Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Actividad grupal y foro. Detección de anomalías y técnicas de agrupamiento

Introducción

la detección de anomalías y las técnicas de agrupamiento son dos áreas importantes del aprendizaje automático y la minería de datos.

La detección de anomalías se refiere a la identificación de observaciones o eventos que se desvían significativamente de la norma o patrón esperado. Es decir, se trata de identificar datos que son raros o inusuales en un conjunto de datos. Esto puede ser útil en una variedad de aplicaciones, como la detección de fraudes en transacciones financieras o la identificación de fallas en equipos mecánicos.

Existen varias técnicas para la detección de anomalías, como la detección basada en reglas, la detección basada en estadísticas y la detección basada en aprendizaje automático. El aprendizaje automático se ha convertido en una técnica popular para la detección de anomalías, ya que permite identificar patrones complejos y no lineales en los datos.

Por otro lado, las técnicas de agrupamiento se refieren a la clasificación de un conjunto de datos en grupos o clúster, donde cada grupo contiene observaciones que son similares entre sí. Esto puede ser útil para identificar patrones en los datos y para la segmentación de clientes en marketing. Existen varias técnicas de agrupamiento, como el k-means, el clustering jerárquico y el clustering espectral. Cada técnica tiene sus propias ventajas y desventajas, y la elección de una técnica depende del conjunto de datos y del problema que se está abordando. Algunos de estos ejemplos los presentamos en la siguiente actividad.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Limpieza y Procesado de Datos

La primera etapa consiste en analizar los datos e importar las librerías necesarias, de esta manera identificar los datos categóricos y de ser necesario normalizarlos, de esa manera facilitar las operaciones y análisis en etapas futuras, como primera impresión, notamos valores alfanuméricos representando campos boléanos, por lo que una etapa de limpieza y preparación fue necesaria.

```
#importacón de las librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler,LabelEncoder,normalize
from sklearn.cluster import KMeans
```

```
#Conociendo las variables a trabajar
df = pd.read_csv("creditcardcsvpresent.csv")
df.head()
```

	Merchant_id	Transaction date	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransaction	isHighRiskCountry	Daily_chargeback_avg_amt	6_month_avg_chbk_amt	6 month_chbk_fre
0	3160040998	NaN	100.0	3000.0	N	5	Y	Y	0	0.0	
1	3160040998	NaN	100.0	4300.0	N	5	Y	Y	0	0.0	
2	3160041896	NaN	185.5	4823.0	Y	5	N	N	0	0.0	
3	3160141996	NaN	185.5	5008.5	Y	8	N	N	0	0.0	
4	3160241992	NaN	500.0	26000.0	N	0	Y	Y	800	677.2	

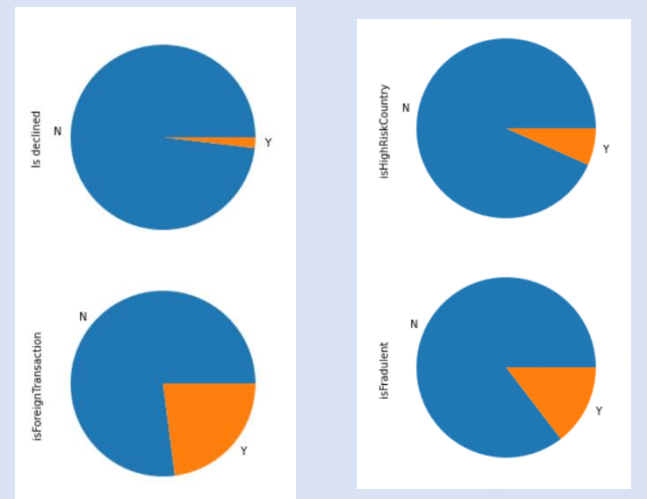
Para conocer las variables de una mejor manera, se uso la función info(), la cual nos permite ver el tipo de dato con el que estaremos trabajando para cada variable

In [4]:
```
#Visualización de los datos completos
df.info()
```

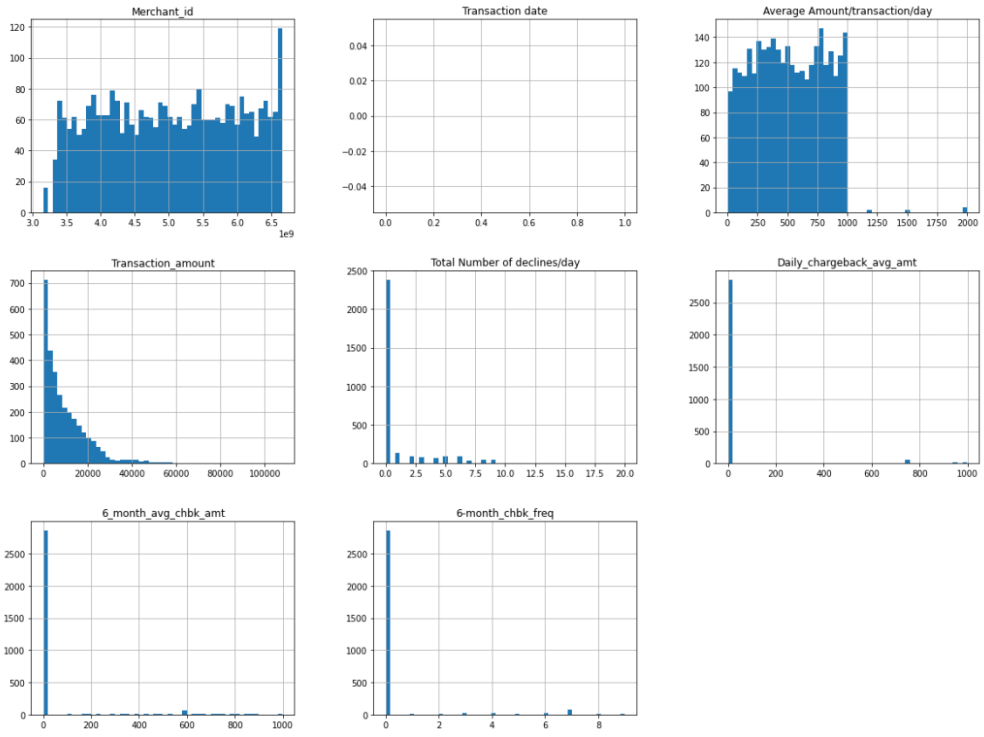
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3075 entries, 0 to 3074
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Merchant_id                     3075 non-null   int64
 1   Transaction 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
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Para analizar los datos categóricos, se realizaron unas graficas de pastel, para observar que tipos de categorías manejan y el porcentaje que representan cada dato.



Al mismo tiempo usamos un análisis sobre todas las variables numéricas, esto con el objetivo de conocer sus frecuencias, variaciones, medias y demás métricas, todo esto con el fin de identificar cuáles de las columnas representan mayor relevancia para nuestro objetivo el cual es identificar las transacciones fraudulentas en este set de datos, siendo estos campos para nosotros nuestras variables categóricas.



Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Se realizo un análisis para la identificación de datos nulos, mediante la función is null(), la cual nos regresa todos los datos vacíos del dataset y mediante la función. Sum(), se realizo para sumas la cantidad de valores nulos.

```
In [9]:
#Visualización de datos nulos
df.isnull().sum()
```

```
Out[9]:
Merchant_id                        0
Transaction date                3075
Average Amount/transaction/day     0
Transaction_amount                 0
Is declined                        0
Total Number of declines/day       0
isForeignTransaction               0
isHighRiskCountry                  0
Daily_chargeback_avg_amt           0
6_month_avg_chbk_amt               0
6-month_chbk_freq                  0
isFradulent                        0
dtype: int64
```

Observamos que la variable 'Transaction date' presenta todos los valores nulos por lo que se decidió eliminar, con la función drop() al data set con el cual se estará trabajando.

```
In [10]:
#Eliminación de la variable Transaction Date
df.drop(["Transaction date"],axis=1, inplace=True)
```

Se realizo un método para cambiar todas las variables categóricas (object) y transformarlas a numéricas, mediante la librería labelencoder.

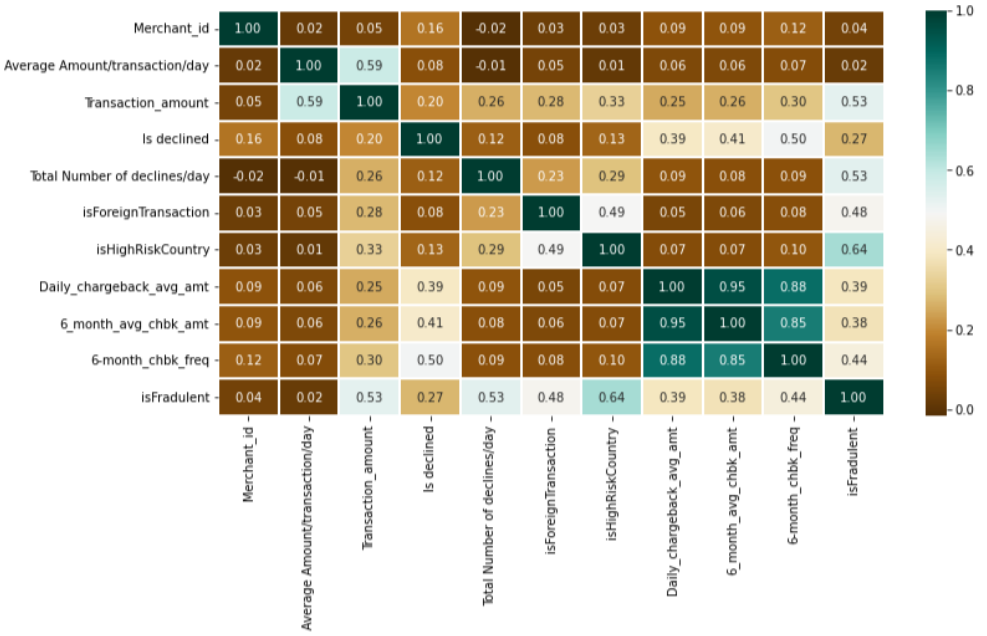
```
In [11]:
# Metodo para cambiar las variables categoricas a numericas
dsEtiquetas=df.select_dtypes(include=['object'])
le=LabelEncoder()
for c in dsEtiquetas.columns:
    le.fit(df[c].values)
    df[c] = le.transform(df[c].values)
```

Una vez aplicado el método para cambiar las variables categóricas a numéricas, se realizado una matriz de correlación para identificar que variables tienen mas peso sobre nuestra variable objetivo.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

```
In [13]:
#matriz de correlación de los datos seleccionados
plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)
```

```
Out[13]: <AxesSubplot:>
```

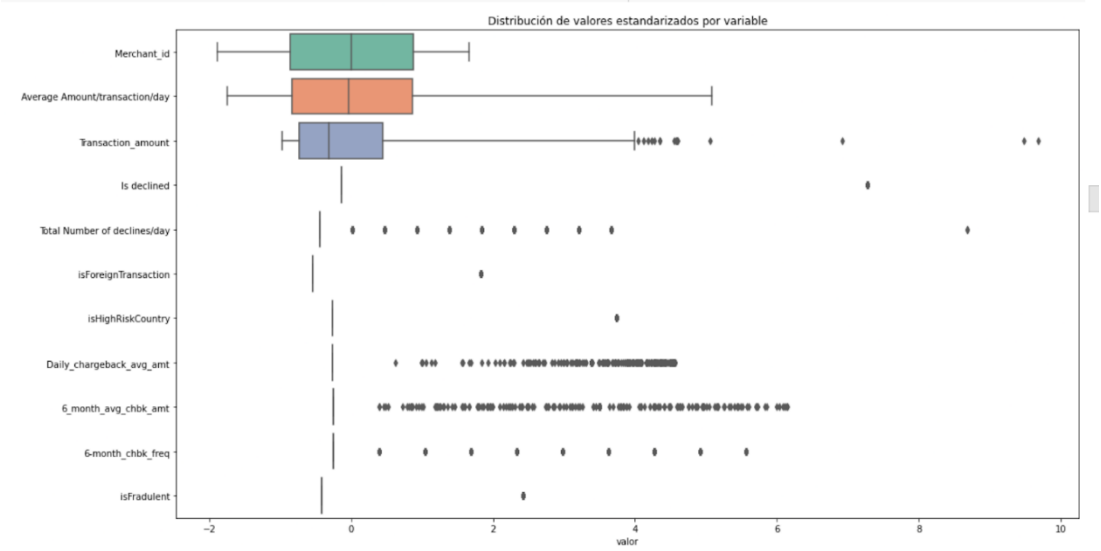


Un análisis superficial de los datos muestra que al menos 2 variables tienen una correlación dudosa con nuestra variable objetivo, al analizar estos campos más a detalle descubrimos que el análisis inicial tiene un sentido un poco más claro ya que las variables que no comparten relación son una que tiene la función de identificador y una columna representada por información nula.

Posteriormente, se realizó una conversión de escalamiento de los datos, esto para tener todos los datos en una misma escala, ya que este tipo de modelos funciona mejor con los datos ya escalados y dejados de una manera más tratable.

```
In [14]:
#Scalamiento de los datos
scalar = StandardScaler()
dataScalar = pd.DataFrame(scalar.fit_transform(df),
                          columns = df.columns)
```

Se realizo un diagrama de caja-bigote, con los datos ya escalados, esto para darnos una idea visualmente de que posibles anomalías se pueden presentar para cada variable.



Observamos que la variable 'transacion amount' presenta muchas anomalías o irregularidades, esto ya que presenta puntos por fuera de los vigotes y mientras que las demás variables, la mayoría las identifica como daros irregulares ya que no puede formar la caja.

Una vez identificado todos estos datos, se procedió con la creación de los algoritmos para outliers y clustering respectivos.

Técnica de IsoletionForest para detección de anomalías

De acuerdo al autor Joaquín Amat Rodrigo (Joaquín Amat Rodrigo, 2015) el algoritmo Isolation Forest es una técnica de detección de anomalías utilizada en el aprendizaje automático y la minería de datos. Su objetivo es identificar observaciones anómalas en un conjunto de datos mediante la construcción de árboles de decisión aleatorios.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

En el algoritmo Isolation Forest, cada árbol se construye seleccionando aleatoriamente una característica y un valor de umbral para dividir el conjunto de datos en dos partes. Este proceso se repite hasta que cada observación queda aislada en su propia hoja del árbol. La puntuación de anormalidad de cada observación se calcula como la profundidad promedio de los árboles en los que aparece la observación.

Tiene varias ventajas, como su eficiencia computacional y su capacidad para manejar grandes conjuntos de datos con características de alta dimensión. Además, es robusto a los valores atípicos y puede identificar observaciones anómalas tanto en datos unimodales como en datos multimodales.

Se utiliza en una variedad de aplicaciones, como la detección de fraude en transacciones financieras, la detección de intrusiones en sistemas de seguridad y la identificación de enfermedades en datos médicos. También se utiliza en el análisis de datos de series temporales y en la clasificación de imágenes.

Es importante tener en cuenta que, aunque el algoritmo Isolation Forest es eficaz en la detección de anomalías, no es adecuado para la identificación de las causas subyacentes de las anomalías. Por lo tanto, es necesario realizar un análisis más profundo para comprender las causas de las anomalías y tomar medidas adecuadas.

Lo primero que realizamos para crear nuestro algoritmo, fue crear nuestro set de datos con las variables a trabajr y aplicamos aprendizaje no supervisado, es decir no incluimos en nuestro data, la viarble ‘is fradulent’.

In [25]:
```
from sklearn.ensemble import IsolationForest

data2 = dataScalar[['isHighRiskCountry',
          'Total Number of declines/day',
          'Transaction_amount',
          'isForeignTransaction',
          '6-month_chbk_freq',
          'Daily_chargeback_avg_amt',
          'Is declined'
]]
```

Una vez realizado esto, se creo nuestro modelo y se ajustó, mediante la función IsolationForest(), estuvimos probando y decidimos usar una contaminación de 0.14 por que con esto nuestro modelo reacciono de una mejor manera.

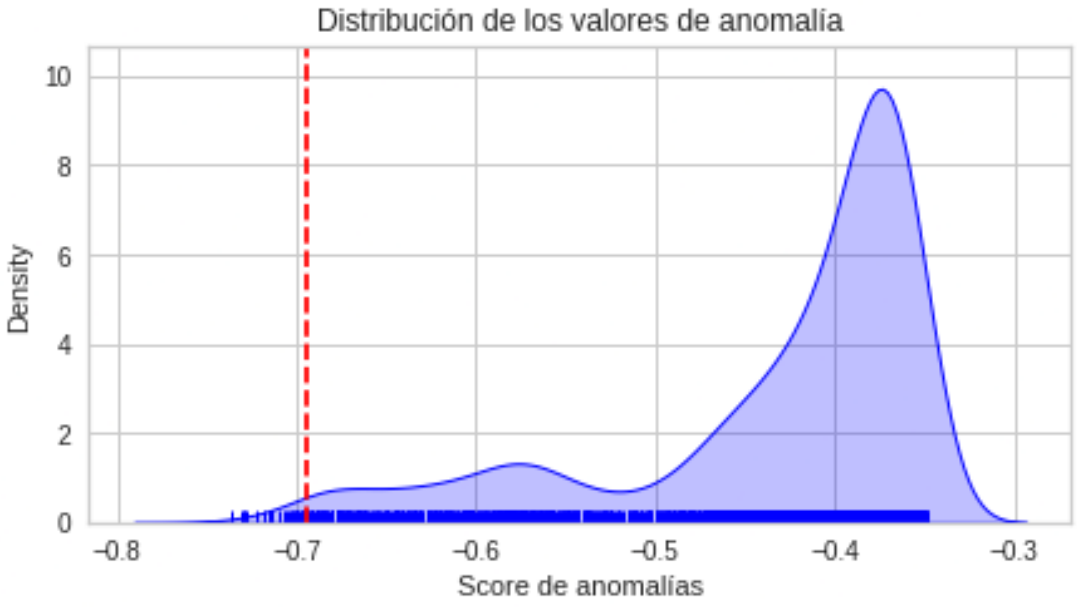
```
11 modelo_isof = IsolationForest(n_estimators = 1500, max_samples = 'auto', n_jobs = -1, contamination = float(0.14), max_featu
12 modelo_isof.fit(data2)
```

```
C:\Users\jcarl\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but Isolation
Forest was fitted with feature names
  warnings.warn(
```

Para conseguir una predicción de detección de anomalías usamos el modelo modelo_isof entrenado previamente en los datos data2, este modelo asigna una etiqueta de clase a cada ejemplo de data2 según si se considera que es un valor normal, usando esta clasificación, la información se considerara como anómala si su score de anomalía es menor que el cuantil 0.01 de la distribución de scores de anomalía.

```
1 clf_anomalia = modelo_isof.predict(data2)
2 clf_anomalia
```

```
array([ 1,  1,  1, ...,  1, -1, -1])
```



La línea vertical de color rojo representa el cuantil del 0.01 de la distribución de puntuaciones de anomalía utilizando el método axvline del eje de subplot ax.

Para realizar predicciones de futuras anomalías, se utilizó la función decision_function() y posteriormente, se realizo una muestra de los resultados, los datos con valor -1 los identifica como posibles anomalías o errores.

```
In [29]:
# Predecir las anomalías en los datos
anomaly_scores = modelo_isof.decision_function(data2)


# Mostrar los resultados de la detección de anomalías
anomaly_results = pd.DataFrame({'anomaly_score': anomaly_scores, 'is_anomaly': clf_anomalia})
print(anomaly_results)
```

```
      anomaly_score  is_anomaly
0         -0.036363          -1
1         -0.031995          -1
2         -0.037389          -1
3         -0.087381          -1
4         -0.118175          -1
...             ...         ...
3070      -0.109968          -1
3071      -0.118722          -1
3072      -0.111952          -1
3073      -0.158960          -1
3074      -0.162569          -1

[3075 rows x 2 columns]
```

Después de realizar el paso anterior, se calculo el accuracy del modelo, para realizar esto recurrimos a la etiqueta original del data set que contiene el valor de la variable objetivo y la comparamos contra las anomalías que detecto el modelo realizado.

```
In [31]:
#Se realiza un conteo de los ourliers del DataFrame original
outliers_counter = len(df[df['isFradulent'] == 1])
outliers_counter
```

```
Out[31]: 448
```

```
In [32]:
#Se calcula un accuracy comparando lo detectado por el modelo vs la etiqueta original
print("Accuracy percentage:", 100*list(anomaly_results['is_anomaly']).count(-1)/(outliers_counter))
```

```
Accuracy percentage: 96.20535714285714
```

Como se observa, el modelo obtuvo un acurracy de un 96.2 %, lo cual es bastante bueno, pues esto indica que detecto el mismo número de variables atípicas.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Técnica de Clustering K-Means

De acuerdo al sitio web www.aprendemachinelearning.com (K-Means En Python Paso a Paso, 2018) el algoritmo K-means es una técnica popular de agrupamiento utilizada en el aprendizaje automático y la minería de datos. Su objetivo es clasificar un conjunto de datos en k-clústers o grupos, donde cada grupo contiene observaciones que son similares entre sí.

Comienza seleccionando k centroides aleatorios, que representan los centros iniciales de los k-clústers. Luego, el algoritmo asigna cada observación del conjunto de datos al clúster cuyo centroide está más cercano a esa observación. A continuación, el algoritmo recalcula los centroides de cada clúster como la media de as observaciones asignadas a ese clúster. Este proceso se repite hasta que los centroides de los k-clústers convergen, es decir, ya no cambian.

El algoritmo K-means tiene varias ventajas como su simplicidad y eficiencia computacional en comparación con otros algoritmos de agrupamiento. Además, puede manejar grandes conjuntos de datos y es fácil de implementar en diferentes lenguajes de programación.

Así mismo, se utiliza en una variedad de aplicaciones, como la segmentación de clientes en marketing, la clasificación de imágenes y la detección de fraudes en transacciones financieras. En el campo de la bioinformática, también se utiliza para agrupar genes y proteínas similares en conjuntos de datos de expresión génica.

Lo primero que se realizo en el algoritmo fue crear, las posiciones de los clusters de forma aleatoria y se eliminaron las 3 filas del detaset con el que se trabajó en la practita.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

In [16]:
```
#Selección de datos al azar
#para posteriormente verificar al cruster al que pertenecen
data = dataScalar.copy()
indices = [340, 2905,1060]
muestras = pd.DataFrame(data.loc[indices],columns=data.keys()).reset_index(drop=True)
```

In [17]:
```
#Eliminamos las 3 filas del conjunto de datos
X = data.drop(indices,axis=0)
```

Una vez realizado, se procedio a usar el método del codo para determinar el valor optimo de K, a continuación, se da una breve explicación de este método.

Aplicación del Método del Codo.

De acuerdo al autor Juan Carlos Tovar (Método Elbow Y Método de Siluetas Para La Determinación Del Número de Clústeres En K-Means - Comunidad Huawei Enterprise, 2023) el método del codo (Elbow) es una representación gráfica de cómo encontrar la 'K' óptima en un clúster de K-means. Funciona encontrando WCSS (Suma de cuadrados dentro del grupo - Within-Cluster Sums of Squares), es decir, la suma de la distancia al cuadrado entre los puntos de un grupo y el centroide del grupo.

Para aplicar este método, fue necesaria la instalación de la librería yellowbrick.

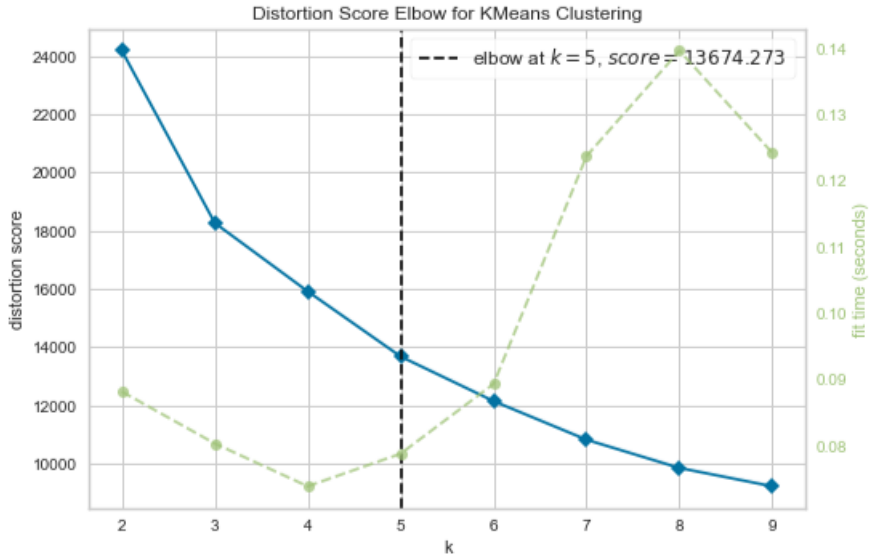
In [18]:
```
#Instalar yellowbrick para visualizar la curva del codo
!pip install yellowbrick
```

```
Requirement already satisfied: yellowbrick in c:\users\jcarl\anaconda3\lib\site-packages (1.5)
Requirement already satisfied: cycler>=0.10.0 in c:\users\jcarl\anaconda3\lib\site-packages (from yellowbrick) (0.11.0)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\jcarl\anaconda3\lib\site-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: numpy>=1.16.0 in c:\users\jcarl\anaconda3\lib\site-packages (from yellowbrick) (1.21.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\jcarl\anaconda3\lib\site-packages (from yellowbrick) (3.5.
1)
Requirement already satisfied: scipy>=1.0.0 in c:\users\jcarl\anaconda3\lib\site-packages (from yellowbrick) (1.7.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\jcarl\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->
yellowbrick) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\jcarl\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.
0.2->yellowbrick) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\jcarl\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2-
>yellowbrick) (1.3.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\jcarl\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yel
lowbrick) (9.0.1)
Requirement already satisfied: packaging>=20.0 in c:\users\jcarl\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->y
ellowbrick) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\jcarl\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2-
>yellowbrick) (4.25.0)
Requirement already satisfied: six>=1.5 in c:\users\jcarl\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib!=
3.0.0,>=2.0.2->yellowbrick) (1.16.0)
Requirement already satisfied: joblib>=0.11 in c:\users\jcarl\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->yellowbric
k) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\jcarl\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->ye
llowbrick) (2.2.0)
```

Una vez hecho esto, se importa la librería, se instancia el método del modelo k-means y el visualizador, asi mismo se ajustan los datos al modelo y finalmente se muestra la gráfica.

```
In [19]:
from yellowbrick.cluster import KElbowVisualizer
```

```
In [20]:
# Instanciar el modelo de clustering y el visualizador
km = KMeans(random_state=42)
visualizer = KElbowVisualizer(km, k=(2,10))

visualizer.fit(X)        # Ajustar los datos al visualizador
visualizer.show()        # Finalizar y mostrar la figura
```



Como se observa en la imagen, la grafica se muestra como un codo y nos marca mediante una línea negra el valor optimo de K, que para este caso indica que lo mejor es utilizar un valor de K=5.

Una vez identificados el valor de K, se procede a entrenar el algoritmo con nuestro dataset y se obtienen los datos de los centroides y las etiquetas.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

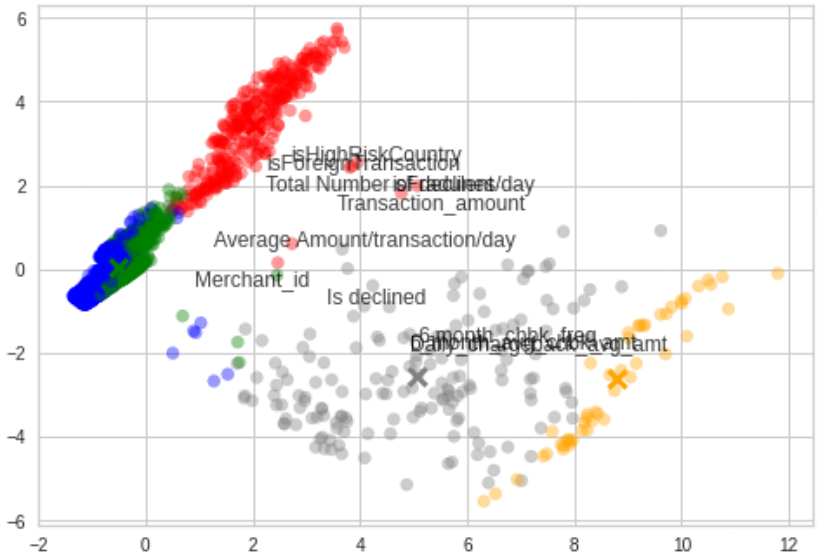
In [21]:
```
algoritmo = KMeans(n_clusters = 5,init = 'k-means++', max_iter = 300,n_init = 10)
```

In [22]:
```
#se entrena al algoritmo
algoritmo.fit(X)

#se obtienen los datos de los centroides y las etiquetas
centroides, etiquetas = algoritmo.cluster_centers_, algoritmo.labels_
```

In [23]:
```
muestra_pred = algoritmo.predict(muestras)
```

Finalmente, se grafican los resultados, tal y como se muestra a continuación.

In [24]:
```
### GRAFICAR LOS DATOS JUNTO A LOS RESULTADOS ###
# Se aplica la reducción de dimensionalidad a los datos
from sklearn.decomposition import PCA

modelo_pca = PCA(n_components = 2)
modelo_pca.fit(X)
pca = modelo_pca.transform(X)

#Se aplicar la reducción de dimsensionalidad a los centroides
centroides_pca = modelo_pca.transform(centroides)

# Se define los colores de cada clúster
colores = ['blue', 'red', 'green', 'orange', 'gray']

#Se asignan los colores a cada clústeres
colores_cluster = [colores[etiquetas[i]] for i in range(len(pca))]

#Se grafica los componentes PCA
plt.scatter(pca[:, 0], pca[:, 1], c = colores_cluster,
            marker = 'o',alpha = 0.4)

#Se grafican los centroides
plt.scatter(centroides_pca[:, 0], centroides_pca[:, 1],
            marker = 'x', s = 100, linewidths = 3, c = colores)

#Se guadan los datos en una variable para que sea fácil escribir el código
xvector = modelo_pca.components_[0] * max(pca[:,0])
yvector = modelo_pca.components_[1] * max(pca[:,1])
columnas = data.columns

#Se grafican los nombres de los clústeres con la distancia del vector
for i in range(len(columnas)):
    #Se grafican los vectores
    #plt.arrow(0, 0, xvector[i], yvector[i], color = 'black',
              # width = 0.0005, head_width = 0.02, alpha = 0.75)
    #Se colocan los nombres
    plt.text(xvector[i], yvector[i], list(columnas)[i], color='black',
             alpha=0.75)

plt.show()
```



Al analizar el resultado final de clasificación, se pueden observar los 5 clústers definidos por una X en su respectivo color (azul, rojo, verde, naranja y gris). Dentro de estos subconjuntos se observan valores muy alejados de sus centroides, siendo estos valores atípicos ya que no están correctamente categorizados.

Ventajas y desventajas del algoritmo K-means:

Ventajas:

- Es simple y fácil de implementar.
- Es computacionalmente eficiente en el tiempo para conjuntos de datos pequeños y medianos.
- Es muy escalable, por lo que se puede utilizar con grandes conjuntos de datos.
- Es ampliamente utilizado en la industria y en la investigación.

Desventajas:

- Es muy sensible a los valores atípicos y al ruido.
- La elección del número de clústers no siempre es obvia y puede requerir ajustes manuales.
- No es efectivo para clústers no esféricos o de forma irregular.

- La calidad de los resultados depende en gran medida de la elección inicial de los centroides.

Aplicaciones:

- Segmentación de clientes o usuarios.
- Agrupamiento de productos o servicios similares.
- Compresión de imágenes y reducción de dimensionalidad.
- Detección de anomalías.

Cuando utilizarlo:

- Cuando los datos son de alta dimensionalidad y los clústers son esféricos.
- Cuando se desea segmentar un conjunto de datos en clústers mutuamente exclusivos.

Cuando no utilizarlo:

- Cuando los datos tienen outliers o ruido.
- Cuando los clústers son de forma irregular o no esférica.
- Cuando no se tiene una idea clara del número de clústers que se necesitan.

Ventajas y desventajas del algoritmo Isolation Forest:

Ventajas:

- Es computacionalmente eficiente en el tiempo para conjuntos de datos grandes.
- Es menos sensible a los outliers y al ruido que otros algoritmos de detección de anomalías.
- Es capaz de detectar diferentes tipos de anomalías (puntos anómalos, valores anómalos, etc.).
- No requiere conocimiento previo de la distribución de los datos.
- Puede ser utilizado para detección de anomalías en tiempo real.

Desventajas:

- Puede ser computacionalmente costoso en memoria cuando se utiliza con grandes conjuntos de datos.
- La calidad de los resultados puede depender en gran medida de los parámetros seleccionados.
- No es tan interpretable como otros algoritmos de detección de anomalías.

Aplicaciones:

- Detección de fraudes en transacciones financieras.
- Detección de intrusos en sistemas de seguridad informática.
- Detección de anomalías en sistemas de monitoreo de redes y servidores.
- Detección de anomalías en sensores y dispositivos IoT.

Cuando utilizarlo:

- Cuando se desea detectar anomalías en un conjunto de datos grande y complejo.
- Cuando se necesita un enfoque de detección de anomalías no supervisado.
- Cuando se desea detectar diferentes tipos de anomalías.

Cuando no utilizarlo:

- Cuando se tiene un conjunto de datos muy pequeño.
- Cuando se necesita una interpretación precisa de los resultados de detección de anomalías.
- Cuando los datos tienen una estructura clara y predecible.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Conclusiones

No se puede decir que un algoritmo es mejor que otro de manera general, ya que la elección del algoritmo depende de las características específicas del problema a resolver. Tanto K-means como Isolation Forest son algoritmos útiles en diferentes situaciones. K-means es especialmente útil cuando se trata de agrupar datos en diferentes grupos o clústers, y cuando los datos son relativamente bien distribuidos y separables. Sin embargo, puede tener dificultades para identificar clústers de formas complejas o no convencionales, y puede ser sensible a los valores atípicos.

Por otro lado, Isolation Forest es especialmente útil cuando se trata de detectar valores atípicos o anomalías en un conjunto de datos, y es capaz de detectar valores atípicos en datos de alta dimensionalidad. Sin embargo, puede tener dificultades para detectar anomalías en conjuntos de datos densamente poblados o con patrones de anomalías similares a los de los datos normales.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Bibliografía

Tchanake, C. (2021, October 1st). Credit Card Fraud Detection. Kaggle. https://www.kaggle.com/code/fernolf/credit-card-fraud-detection

Bagnato, J. (2018, March 12th). K-Means en Python paso a paso. Aprende Machine Learning. https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/

Amat, J. (2020 May). Detección de anomalías: Isolation Forest. Cienciadedatos.net. https://www.cienciadedatos.net/documentos/66_deteccion_anomalias_isolationforest.html

Fernández, A. (s.f.). Isolation Forest en Python. Ander Fernández. https://anderfernandez.com/blog/isolation-forest-en-python/.

Anomaly detection: A survey: ACM Computing Surveys: Vol 41, No 3. (2023). ACM Computing Surveys (CSUR). https://dl.acm.org/doi/10.1145/1541880.1541882

Método Elbow y método de siluetas para la determinación del número de clústeres en K-means - Comunidad Huawei Enterprise. (2023). Comunidad Huawei Enterprise; https://forum.huawei.com/enterprise/es/m%C3%A9todo-elbow-y-m%C3%A9todo-de-siluetas-para-la-determinaci%C3%B3n-del-n%C3%BAmero-de-cl%C3%BAsteres-en-k-means/thread/1059244-100757

AprendeIA con Lidgi Gonzalez. (2020, August 11th). ALGORITMO KMEANS - PRACTICA CON PYTHON | #7 Curso Aprendizaje no Supervisado con Python [Archivo de Video]. YouTube. https://www.youtube.com/watch?v=xLkhiIcTJF8

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos:	29/03/2023
	Nombre:	

Integrantes equipo 26:

Francisco Javier Guerrero Enríquez

Cesar Geovanni Machuca Pereida

Bryan Daniel Moreno Abrego

Juan Carlos Romero Perez