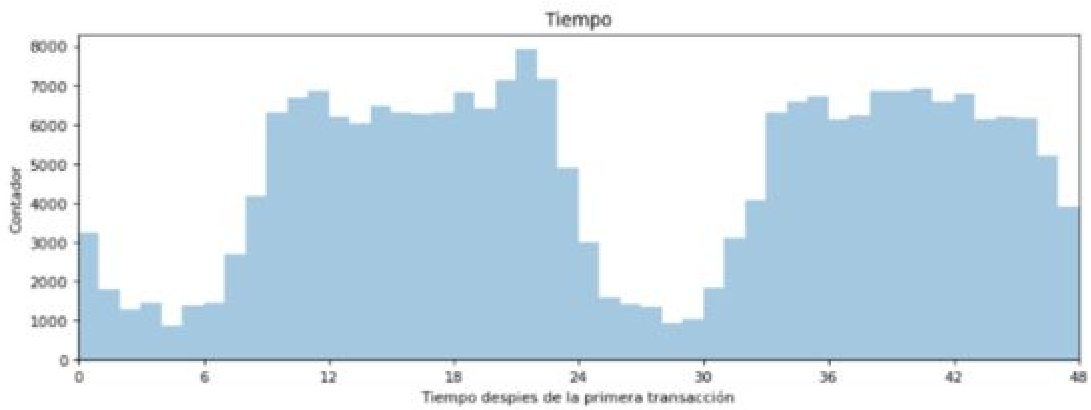
```python
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Amount per transaction by class')
bins = 50
ax1.hist(frauds.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();
```



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

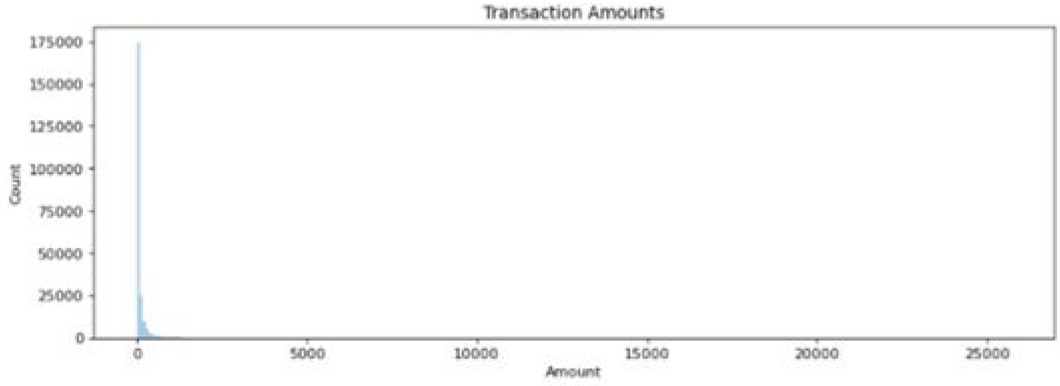
```python
plt.figure(figsize=(12,4), dpi=80)
sns.distplot(X_train['Time'], bins=48, kde=False)
plt.xlim([0,48])
plt.xticks(np.arange(0,54,6))
plt.xlabel('Tiempo despies de la primera transacción')
plt.ylabel('Contador')
plt.title('Tiempo')
```

```
Text(0.5, 1.0, 'Tiempo')
```



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

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



- Diagrama de caja para poder obervar los valores atipicos que no se pueden diferenciar en histograma realizado anteriorment

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

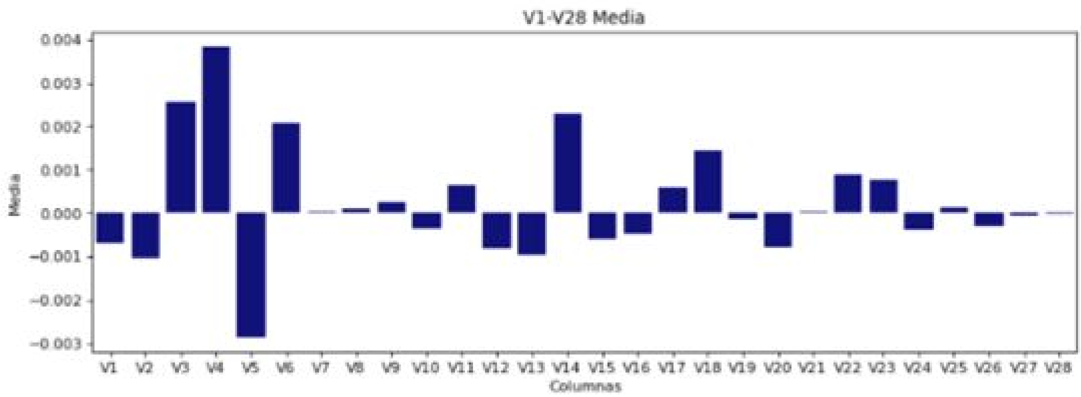
Text(0.5, 1.0, 'Transaction Amounts')

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

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

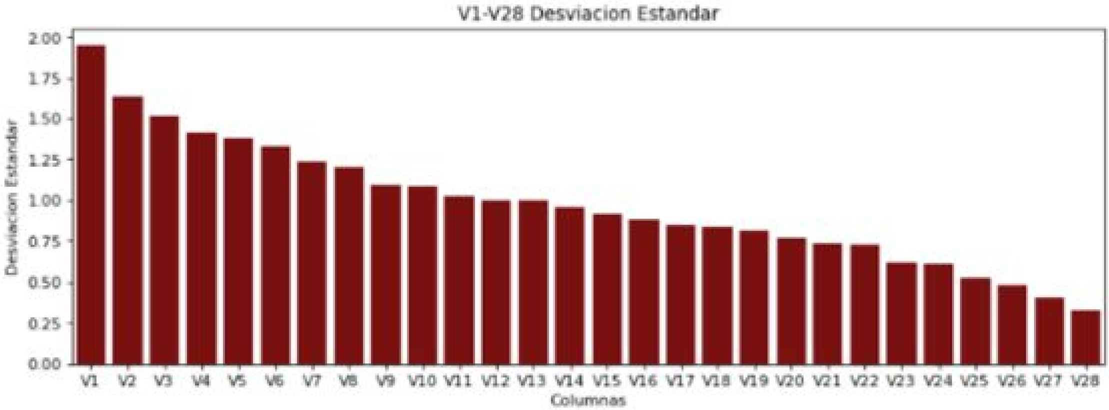

```
plt.figure(figsize=(12,4), dpi=80)
sns.barplot(x=pca_vars, y=X_train[pca_vars].mean(), color='darkblue')
plt.xlabel('Columnas')
plt.ylabel('Media')
plt.title('V1-V28 Media')
```

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



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

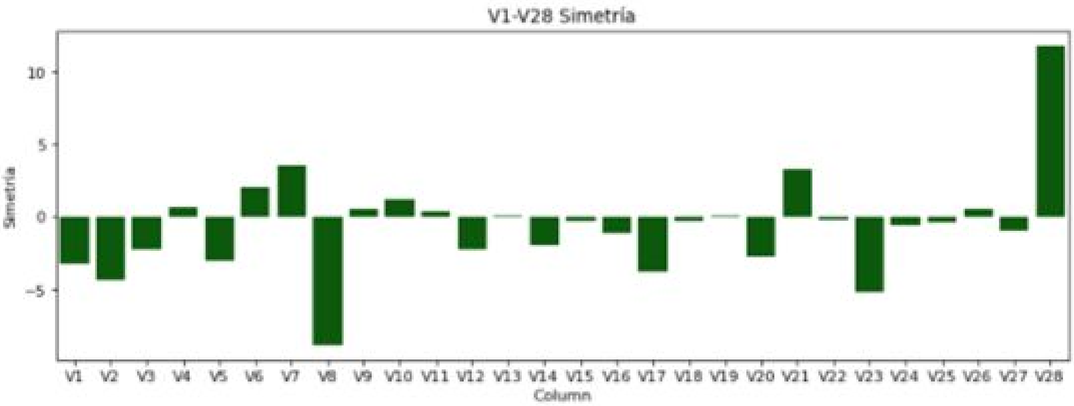
```
Text(0.5, 1.0, 'V1-V28 Desviacion Estandar')
```



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

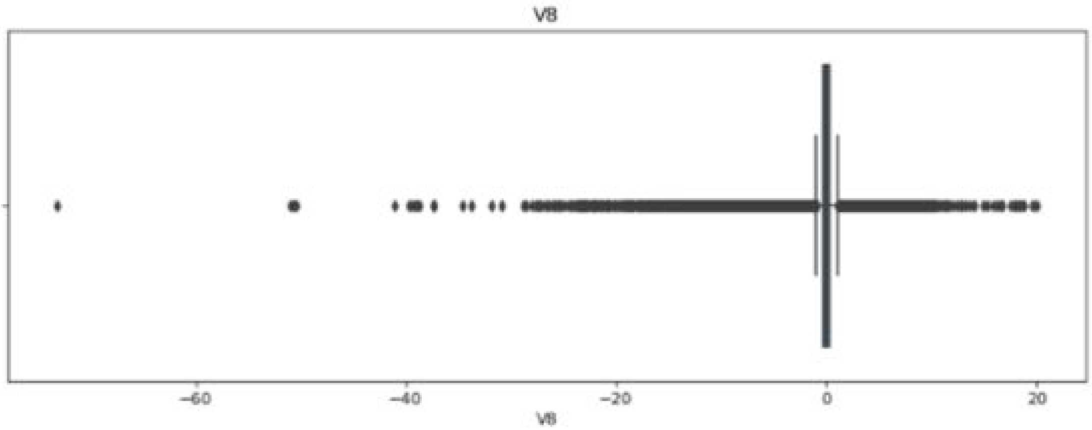
```python
plt.figure(figsize=(12,4), dpi=80)
sns.barplot(x=pca_vars, y=X_train[pca_vars].skew(), color='darkgreen')
plt.xlabel('Column')
plt.ylabel('Simetría')
plt.title('V1-V28 Simetría')
```

```
Text(0.5, 1.0, 'V1-V28 Simetría')
```



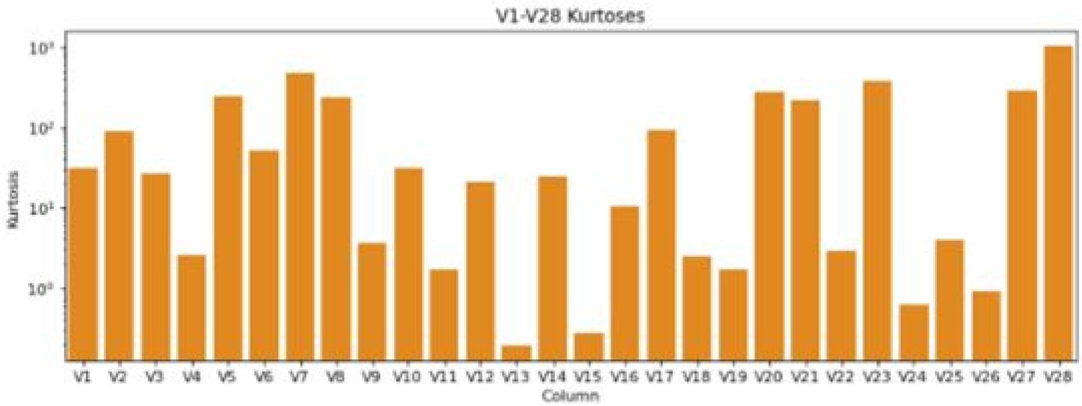
```
plt.figure(figsize=(12,4), dpi=80)
sns.boxplot(X_train['V8'])
plt.title('V8')
```

```
Text(0.5, 1.0, 'V8')
```




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

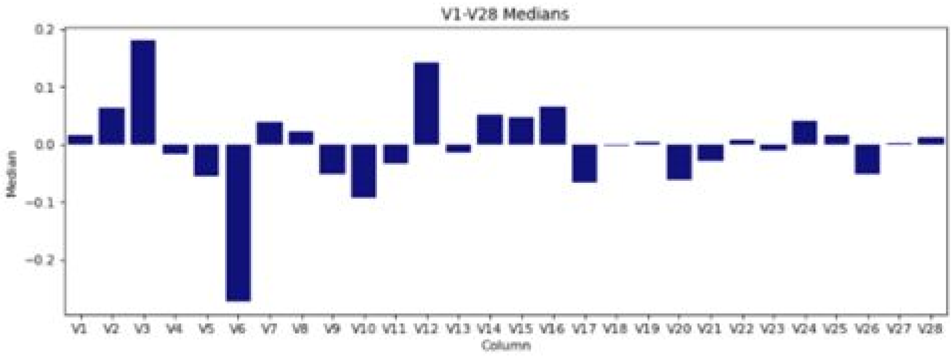
```
Text(0.5, 1.0, 'V1-V28 Kurtoses')
```



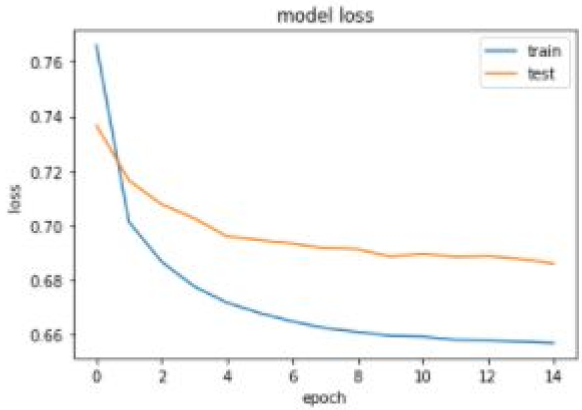
Hemos aprendido que muchas de las variables de PCA tienen colas pesadas. La gran cantidad de valores atípicos en "V1-V28" nos motiva a considerar estadísticas descriptivas sólidas. Grafiquemos las medianas:

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

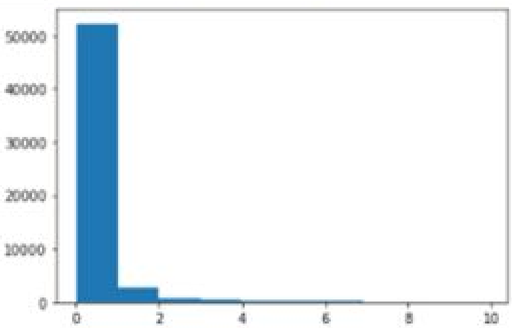
```
Text(0.5, 1.0, 'V1-V28 Medians')
```



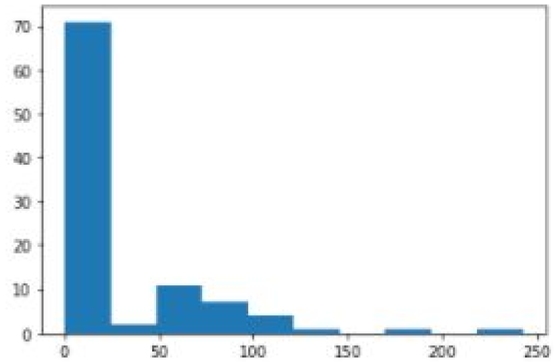

```
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right');
```



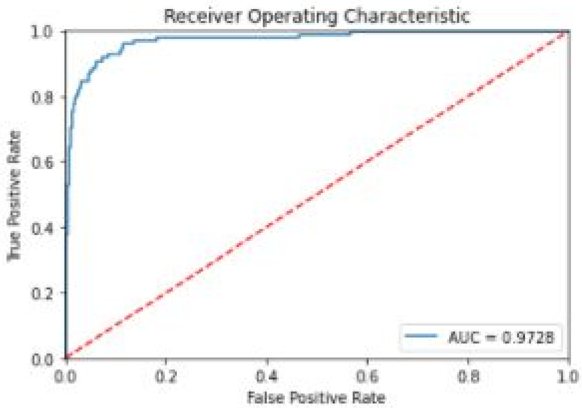

```
fig = plt.figure()
ax = fig.add_subplot(111)
normal_error_df = error_df[(error_df['true_class']== 0) & (error_df['reconstruction_error'] < 10)]
_ = ax.hist(normal_error_df.reconstruction_error.values, bins=10)
```



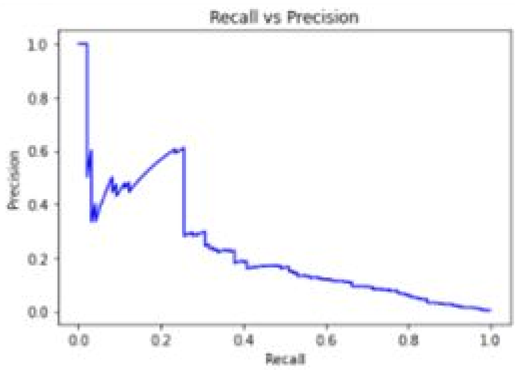
```
fig = plt.figure()
ax = fig.add_subplot(111)
fraud_error_df = error_df[error_df['true_class'] == 1]
_ = ax.hist(fraud_error_df.reconstruction_error.values, bins=10)
```



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




```
precision, recall, th = precision_recall_curve(error_df.true_class, error_df.reconstruction_error)
plt.plot(recall, precision, 'b', label='Precision-Recall curve')
plt.title('Recall vs Precision')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.show()
```



2.5. Modelos

2.5.1. Regresión Logística

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDClassifier
```

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

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

```
import warnings
with warnings.catch_warnings(): # Suppress warnings from the matthews_corrcoef function
    warnings.simplefilter("ignore")
    grid_sgd.fit(X_train, y_train)
```

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

```
grid_sgd.best_score_
```

```
grid_sgd.best_params_
```

2.5.2 Random Forest

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

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

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

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

```
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)
```

```
grid_rf.best_score_
```

```
1.0
```

The random forest performed much b

```
grid_rf.best_params_
```

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

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

```
def classification_eval(estimator, X_test, y_test):
    y_pred = estimator.predict(X_test)

    # Number of decimal places based on number of samples
    dec = np.int64(np.ceil(np.log10(len(y_test))))

    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test, y_pred), '\n')

    print('CLASSIFICATION REPORT')
    print(classification_report(y_test, y_pred, digits=dec))

```

```
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

```
import tensorflow as tf
from tensorflow import keras
```

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

	Resultados	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	0.2778
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.6386
2	0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.7716
3	0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-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 × 32 columns

```
X = data.iloc[:, data.columns != 'Class']
y = data.iloc[:, data.columns == 'Class']
from sklearn.model_selection import train_test_split
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)
```

```python
nb_epoch = 15
batch_size = 32
autoencoder.compile(optimizer='adam',
                    loss='mean_squared_error',
                    metrics=['accuracy'])
checkpointer = ModelCheckpoint(filepath="model.h5",
                               verbose=0,
                               save_best_only=True)
tensorboard = TensorBoard(log_dir='/media/old-tf-hackers-7/logs',
                          histogram_freq=0,
                          write_graph=True,
                          write_images=True)
history = autoencoder.fit(X_train, X_train,
                    epochs=nb_epoch,
                    batch_size=batch_size,
                    shuffle=True,
                    validation_data=(X_test, X_test),
                    verbose=1,
                    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 wil
l 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 `o
n_train_batch_end` is slow compared to the batch time (batch time: 0.0400s vs `on_train_batch_end` time: 0.7258s). Check your c
allbacks.
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: 0.6840 - val_loss: 0.7078 - val_accuracy:
```

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

	reconstruction_error	true_class
count	56962.000000	56962.000000
mean	0.685082	0.001720
std	3.212940	0.041443
min	0.035913	0.000000
25%	0.223169	0.000000
50%	0.357303	0.000000
75%	0.567823	0.000000
max	242.401172	1.000000

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


```
#Ver la actuación del modelo
from sklearn.metrics import classification_report
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')
```

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

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

3. Funcionalidades Adicionalidades

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: filen
e208e96c877eaef2f6c1e7ab55768f744e8230e
  Stored in directory: c:\users\gustavo\ap
26f
Successfully built python-sendmail
Installing collected packages: python-send
Successfully installed python-sendmail-0.3
```

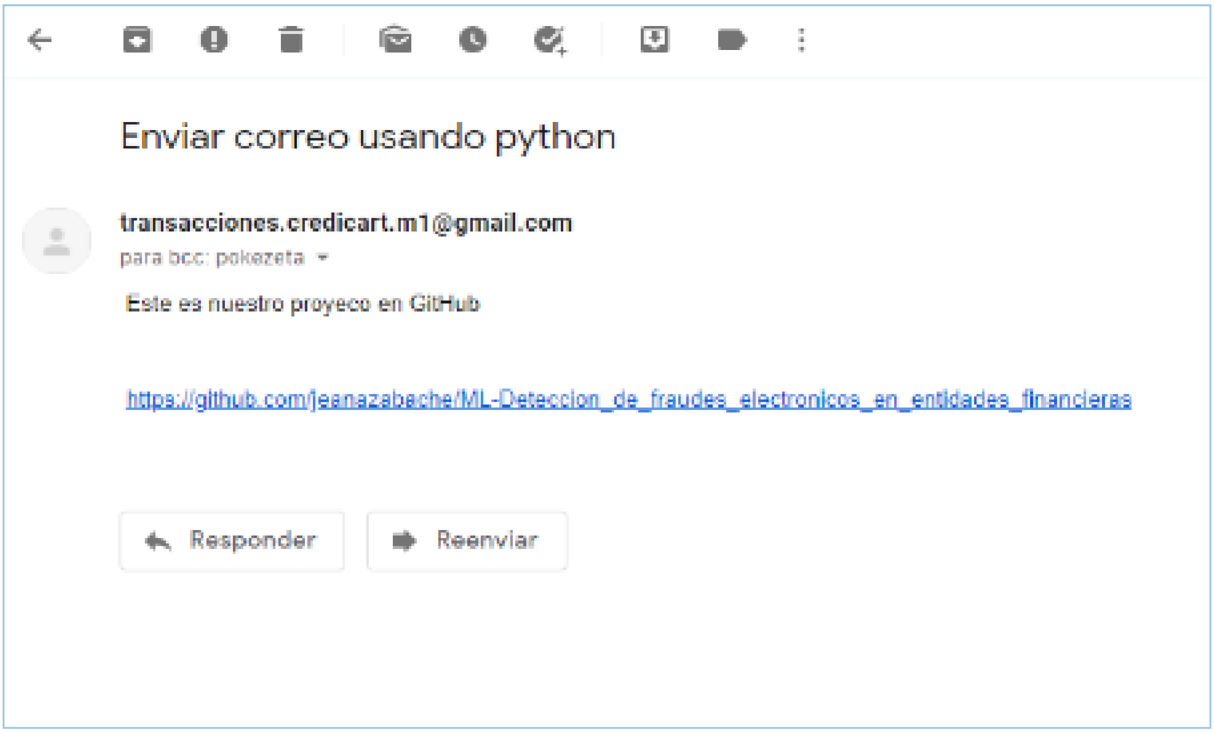
```
: 1  !Pip install secure-smtplib
```

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

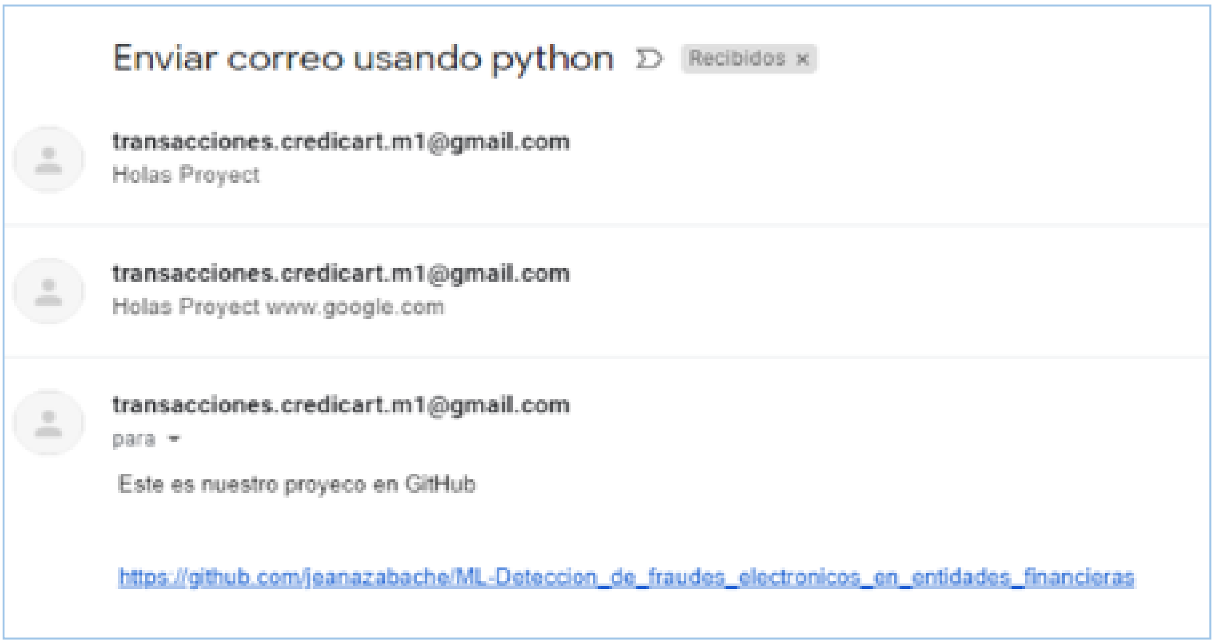
```
: 1  import smtplib
```

```
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

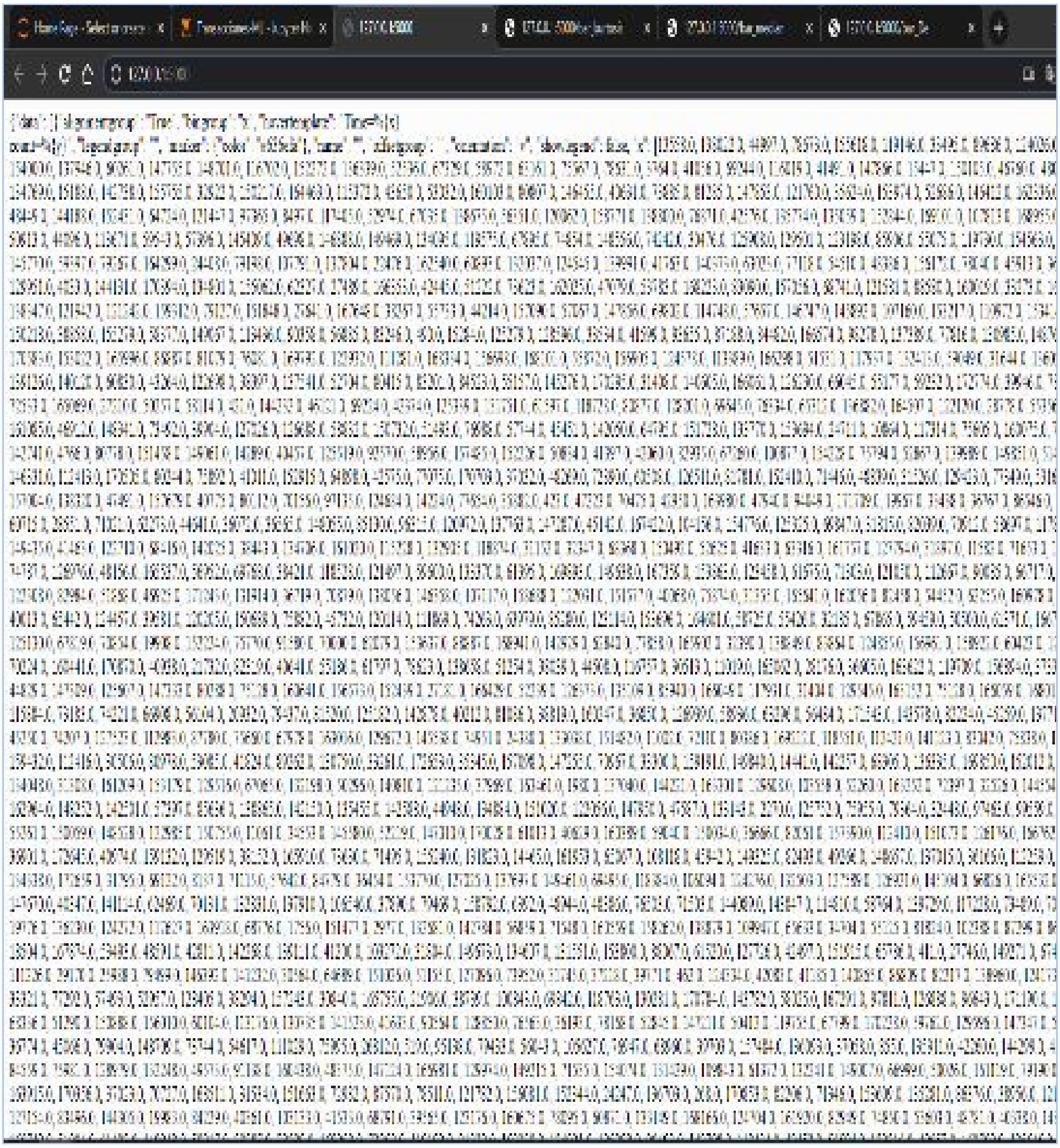


4. Aplicación Web

4.1 Gráficos realizados con Plotly y transformados a Json

```python
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')
```

4.2 Framework de Flak



4.3 Código en Visual Studio Code

Ejecutar Terminal Ayuda Introduccion.vue - proyecto-1 - Visual Studio Code

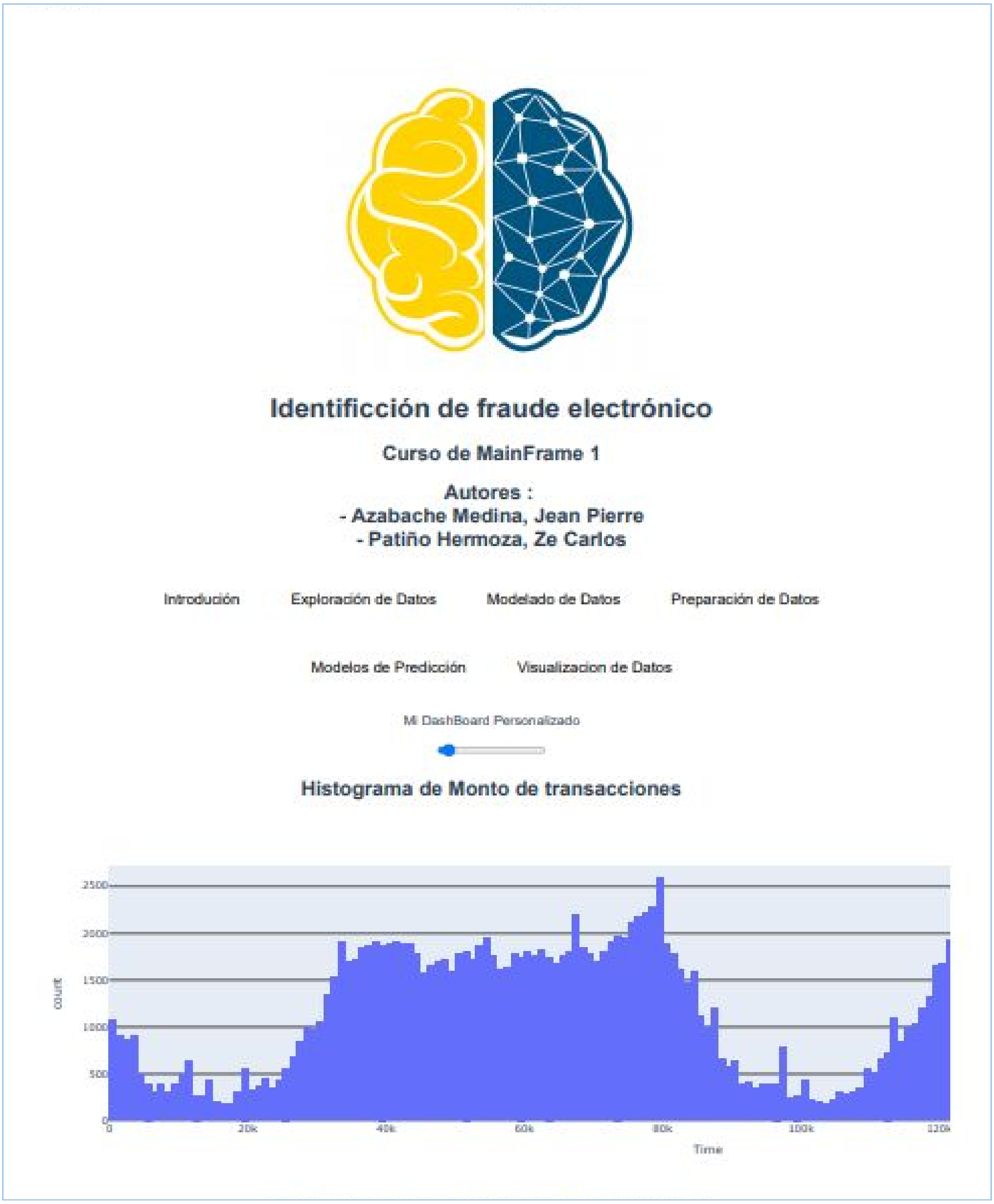
App.vue Introduccion.vue X

src > components > Introduccion.vue > {} "Introduccion.vue" > template > div.Indroduccion

```
<template>
  <div class="Indroduccion">
      <h1 :style="'font-size: ' + range + 'px'">{{msg}}</h1>
      <h1>Identificción de fraude electrónico</h1>
      <h2>Curso de MainFrame 1</h2>
      <h2>Autores : <br>
        - Azabache Medina, Jean Pierre<br>
        - Patiño Hermoza, Ze Carlos<br></h2>
      <p></p>
      <p></p>
    <button type="button" @click="tab=1">Introdución </button>
    <button type="button" @click="tab=2">Exploración de Datos </button>
     <button type="button" @click="tab=3">Modelado de Datos </button>
    <button type="button" @click="tab=4">Preparación de Datos </button>
    <button type="button" @click="tab=5">Modelos de Predicción</button>
    <button type="button" @click="tab=6">Visualizacion de Datos </button>
    <p>Mi DashBoard Personalizado</p>
    <input type="range" min="8" max="80" v-model="range">


  <div>

    <h2>Histograma de Monto de transacciones</h2>
    <Plotly v-if="tab==6" :data="grafico1.data" :layout="grafico1.layout" :display-mode-bar="true"></Plotly>
    <h2>Diagrama de caja de Monto de transacciones</h2>
    <Plotly v-if="tab==6" :data="grafico2.data" :layout="grafico2.layout" :display-mode-bar="true"></Plotly>
     <h2>Diagrama de barras de la Media entre V1 - V28</h2>
    <Plotly v-if="tab==6" :data="grafico3.data" :layout="grafico3.layout" :display-mode-bar="true"></Plotly>
    <h2>Diagrama de Barras de la desviación estandar entre V1 - V28</h2>
    <Plotly v-if="tab==6" :data="grafico5.data" :layout="grafico5.layout" :display-mode-bar="true"></Plotly>
    <h2>Diagrama de Barras de la simetría entre V1 - V28</h2>
    <Plotly v-if="tab==6" :data="grafico6.data" :layout="grafico6.layout" :display-mode-bar="true"></Plotly>
    <h2>Diagrama de Barras de la varialbe V8</h2>
    <Plotly v-if="tab==6" :data="grafico7.data" :layout="grafico7.layout" :display-mode-bar="true"></Plotly>
     <h2>Diagrama de Caja de la variable entre V8</h2>
    <Plotly v-if="tab==6" :data="grafico8.data" :layout="grafico8.layout" :display-mode-bar="true"></Plotly>
     <h2>Diagrama de Barras de la curtosis entre V1 - V28</h2>
    <Plotly v-if="tab==6" :data="grafico9.data" :layout="grafico9.layout" :display-mode-bar="true"></Plotly>
     <h2>Diagrama de Barras del rango intercuartil entre V1 - V28</h2>
    <Plotly v-if="tab==6" :data="grafico10.data" :layout="grafico10.layout" :display-mode-bar="true"></Plotly>
```

Lín. 19, col. 1 Espacios: 2 UTF-8 LF Vue

4.4 Web Service



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).