Aprendizaje Automático

Actividad 3: Detección de anomalías y técnicas de agrupamiento

Equipo 39

- Cortés Orozco, Wilfrido
- Pinales Ayala, Omar Isaí
- Villalpando Velázquez, Juan Carlos

Introducción

En la siguiente actividad se analizó un conjunto de datos que contiene varias operaciones realizadas con tarjetas de crédito y se les aplicaron dos distintas técnicas no supervisadas, una para la detección de anomalías y otra para realizar agrupaciones de los datos.

Librerías y Datos

Cargamos las librerías necesarias

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Leemos el dataset de entrenamiento

In [ ]:
```
datos = pd.read_csv("creditcardcsvpresent.csv")
datos.head()
```

Out[ ]:

	Merchant_id	Transaction date	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isFor
0	3160040998	NaN	100.0	3000.0	N	5	
1	3160040998	NaN	100.0	4300.0	N	5	

	Merchant_id	Transaction date	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isFor
2	3160041896	NaN	185.5	4823.0	Y	5	
3	3160141996	NaN	185.5	5008.5	Y	8	
4	3160241992	NaN	500.0	26000.0	N	0	

Análisis exploratorio

Evaluamos el estado general del dataset

In [ ]:
```
datos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3075 entries, 0 to 3074
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Merchant_id                     3075 non-null   int64  
 1   Transaction date                0 non-null      float64
 2   Average Amount/transaction/day  3075 non-null   float64
 3   Transaction_amount              3075 non-null   float64
 4   Is declined                     3075 non-null   object 
 5   Total Number of declines/day    3075 non-null   int64  
 6   isForeignTransaction            3075 non-null   object 
 7   isHighRiskCountry               3075 non-null   object 
 8   Daily_chargeback_avg_amt        3075 non-null   int64  
 9   6_month_avg_chbk_amt            3075 non-null   float64
 10  6-month_chbk_freq               3075 non-null   int64  
 11  isFradulent                     3075 non-null   object 
dtypes: float64(4), int64(4), object(4)
memory usage: 288.4+ KB
```

La columna **Transaction date** esta completamente vacía, decidimos eliminarla ya que no aporta nada al conjunto de datos

Eliminamos la columna Transaction date y creamos un nuevo dataframe para conservar el original

In [ ]:
```
datos_filtrados = datos.drop(["Transaction date"], axis=1)
datos_filtrados.shape
```

Out[ ]:
```
(3075, 11)
```

Nos quedamos con un dataframe de 11 columnas y 3075 registros.

Identificamos la cantidad y tipo de columnas que tenemos en el dataframe

In [ ]:
```
datos_filtrados.dtypes.value_counts()
```

Out[ ]:
```
int64      4
object     4
float64    3
dtype: int64
```

Análisis variables numéricas

Obtenemos datos estadísticos de las variables numéricas

In [ ]:
```
datos_filtrados.describe()
```

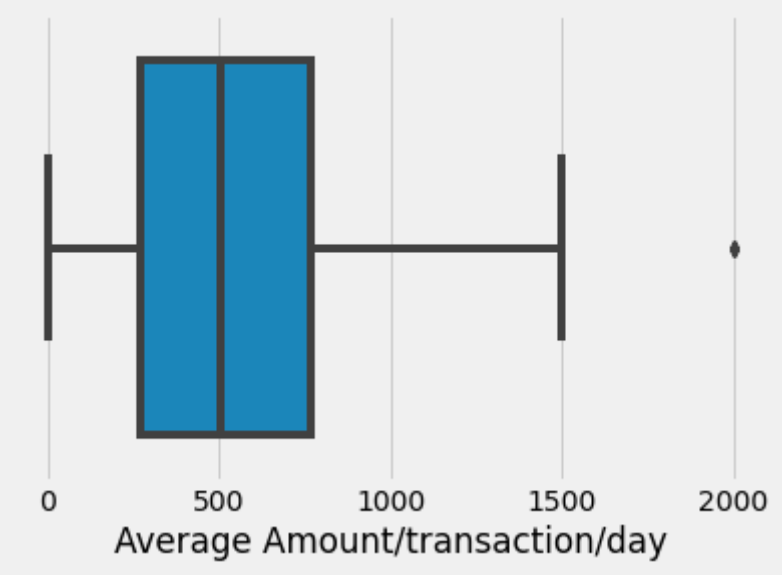
Out[ ]:

	Merchant_id	Average Amount/transaction/day	Transaction_amount	Total Number of declines/day	Daily_chargeback_avg_a
count	3.075000e+03	3075.000000	3075.000000	3075.000000	3075.000(
mean	5.026634e+09	515.026556	9876.399210	0.957398	55.737
std	9.870778e+08	291.906978	10135.331016	2.192391	206.634
min	3.160041e+09	4.011527	0.000000	0.000000	0.000(
25%	4.170814e+09	269.788047	2408.781147	0.000000	0.000(
50%	5.025578e+09	502.549575	6698.891856	0.000000	0.000(
75%	5.889625e+09	765.272803	14422.568935	0.000000	0.000(
max	6.665906e+09	2000.000000	108000.000000	20.000000	998.000(

Visualizamos la distribución de la columna Average_Amount/transaction/day

In [ ]:
```
sns.boxplot(datos_filtrados['Average Amount/transaction/day'])
```

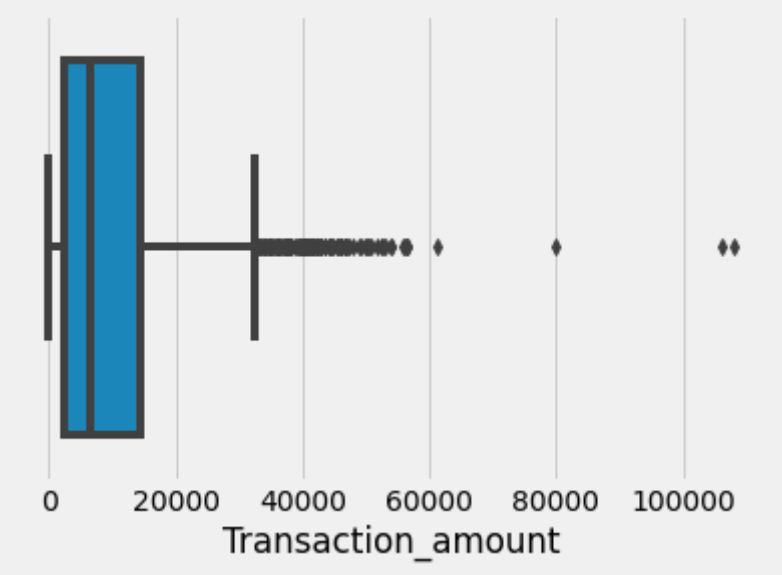
Out[ ]:
```
<AxesSubplot:xlabel='Average Amount/transaction/day'>
```



Visualizamos la distribución de la columna Transaction_amount

In [ ]:
```
sns.boxplot(datos_filtrados['Transaction_amount'])
```

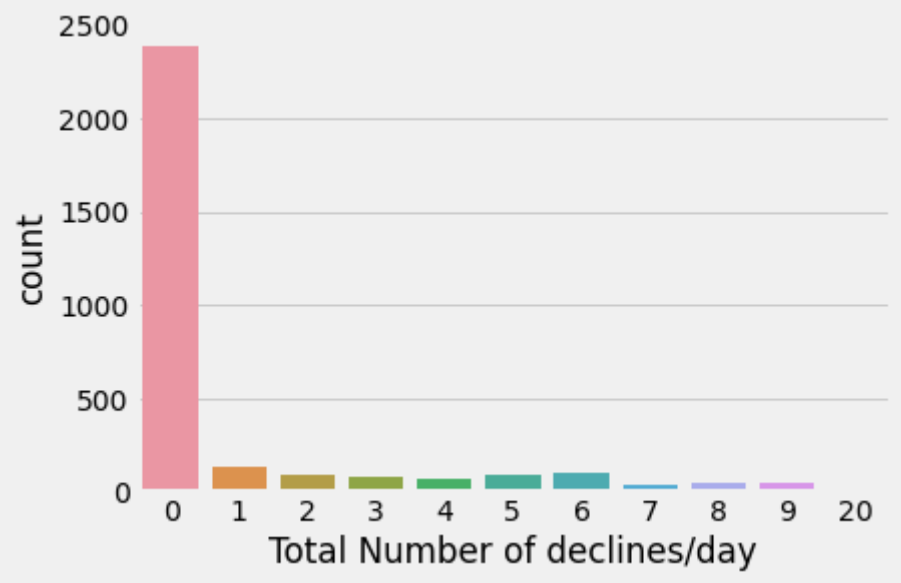
Out[ ]:
```
<AxesSubplot:xlabel='Transaction_amount'>
```



Visualizamos la distribución de la columna Total_Number_of_declines_day

In [ ]:
```
sns.countplot(datos_filtrados["Total Number of declines/day"])
```

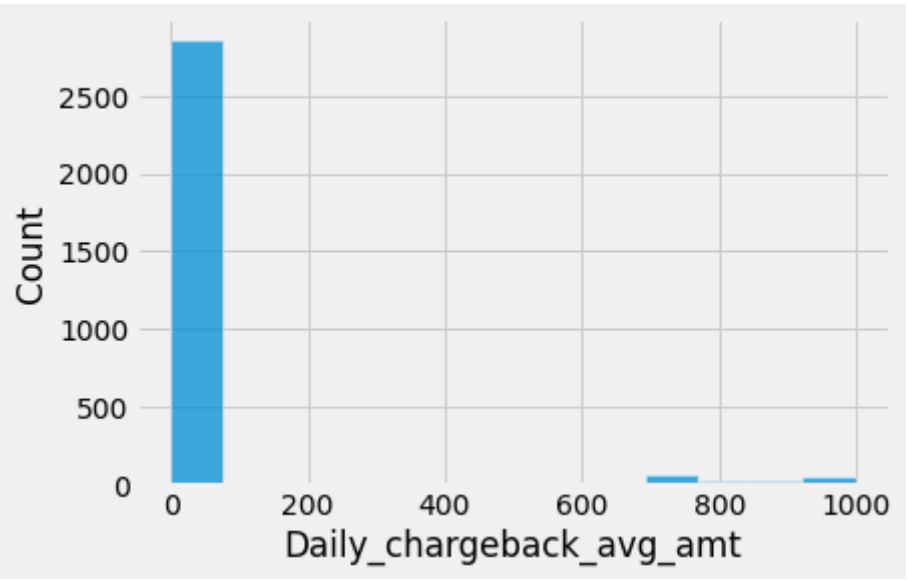
Out[ ]:
```
<AxesSubplot:xlabel='Total Number of declines/day', ylabel='count'>
```



Visualizamos la distribución de la columna Daily_chargeback_avg_amt

In [ ]:
```
sns.histplot(datos_filtrados["Daily_chargeback_avg_amt"])
```

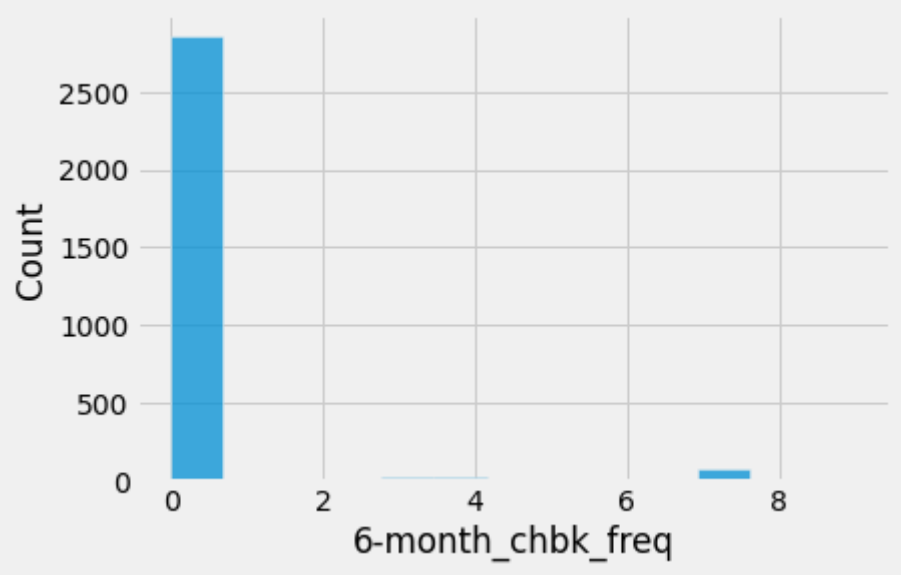
Out[ ]:
```
<AxesSubplot:xlabel='Daily_chargeback_avg_amt', ylabel='Count'>
```



Visualizamos la distribución de la columna Daily_chargeback_avg_amt

In [ ]:
```
sns.histplot(data=datos_filtrados, x="6-month_chbk_freq")
```

Out[ ]:
```
<AxesSubplot:xlabel='6-month_chbk_freq', ylabel='Count'>
```



Análisis de variables categóricas

Obtenemos las categorías de cada variable categórica y su frecuencia

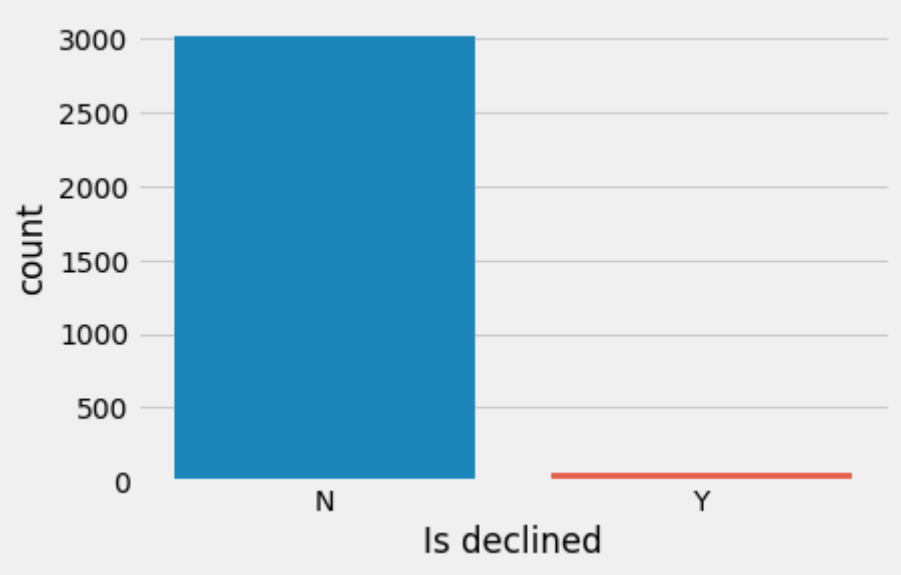
In [ ]:
```
datos_categoricos = datos_filtrados.select_dtypes(exclude=['int64', 'float64'])
for col in datos_categoricos.columns:
    print(datos_categoricos[col].unique())
    print(datos_categoricos[col].value_counts())
```

```
['N' 'Y']
N    3018
Y      57
Name: Is declined, dtype: int64
['Y' 'N']
N    2369
Y     706
Name: isForeignTransaction, dtype: int64
['Y' 'N']
N    2870
Y     205
Name: isHighRiskCountry, dtype: int64
['Y' 'N']
N    2627
Y     448
Name: isFradulent, dtype: int64
```

Visualizamos la distribución de la columna Is_declined

In [ ]:
```
sns.countplot(datos_filtrados["Is declined"])
```

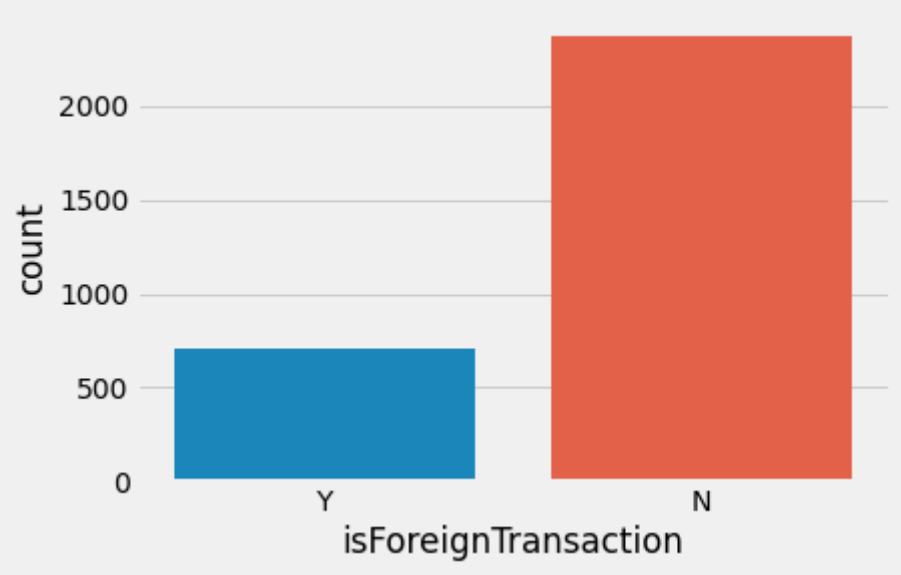
Out[ ]:
```
<AxesSubplot:xlabel='Is declined', ylabel='count'>
```



Visualizamos la distribución de la columna isForeignTransaction

In [ ]:
```
sns.countplot(datos_filtrados["isForeignTransaction"])
```

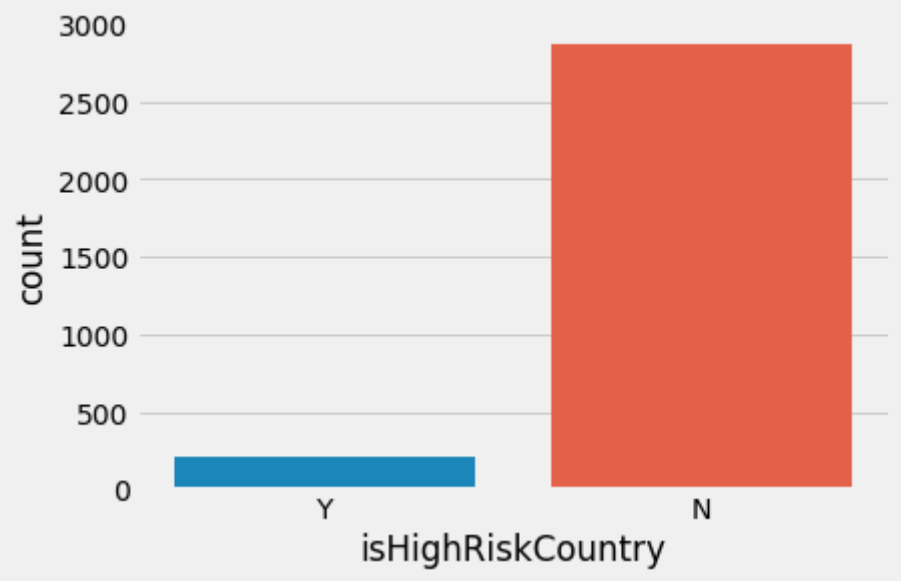
Out[ ]:
```
<AxesSubplot:xlabel='isForeignTransaction', ylabel='count'>
```



Visualizamos la distribución de la columna isHighRiskCountry

In [ ]:
```
sns.countplot(datos_filtrados["isHighRiskCountry"])
```

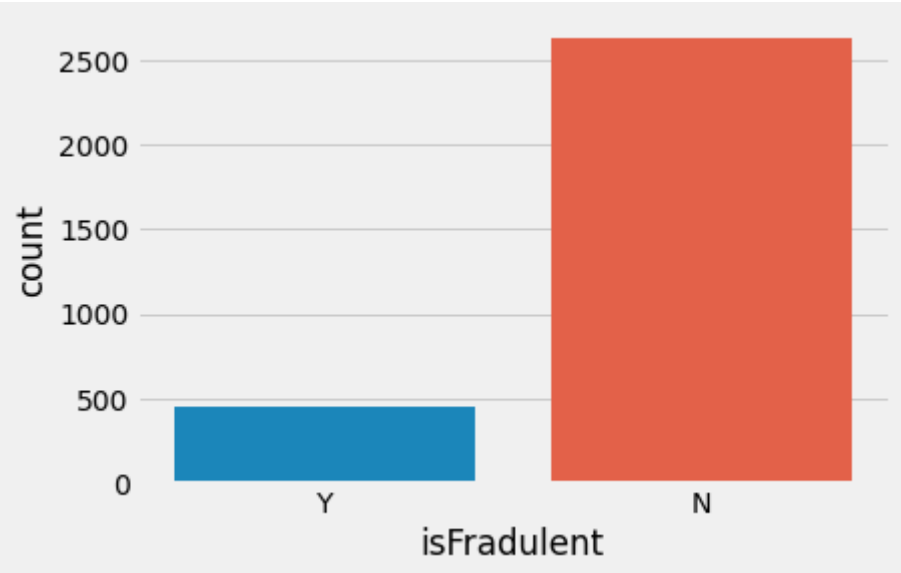
Out[ ]:
```
<AxesSubplot:xlabel='isHighRiskCountry', ylabel='count'>
```



Visualizamos la distribución de la columna isFradulent

In [ ]:
```
sns.countplot(datos_filtrados["isFradulent"])
```

Out[ ]:
```
<AxesSubplot:xlabel='isFradulent', ylabel='count'>
```



Análisis de correlación

Las columnas categóricas con valores Y/N las cambiamos a valores 1/0 para poder correlacionarlas:

In [ ]:
```
for column in ['Is declined', 'isForeignTransaction', 'isHighRiskCountry', 'isFradulent
  datos_filtrados[column].replace(to_replace=['Y', 'N'], value=[1, 0], inplace=True)
```

```
datos_filtrados.head()
```

Out[ ]:

	Merchant_id	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransacti
0	3160040998	100.0	3000.0	0	5	
1	3160040998	100.0	4300.0	0	5	
2	3160041896	185.5	4823.0	1	5	
3	3160141996	185.5	5008.5	1	8	
4	3160241992	500.0	26000.0	0	0	

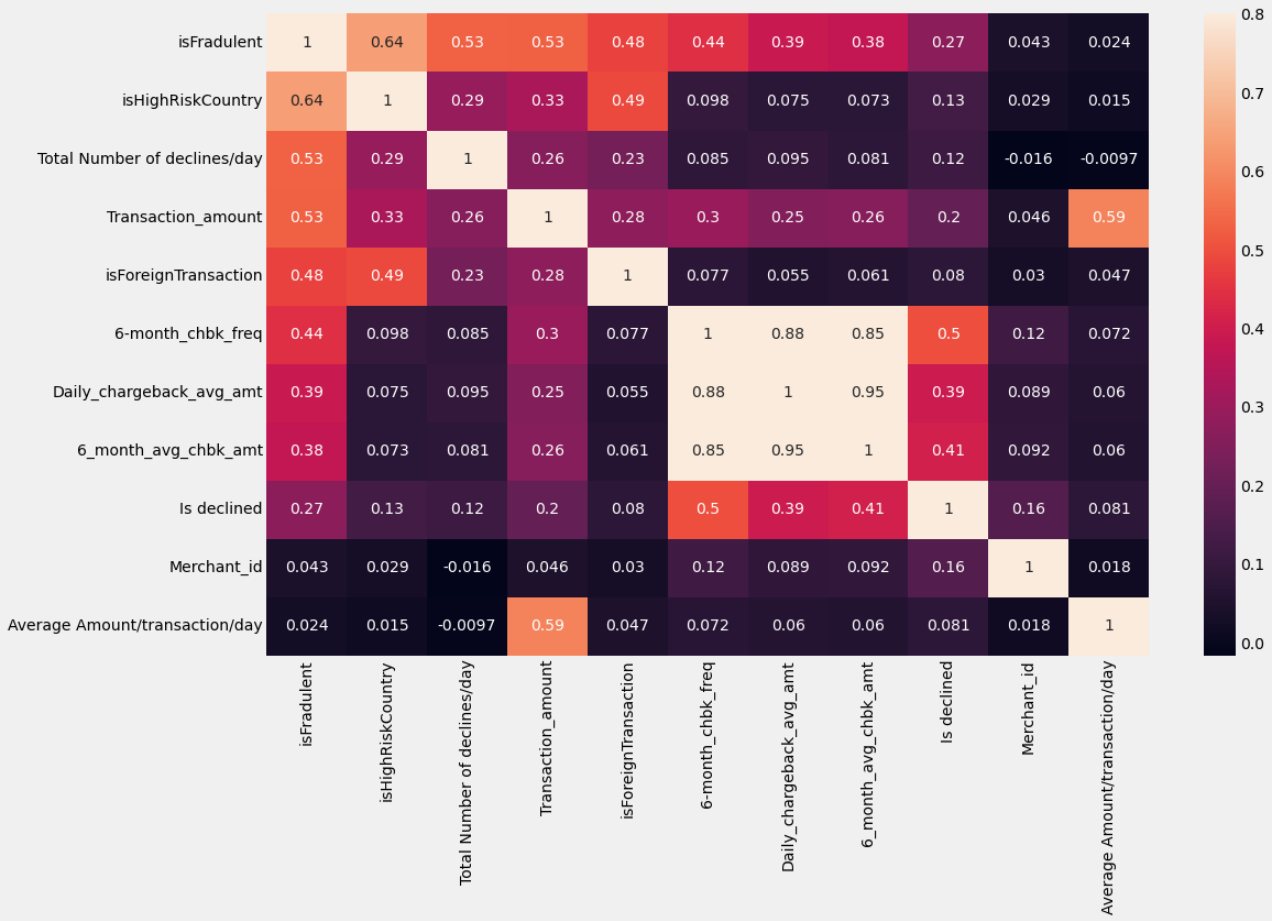
Obtenemos la correlación existente entre cada una de las variables

In [ ]:

```
matriz_correlacion = datos_filtrados.corr()
matriz_correlacion.sort_values('isFradulent', axis=0, ascending=False, inplace=True)
matriz_correlacion.sort_values('isFradulent', axis=1, ascending=False, inplace=True)
plt.figure(figsize=(16,10))
sns.heatmap(matriz_correlacion, vmax=.8, annot=True)
```

Out[ ]:

```
<AxesSubplot:>
```



Obtenemos los datos de trabajo removiendo la columna isFraudulent

In [ ]:
```
datos_de_trabajo = datos_filtrados.drop(['isFradulent'], axis=1)
datos_de_trabajo.head()
```

Out[ ]:

	Merchant_id	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransactio
0	3160040998	100.0	3000.0	0	5	
1	3160040998	100.0	4300.0	0	5	
2	3160041896	185.5	4823.0	1	5	
3	3160141996	185.5	5008.5	1	8	
4	3160241992	500.0	26000.0	0	0	

Detección de anomalías usando Isolation Forest

Creamos el modelo y lo ajustamos a los datos de trabajo:

In [ ]:
```
from sklearn.ensemble import IsolationForest
model = IsolationForest(n_estimators=100, max_samples='auto', random_state=9472)
model.fit(datos_de_trabajo)

print(model.get_params())
```

```
{'bootstrap': False, 'contamination': 'auto', 'max_features': 1.0, 'max_samples': 'aut
o', 'n_estimators': 100, 'n_jobs': None, 'random_state': 9472, 'verbose': 0, 'warm_star
t': False}
```

Obtenemos el score de anomalía de cada dato y cuales se clasificaron como anomalías:

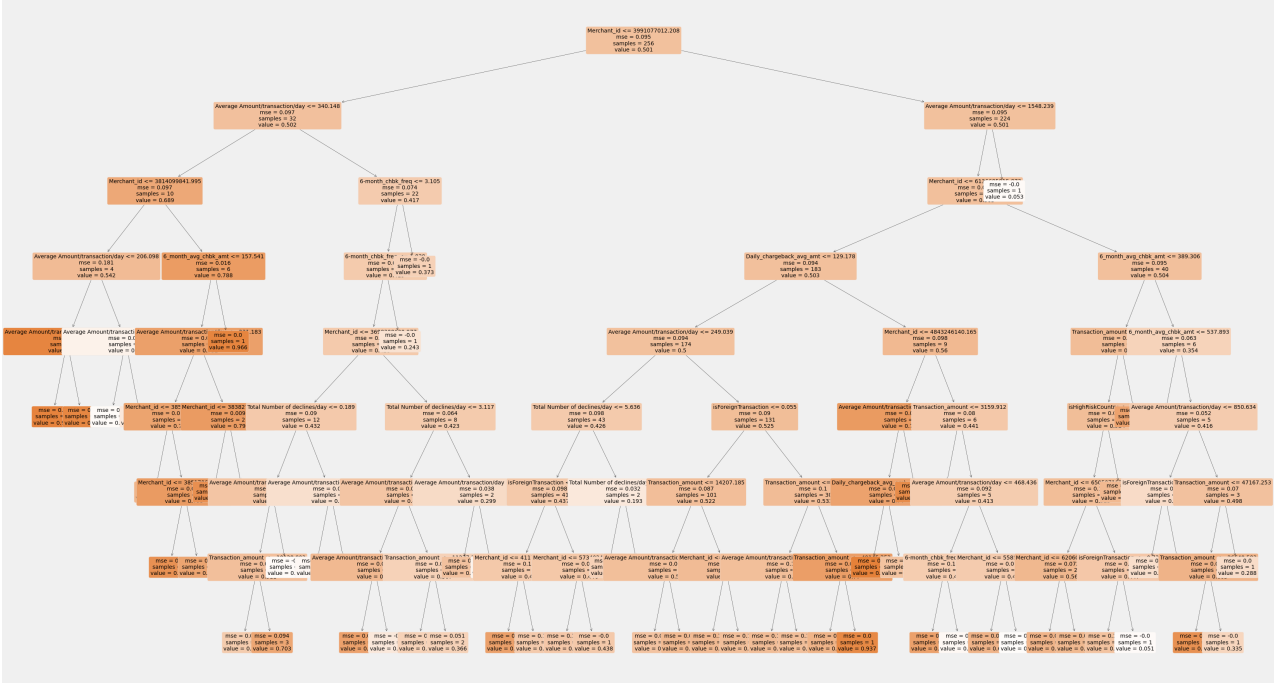
In [ ]:
```
datos_analizados = pd.DataFrame()

datos_analizados['anomalos'] = model.decision_function(datos_de_trabajo)
datos_analizados['anomaly_score'] = model.predict(datos_de_trabajo)
datos_analizados['score samples'] = model.score_samples(datos_de_trabajo)
datos_analizados.head()
```

Out[ ]:

	anomalos	anomaly_score	score samples
0	-0.124102	-1	-0.624102
1	-0.118971	-1	-0.618971
2	-0.120209	-1	-0.620209
3	-0.169662	-1	-0.669662
4	-0.208558	-1	-0.708558

Visualizamos el primer árbol para tener una ligera idea de como quedó el modelo:

In [ ]:
```
from sklearn.tree import plot_tree
arbol1 = model.estimators_[0]
plt.figure(figsize=(50,30))
a = plot_tree(arbol1, feature_names=datos_de_trabajo.columns.values.tolist(), filled=Tr
```



Contamos el número de datos clasificados como anomalías

In [ ]:
```
datos_analizados[datos_analizados['anomaly_score']==-1].shape
```

Out[ ]:
```
(523, 3)
```

Comparamos con los que son realmente anomalías

In [ ]:
```
datos_filtrados[datos_filtrados['isFradulent']==1].shape
```

Out[ ]:
```
(448, 11)
```

Binarizamos los valores para poder compararlos con los reales y evaluar la precisión

In [ ]:
```
datos_analizados.loc[datos_analizados['anomaly_score'] == 1, 'isFradulent'] = 0
datos_analizados.loc[datos_analizados['anomaly_score'] == -1, 'isFradulent'] = 1
datos_analizados.head()
```

Out[ ]:

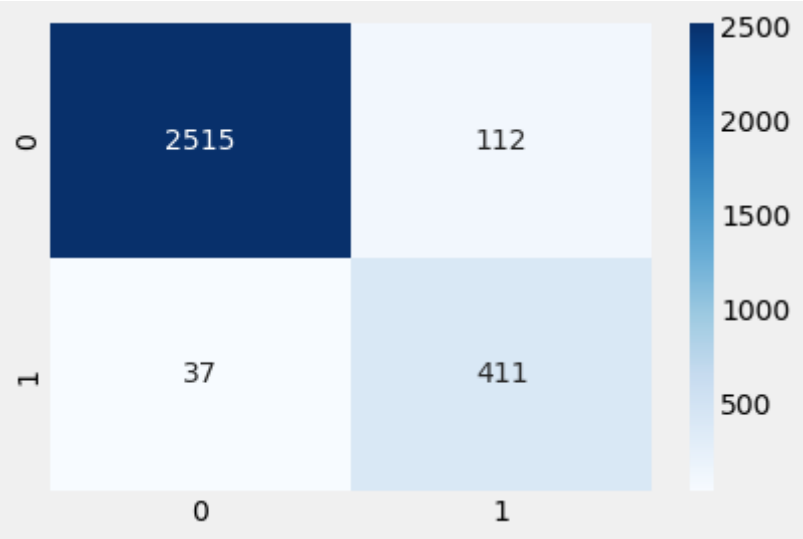
	anomalos	anomaly_score	score samples	isFradulent
0	-0.124102	-1	-0.624102	1.0
1	-0.118971	-1	-0.618971	1.0
2	-0.120209	-1	-0.620209	1.0
3	-0.169662	-1	-0.669662	1.0

	anomalos	anomaly_score	score samples	isFradulent
4	-0.208558	-1	-0.708558	1.0

Obtenemos la matriz de confusión para darnos una idea de la efectividad del modelo

Obtenemos que 36 operaciones fueron evaluados como fraudulentos cuando no lo son, y 119 operaciones fraudulentas no fueron clasificadas como fraudulentas.

In [ ]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(datos_filtrados['isFradulent'], datos_analizados['isFradulent'])
ax=sns.heatmap(cm, cmap="Blues", annot=True, fmt='d')
```



Ventajas y desventajas Isolation Forest

Ventajas

- Su implementación es sencilla.
- Puede generar buenos resultados sin necesidad de muchos ajustes manuales.

Desventajas

- Requiere de configuraciones inciales para determinar el número de árboles y opcionalmente el número de muestras.
- Compleja visualización del modelo como un todo debido a que son múltiples árboles.

K-MEAN con Revisión por Coeficiente de Silueta

Obtención de k-means con 2 clúster, más adelante se comprueba si es la mejor decisión

In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters=2, random_state=42).fit(datos_de_trabajo)
kmeans.labels_
```

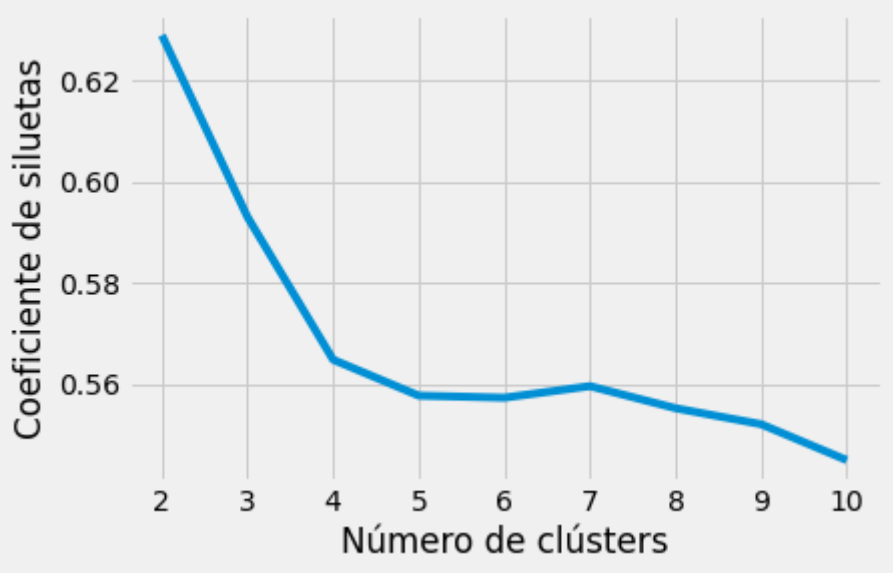
Out[ ]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [ ]:
```
silhouette_coefficients = []
for i in range(2,11):
    km = KMeans(n_clusters = i, random_state=42)
    km.fit_predict(datos_de_trabajo)
    score = silhouette_score(datos_de_trabajo, km.labels_, metric='euclidean')
    print('Número de clusters ',i,'Silhouette Score: %.3f' % score)
    silhouette_coefficients.append(score)
```

```
Número de clusters  2 Silhouette Score: 0.629
Número de clusters  3 Silhouette Score: 0.593
Número de clusters  4 Silhouette Score: 0.565
Número de clusters  5 Silhouette Score: 0.558
Número de clusters  6 Silhouette Score: 0.557
Número de clusters  7 Silhouette Score: 0.560
Número de clusters  8 Silhouette Score: 0.555
Número de clusters  9 Silhouette Score: 0.552
Número de clusters  10 Silhouette Score: 0.545
```

In [ ]:
```
plt.style.use("fivethirtyeight")
plt.plot(range(2,11), silhouette_coefficients)
plt.xticks(range(2,11))
plt.xlabel("Número de clústers")
plt.ylabel("Coeficiente de siluetas")
plt.show
```

Out[ ]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



De acuerdo a la gráfica, lo más óptimo es utilizar 2 clúster, más clúster obtienen peor rendimiento.

Se preparan los datos para su visualización.

In [ ]:
```
# Etiquetas a utilizar
etiquetas = pd.DataFrame(kmeans.labels_)
etiquetasUnidas = pd.concat((datos_de_trabajo,etiquetas),axis=1)
etiquetasUnidas = etiquetasUnidas.rename({0:'etiquetas'},axis=1)
```

In [ ]:
```
etiquetasUnidas.head()
```

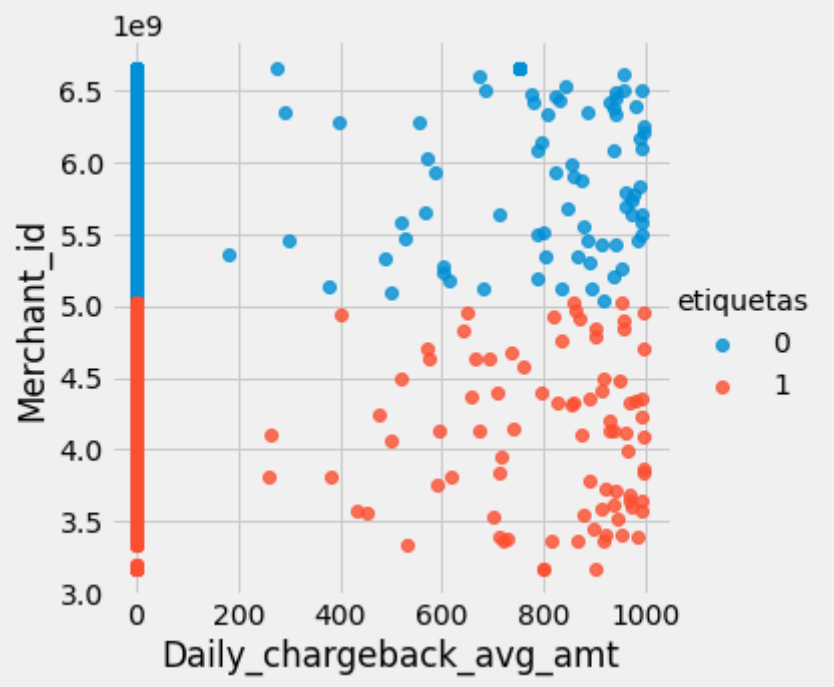
Out[ ]:

	Merchant_id	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransacti
0	3160040998	100.0	3000.0	0	5	
1	3160040998	100.0	4300.0	0	5	
2	3160041896	185.5	4823.0	1	5	
3	3160141996	185.5	5008.5	1	8	
4	3160241992	500.0	26000.0	0	0	

Se visualiza un par de variables únicamente, no hay una visualización "evidente" debido a la cantidad de dimensiones que se obtienen por la cantidad de variables.

In [ ]:
```
sns.lmplot(x='Daily_chargeback_avg_amt',y='Merchant_id',data=etiquetasUnidas,hue='etiqu
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1ab81b65ee0>



In [ ]:
```
datos_de_trabajo
```

Out[ ]:

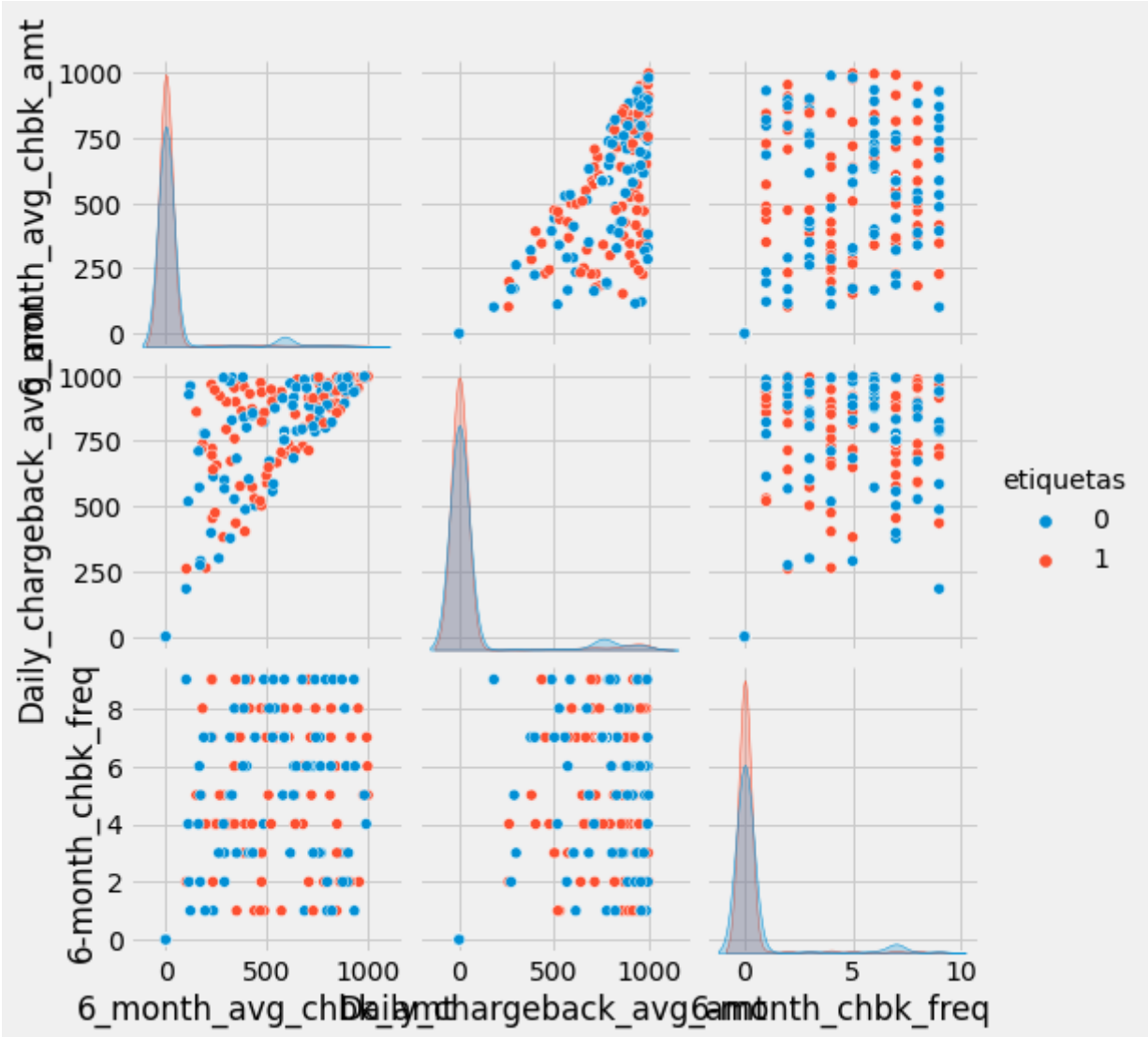
	Merchant_id	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransa
0	3160040998	100.0	3000.0	0	5	
1	3160040998	100.0	4300.0	0	5	
2	3160041896	185.5	4823.0	1	5	
3	3160141996	185.5	5008.5	1	8	
4	3160241992	500.0	26000.0	0	0	
...	...	...	...	...	...	
3070	6661273532	500.0	11000.0	1	0	
3071	6661273532	800.0	0.0	1	0	
3072	6661273533	800.0	20800.0	1	0	
3073	6661273532	1500.0	12000.0	1	0	
3074	6661273533	1500.0	36000.0	1	0	

3075 rows × 10 columns

Se muestra una tabla con todas las visualizaciones 1 contra 1 posibles.

In [ ]:
```
# Etiquetas a utilizar
etiquetas = pd.DataFrame(kmeans.labels_)
etiquetasUnidasPlot = pd.concat((datos_de_trabajo[['6_month_avg_chbk_amt','Daily_charge
etiquetasUnidasPlot = etiquetasUnidasPlot.rename({0:'etiquetas'},axis=1)

sns.pairplot(etiquetasUnidasPlot,hue='etiquetas')
```

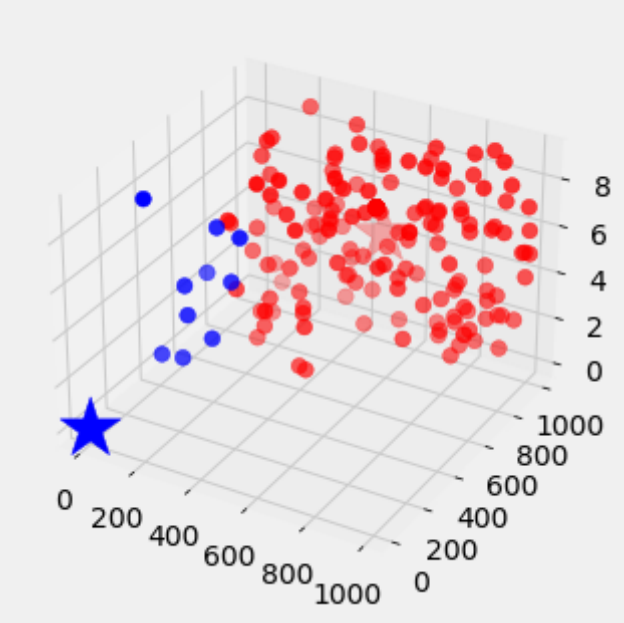
Out[ ]: <seaborn.axisgrid.PairGrid at 0x1ab81b65a30>



In [ ]:
```
# Atributos a tener en cuenta en base a la matriz de correlaciones:

# 6_month_avg_chbk_amt
# Daily_chargeback_avg_amt
```

```
# 6-month_chbk_freq
from mpl_toolkits.mplot3d import Axes3D

temp_X = np.array(datos_de_trabajo[['6_month_avg_chbk_amt','Daily_chargeback_avg_amt','

kmeans_temp = KMeans(n_clusters=2, random_state=42).fit(temp_X)
etiquetasY = kmeans_temp.predict(temp_X)
centros = kmeans_temp.cluster_centers_
colores = ["blue",'red']
asignar = []
for row in etiquetasY:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(temp_X[:, 0], temp_X[:, 1], temp_X[:, 2], c=asignar,s=60)
ax.scatter(centros[:, 0], centros[:, 1], centros[:, 2], marker='*', c=colores, s=1000)
```

Out[ ]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1ab836190d0>



Ventajas y desventajas K-mean

El método de K-mean es computacionalmente costoso por la cantidad de iteraciones que tiene que realizar para lograr converger, pero es poderoso para agrupar en base a distancias.

Ventajas

- El algoritmo puede trabajar sin restricciones de dimensiones.
- Se puede evaluar con mecanismos relativamente simples.
- Garantiza la convergencia.

- Su implementación es sencilla.
- Puede enfrentarse a set de datos grandes.

Desventajas

- Requiere de un dato inicial que determinará todo el proceso (el número de clústers)
- Puede converger a óptimos locales, lo que conlleva que en ciertos casos sea inferior a otros métodos como EM (Expectation-Maximization).
- El agrupamiento se puede ver afectado por las anomalías.
- Depende considerablemente de los valores iniciales.

Referencias

Google. (2021). k-Means Advantages and Disadvantages . octubre 11, 2021, de Google Sitio web: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages

Else-if. (2018). Visualizing Multidimensional Clusters. octubre 11, 2021, de Kaggle Sitio web: https://www.kaggle.com/ellecf/visualizing-multidimensional-clusters

Bagnato, J.. (2018). K-Means en Python paso a paso. octubre 11, 2021, de aprendemachinelearning.com Sitio web: https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/