0.1 Portada

Universidad Autónoma de Querétaro

Maestría en Ciencias en Inteligencia Artificial

Materia - Tópicos Selectos IV | Machine Learning

Actividad - Análisis de datos

Profesor - Dr. Marco Antonio Aceves Fernández

Alumno - Juan Ignacio Ortega Gómez
Expediente - 309236

Quertearo, Queretaro a 25 de Febrero de 2022

0.2 Introducción

En el mundo, cada vez nos vemos más abrumados por los datos. Con datos en cada uno de los dispositivos que manejamos diariamente, datos dentro de la web y la nube. Conforme esta cantidad aumenta, las oportunidades para obtener información útil de los datos también. No solo es acerca de observar los datos, es sobre lo que estos nos pueden decir sobre quienes o las cosas que los generan.

Hoy en día podemos encontrar información sobre las técnicas de análisis de datos, conocimiento que nos permitirá no solo poder obtener oportunidades de aprovechamiento con los datos, si no también de conocer las mejores maneras de generar la información y utilizarla.

Por ende, se recomienda tener un acercamiento hacia la técnicas de análisis de datos, lo cual se comienza con este escrito. [1]

En este reporte, se pueden encontrar algoritmos generalizados para utilizar con cualquier base de datos y poder extraer así sus primeras características para el análisis. Además, de encontrar un par de técnicas correctivas que ayudarán a mejorar la base de datos que se ingresa.

0.3 Metodología

DEFINICIONES ESTADÍSTICAS

- La media muestral (Promedio)

Indica el centro de los datos.

Promedio = (1 / n) * sumatoria de Xi [2]

- Desviaciones ((X1 - Xpromedio), . . . , (Xn - Xpromedio)) Distancias de cada valor de la muestra a la media de la muestra. Al ser una resta, genera valores tanto positivos como negativos, por eso se eleva al cuadrado al utilizarse en la varianza y en la desviación estándar, para hacer todos los resultados de la resta positivos.
- Varianza muestral Constituye el promedio de las desviaciones al cuadrado, excepto que lo dividimos entre n-1 en lugar de n.

s^2 = (1 / (n - 1)) * sumatoria de $((Xi - Xpromedio)^2)$ [2]

- Desviación estándar Mide el grado de dispersión.

s = $(s^2)^{1 / 2}$ o raíz cuadrada de s^2 [2]

- Cuartiles

La mediana divide la muestra a la mitad, los cuartiles lo dividen, tanto como sea posible, en cuartos.

Primer cuartil = 0.25 (n + 1) [2]

Segundo cuartil = 0.5 (n + 1) [2] –> Idéntico a la mediana

Tercer cuartil = 0.75 (n + 1) [2]

El resultado te dice el número del valor que representa el X cuartil, de los datos ordenados de forma ascendente.

Solo si el resultado es un entero, si no, se toma el promedio de los valores de la muestra de cualquier lado de este valor, tomando la muestra de forma ordenada ascendente.

- Covarianza

Es una medida de la intensidad de la relación entre dos variables aleatorias [2].

cov(X, Y) = promedio((Xi - promedio de X) * (Yi - promedio de Y)) [2]

CONCEPTOS EN EL MANEJO DE DATOS

- Instancias
  Cada instancia se caracteriza por los valores de los atributos que miden diferentes aspectos por cada medición. Si lo vemos en el sentido de una base de datos, las instancias serían los registros de esta, es decir, las filas.
- Atributos
  Si lo vemos como una base de datos, los atributos serían el equivalente a las columnas. El valor de un atributo para una particular instancia es la medición de la cantidad a la cual el atributo se refiere.
- Clasificación de atributos

-Atributo categórico:

Categóricos (cualitativos): representan categorías más que números. Se dividen a su vez en:

Nominales: no tienen orden significativo.

Ordinales: tienen orden definido.

-Atributo numérico:

Numéricos(cuantitativos): son atributos que son núemros, se dividen a su vez en:

Intervalo: no existe un .

Tasa: el cero existe.

Atributo discreto: Tiene un número finito o contable de valores.

Atributo continuo: Tiene un número infinito de valores posibles.

- Tipos de distribución según histogramas

Histograma Uniforme

Histograma Normal (Unimodal)

Histograma Unimodal sesgado izquierda

Histograma Unimodal sesgado derecha

Histograma multimodal

Histograma exponencial

- Problema de calidad en los datos cualquier dato inusual que se encuentre en la base de datos.

Los problemas más comunes de los datos son:

Valores faltantes.

Problemas de cardinalidad irregular. Se da cuando una característica tiene un dato que no es lo que se esperaría de una característica.

Valores atípicos outliers. Valores que se localizan muy alejados de la tendencia central de una característica. Se puede categorizar en válidos y no válidos [1].

- Balance entre clases

Se busca tener la misma cantidad de ejemplos por cada clase.

- Normalizar los datos

Algunos algoritmos de inteligencia artificial requieren que todos los datos se centren en un rango específico de valores, normalmente de -1 a 1 o de 0 a 1. Incluso si no se requiere que los datos se encuentren dentro de los valores, es buena idea generalmente asegurarse de que los valores se encuentran dentro de un rango específico.

Normalización de valores ordinales
Para normalizar un set ordinal, se tiene que preservar el orden.

Normalización de valores cuantitativos
Lo primero que se tiene que hacer es observar el rango en el cual se encuentran dichos valores y el intervalo al que se quiere normalizar. No todos los valores requieren ser normalizados.

Es necesario realizar los cálculos de las siguientes variables para encontrar el valor normalizado:

-Máximo de los datos = el valor más alto de la observación sin normalizar. [1]

-Mínimo de los datos = el menor alto de la observación sin normalizar. [1]

-Máximo normalizado = el valor más alto limítrofe al que el máximo de los datos será normalizado. [1]

-Mínimo normalizado = el valor más bajo limítrofe al que el mínimo de los datos será normalizado. [1]

-Rango de datos = Máximo de los datos - Mínimo de los datos [1]

-Rango normalizado = Máximo normalizado - Mínimo normalizado [1]

-D = Valor a normalizar - Mínimo de los datos [1]

-DPct = D / Rango de datos [1]

-dNorm = Rango normalizado * DPct [1]

-Normalizado = Mínimo normalizado + dNorm [1]

De esta forma se obtiene el valor normalizado.

- Imputación de los datos

En ciencia de datos, usualmente, se tienen dos enfoques para lidiar con estos valores faltantes: omitir las instancias con valores faltantes o realizar técnicas de imputación y estimar los datos faltantes utilizando los valores existentes.

La imputación es una técnica para reemplazar valores perdidos con valores que se encuentran. El objetivo es la de tener una base de datos con la menor cantidad de instancias faltantes que contengan la misma distribución que lo datos existentes para que puedan ser analizados.

Técnicas determínisticas de imputación Funcionan cuando, de acuerdo a las mimas condiciones de los datos, producen las mismas respuestas. Entre estas técnicas se encuentra la imputación por vecino más cercano.

Imputación por vecino más cercano: Se calcula la distancia entre la instancia a imputar, usualmente, por medio de la distancia euclidiana y los datos que tienen valor establecido. Una vez que se calcula el dato más cercano, se utiliza este para imputar la instancia faltante.

- Distancia euclidiana

La distancia euclidiana está basada en la distancia bidimensional entre dos vectores. Esto es, la distancia entre dos puntos como si se dibujara una línea con una regla de un punto a otro. Específicamente, si se tienen dos punto (x1, y1) y (x2, y2), la distancia euclidiana puede ser calculada mediante la ecuación [1]:

d = raíz cuadrada((x2 - x1)^2 + (y2 - y1)^2) [1]

0.4 Importamos la base de datos en google colab.

[1]:
```
import sys
from google.colab import drive

drive.mount('/content/drive', force_remount = True)
sys.path.append('/content/drive/MyDrive/ML_TSIV')
```

```
Mounted at /content/drive
```

[2]:
```
import pandas as pd
DB = pd.read_csv('/content/drive/MyDrive/ML_TSIV/car_data_for_price_estimation.
 ↪csv')
```

0.4.1 Mostramos los primeros 5 instancias y los títulos de los atributos.

[4]:
```
DB.head()
```

[4]:
```
   make_id  model_id  submodel_id  ...  year  currency
updated_at
0       86     912.0        957.0  ...  2017       PKR  2019-07-25
21:50:04.860913+00
1       73     774.0        599.0  ...  2013       AED  2019-12-27
22:50:07.718331+00
2       86     935.0          NaN  ...  2020       PKR  2020-12-02
22:49:39.536877+00
3        5      41.0          NaN  ...  2012       ZAR  2020-02-16
22:50:05.555076+00
4       22     222.0          NaN  ...  2016       AED  2020-10-04
```

Convertimos la base de datos a un arreglo para manejarla como matriz.

[3]:
```
import numpy as np
DB = DB.to_numpy()
```

0.5 Obtenemos la cantidad de atributos y de instancias por atributo.

[4]:
```
Atributos = DB[0]
NoAtributos = len(Atributos)
Instancias = DB.T[0]
NoInstancias = len(Instancias)

print('La cantidad de atributos es igual a', str(NoAtributos), 'y la cantidad de␣
 ↪instancia es igual a', str(NoInstancias) + '.')
```

```
La cantidad de atributos es igual a 9 y la cantidad de instancia es igual a
1048575.
```

0.6 Definimos las estadísticas de cada uno de los atributos.

0.6.1 Tipo de atributo.

[5]:
```
for idx, element in enumerate(Atributos):
  if str(type(Atributos[idx]))[8 : -2] == 'int':
    TipoAtributo = 'numérico discreto'
  elif str(type(Atributos[idx]))[8 : -2] == 'float':
    TipoAtributo = 'numérico continuo'
  elif str(type(Atributos[idx]))[8 : -2] == 'str':
    TipoAtributo = 'categórico'
  else:
    TipoAtributo = 'desconocido'
  print('Un ejemplo del atributo', str(idx + 1), 'es "' + str(Atributos[idx]) +␣
 ↪'" y es de tipo:', str(type(Atributos[idx]))[8 : -2] +  ', por lo tanto, es un␣
 ↪atributo', TipoAtributo + '.')
```

```
Un ejemplo del atributo 1 es "86" y es de tipo: int, por lo tanto, es un
atributo numérico discreto.
Un ejemplo del atributo 2 es "912.0" y es de tipo: float, por lo tanto, es un
atributo numérico continuo.
Un ejemplo del atributo 3 es "957.0" y es de tipo: float, por lo tanto, es un
atributo numérico continuo.
Un ejemplo del atributo 4 es "2150000" y es de tipo: int, por lo tanto, es un
atributo numérico discreto.
Un ejemplo del atributo 5 es "30000" y es de tipo: int, por lo tanto, es un
atributo numérico discreto.
Un ejemplo del atributo 6 es "PK" y es de tipo: str, por lo tanto, es un
atributo categórico.
Un ejemplo del atributo 7 es "2017" y es de tipo: int, por lo tanto, es un
```

```
atributo numérico discreto.
Un ejemplo del atributo 8 es "PKR" y es de tipo: str, por lo tanto, es un
atributo categórico.
Un ejemplo del atributo 9 es "2019-07-25 21:50:04.860913+00" y es de tipo: str,
por lo tanto, es un atributo categórico.
```

0.6.2 Máximo y mínimo.

[6]:
```
MaximoDeAtributos = []
MinimoDeAtributos = []
for idx, element in enumerate(Atributos):
  CaractMax = max(DB.T[idx])
  CaractMin = min(DB.T[idx])
  MaximoDeAtributos.append(CaractMax)
  MinimoDeAtributos.append(CaractMin)
  print('Los elementos máximo y mínimo del atributo', str(idx + 1), 'son',␣
 ↪str(CaractMax), 'y', str(CaractMin), 'respectivamente.')
```

```
Los elementos máximo y mínimo del atributo 1 son 92 y 1 respectivamente.
Los elementos máximo y mínimo del atributo 2 son 1103.0 y 1.0 respectivamente.
Los elementos máximo y mínimo del atributo 3 son 1099.0 y 6.0 respectivamente.
Los elementos máximo y mínimo del atributo 4 son 4000000000000 y 100
respectivamente.
Los elementos máximo y mínimo del atributo 5 son 1200012000 y 0 respectivamente.
Los elementos máximo y mínimo del atributo 6 son ZA y AE respectivamente.
Los elementos máximo y mínimo del atributo 7 son 2556 y 1926 respectivamente.
Los elementos máximo y mínimo del atributo 8 son ZAR y AED respectivamente.
Los elementos máximo y mínimo del atributo 9 son 2021-05-07 06:09:59.333081+00 y
2019-05-09 21:50:02.633848+00 respectivamente.
```

NOTA:
* En los atributos categóricos se define el maximo como la carcaterística que tiene la letra más lejana en el abecedario, es decir, tomando la más cercana a la A y a la más lejana a la Z. Siendo el valor mínimo la letra más cercana en el abecedario. * Mientras que en el caso de fechas toma como mínimo a las fecha más temprana y máximo a la fecha más tardía.

0.6.3 Media, varianza y desviación estandar.

[7]:
```
medias = []
for idx, element in enumerate(Atributos):
  sumatoria = 0
  sumatoria2 = 0
  NoInsta = 0
  if str(type(Atributos[idx]))[8 : -2] != 'str':
    for idx2, element2 in enumerate(Instancias):
      if str(DB.T[idx][idx2]) != 'nan':
        sumatoria += DB.T[idx][idx2]
        NoInsta += 1
    #media = (1 / NoInstancias) * sumatoria
    media = (1 / NoInsta) * sumatoria
    medias.append(media)
    for idx3, element2 in enumerate(Instancias):
      if str(DB.T[idx][idx3]) != 'nan':
        sumatoria2 += ((DB.T[idx][idx3] - media) ** 2)
    #s2 = (1 / NoInstancias - 1) * sumatoria2
    s2 = (1 / (NoInsta - 1)) * sumatoria2
    s = s2 ** (1 / 2)
    print('Los valores de media, varianza y desviación estandar del atributo',␣
↪str(idx + 1), 'son', str(media) + ',', str(s2), 'y', str(s), 'respectivamente.
↪')
```

```
Los valores de media, varianza y desviación estandar del atributo 1 son
59.29174260305653, 724.629939191225 y 26.918951301847272 respectivamente.

Los valores de media, varianza y desviación estandar del atributo 2 son
643.6038659943823, 95990.09654277256 y 309.82268564902176 respectivamente.

Los valores de media, varianza y desviación estandar del atributo 3 son
719.5627100870934, 83028.08133053653 y 288.1459375568854 respectivamente.

Los valores de media, varianza y desviación estandar del atributo 4 son
4557117.182156737, 1.5273344327328252e+19 y 3908112629.816118 respectivamente.


Los valores de media, varianza y desviación estandar del atributo 5 son
129293.26172758268, 11118300352257.6 y 3334411.5451242067 respectivamente.

Los valores de media, varianza y desviación estandar del atributo 7 son
2011.226562716067, 57.680613812637766 y 7.594775428716623 respectivamente.
```

NOTA:

- En esta ocación solo se toman los valores de media y distribución estándar de los atributos numéricos y se ignoran los valores faltantes.
- Es de verdadera relevancia transformar los atributos categóricos a números?, es decir, en

el aprendizaje máquina realmente aporta en algo agregar ese tipo de características en el proceso de aprendizaje?

0.6.4 Cuartiles

[16]:
```
for idx, element in enumerate(Atributos):
  #sumatoria = 0
  #NoInsta = 0
  if str(type(Atributos[idx]))[8 : -2] != 'str':
    atrib = DB.T[idx]
    atrib.sort()

    NoCuartil1 = 0.25 * (NoInstancias + 1)
    if str(type(NoCuartil1))[8 : -2] != 'int':
      pos1 = round(NoCuartil1)
      if pos1 < NoCuartil1:
        pos2 = pos1 + 1
      else:
        pos2 = pos1 - 1
      NoCuartil1 = round((pos1 + pos2) / 2)
    Cuartil1 = atrib[NoCuartil1 + 1]
    incremento = 1
    while True:
      if str(Cuartil1) == 'nan':
          Cuartil1 = atrib[NoCuartil1]
          NoCuartil1 -= 1
      else:
        break

    NoCuartil2 = 0.5 * (NoInstancias + 1)
    if str(type(NoCuartil2))[8 : -2] != 'int':
      pos1 = round(NoCuartil2)
      if pos1 < NoCuartil2:
        pos2 = pos1 + 1
      else:
        pos2 = pos1 - 1
      NoCuartil2 = round((pos1 + pos2) / 2)
    Cuartil2 = atrib[NoCuartil2 + 1]
    while True:
      if str(Cuartil2) == 'nan':
          Cuartil2 = atrib[NoCuartil2]
          NoCuartil2 -= 1
      else:
        break

    NoCuartil3 = 0.7 * (NoInstancias + 1)
    if str(type(NoCuartil3))[8 : -2] != 'int':
```

```
      pos1 = round(NoCuartil3)
      if pos1 < NoCuartil3:
        pos2 = pos1 + 1
      else:
        pos2 = pos1 - 1
      NoCuartil3 = round((pos1 + pos2) / 2)
    Cuartil3 = atrib[NoCuartil3 + 1]
    while True:
      if str(Cuartil3) == 'nan':
          Cuartil3 = atrib[NoCuartil3]
          NoCuartil3 -= 1
      else:
        break

    print('Los valores del primer, segundo y tercer cuartil del atributo',␣
↪str(idx + 1), 'son', str(Cuartil1) + ',', str(Cuartil2), 'y', str(Cuartil3),␣
↪'respectivamente.')
```

```
Los valores del primer, segundo y tercer cuartil del atributo 1 son 35, 66 y 83
respectivamente.

Los valores del primer, segundo y tercer cuartil del atributo 2 son 359.0, 697.0
y 912.0 respectivamente.

Los valores del primer, segundo y tercer cuartil del atributo 3 son 1037.0,
1092.0 y 956.0 respectivamente.

Los valores del primer, segundo y tercer cuartil del atributo 4 son 65000,
180000 y 470000 respectivamente.

Los valores del primer, segundo y tercer cuartil del atributo 5 son 35000, 85000
y 123456 respectivamente.

Los valores del primer, segundo y tercer cuartil del atributo 7 son 2007, 2013 y
2016 respectivamente.
```

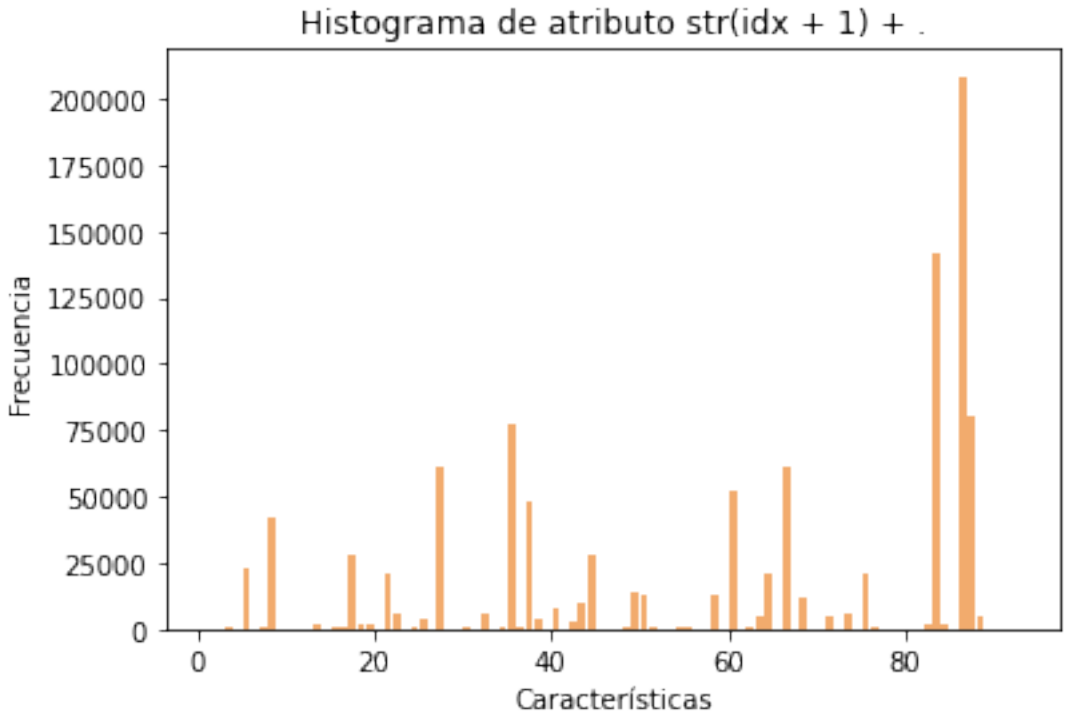
NOTA:

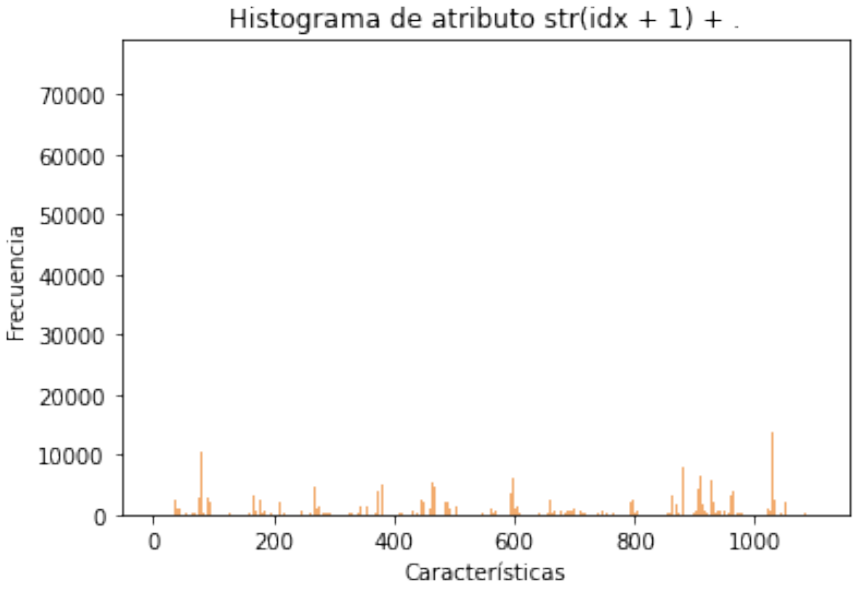
- En esta ocasión solo se toman los valores de los atributos numéricos para el cálculo de cuartiles.

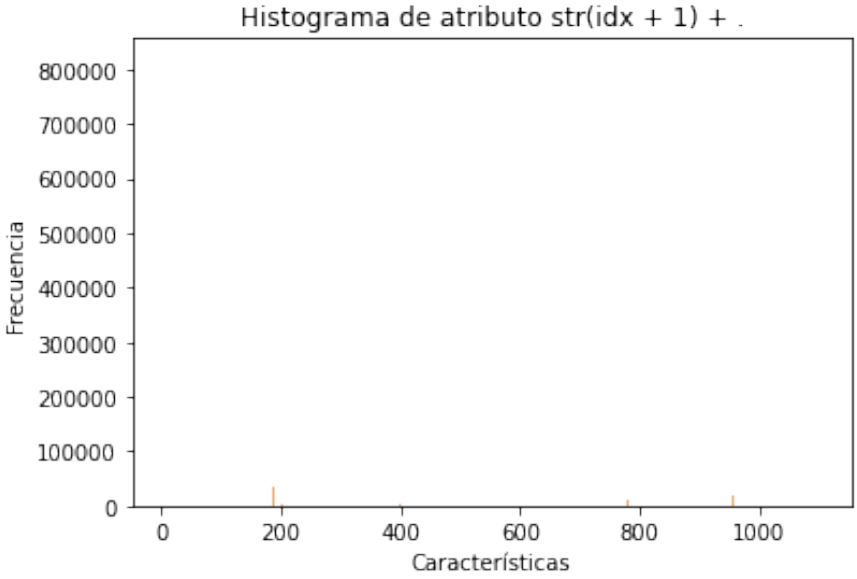
0.6.5 Tipo de distribución

Para definir el tipo, es preferible obtener la gráfica del atributo que permita determinar de manera visual y manual si el atributo presenta una distribución Uniforme, Normal, Unimodal sesgada a la izquierda, Unimodal sesgada a la derecha, multimodal o exponencial.

[ ]:
```
import matplotlib.pyplot as plt

for idx, element in enumerate(Atributos):
  if str(type(Atributos[idx]))[8 : -2] != 'str':
    if max(DB.T[idx]) >= 10000:
      datos = DB.T[idx] / 10000
    else:
      datos = DB.T[idx]

    intervalos = range(int(min(datos)), int(max(datos)) + 2)

    plt.hist(x = datos, bins = intervalos, color = '#F2AB6D', rwidth = 0.85)
    plt.title('Histograma de atributo ' + str(idx + 1) + '.')
    plt.xlabel('Características')
    plt.ylabel('Frecuencia')
    #plot.xticks(intervalos)

    plt.show()
```







NOTA:

- En esta ocación solo se toman los valores de los atributos numéricos para la generación de histograma.
- Todo parece indicar que por la magnitud de los valores la RAM se satura y no termina de analizar cada atributo.

0.7 Relaciones entre atributos.

0.7.1 Covarianza

[18]:
```
for idx, element in enumerate(Atributos):
  if idx < len(medias) - 2:
    if str(type(Atributos[idx]))[8 : -2] != 'str':
      MX = medias[idx]
      MY = medias[idx + 1]
      NoInsta = 0
      sumatoria = 0
      for idx2, element2 in enumerate(Instancias):
        if str(DB.T[idx][idx2]) != 'nan':
          sumatoria += (DB.T[idx][idx2] - MX) * (DB.T[idx + 1][idx2] - MY)
          NoInsta += 1
    #media = (1 / NoInstancias) * sumatoria
    Covarianza = (1 / NoInsta) * sumatoria
    print('La covarianza entre el atributo', str(idx + 1), 'y el atributo',␣
 ↪str(idx + 2), 'es igual a', str(Covarianza) + '.')
```

```
La covarianza entre el atributo 1 y el atributo 2 es igual a nan.
La covarianza entre el atributo 2 y el atributo 3 es igual a nan.
La covarianza entre el atributo 3 y el atributo 4 es igual a 7447607.833457576.
La covarianza entre el atributo 4 y el atributo 5 es igual a 4724537849602135.0.
```

NOTA:

- Es notable que a pesar de podr ser corregido, las variables faltantes en la base de datos generan diversos problemas en los cálculos estadísticos. En este apartado se deja sin corregir el problema de los 'nan', para destacar la relevancia de la imputación posterior.

0.8 No. de datos faltantes por atributo

[19]:
```
NoFaltantes = []
for idx, element in enumerate(Atributos):
  sumatoria = 0
  for idx2, element2 in enumerate(Instancias):
    if str(DB.T[idx][idx2]) == 'nan':
      sumatoria += 1
  NoFaltantes.append(sumatoria)
  print('La cantidad de carcaterísticas faltantes en el atributo', str(idx + 1),␣
 ↪'es igual a', str(sumatoria) + '.')
```

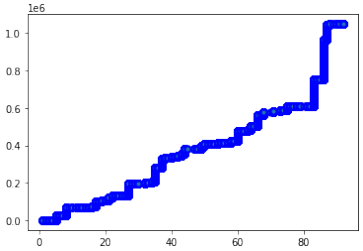
```
La cantidad de carcaterísticas faltantes en el atributo 1 es igual a 0.
La cantidad de carcaterísticas faltantes en el atributo 2 es igual a 120793.
La cantidad de carcaterísticas faltantes en el atributo 3 es igual a 876920.
La cantidad de carcaterísticas faltantes en el atributo 4 es igual a 0.
La cantidad de carcaterísticas faltantes en el atributo 5 es igual a 0.
La cantidad de carcaterísticas faltantes en el atributo 6 es igual a 0.
La cantidad de carcaterísticas faltantes en el atributo 7 es igual a 0.
```

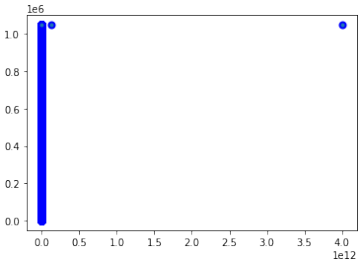
```
La cantidad de carcaterísticas faltantes en el atributo 8 es igual a 0.
La cantidad de carcaterísticas faltantes en el atributo 9 es igual a 0.
```

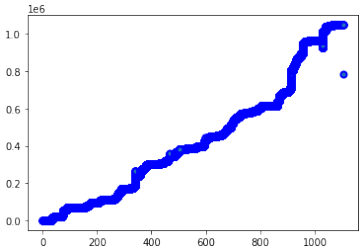
0.9 Núm. de datos atípicos por atributo

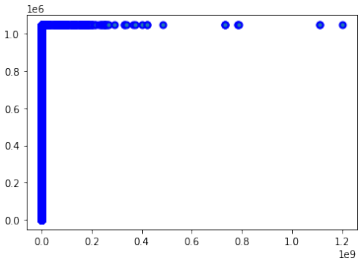
Es más sencillo identificar los valores atípicos mediante la representación gráfica de los datos y de forma manual por el usuario, para interpretar mejor cada dato de la base de datos, de acuerdo al contexto.

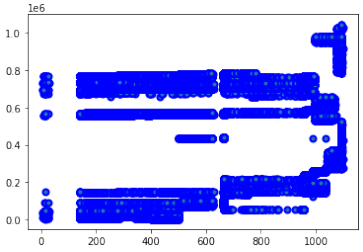
[29]:
```
import matplotlib.pyplot as plt
for idx, element in enumerate(Atributos):
  if str(type(Atributos[idx]))[8 : -2] != 'str':
    X = DB.T[idx]
    Y = []
    for idxX, elemento in enumerate(X):
      Y.append(idxX + 1)
  plt.scatter(X, Y, linewidths = 2, edgecolors = 'blue')
  plt.show()
```

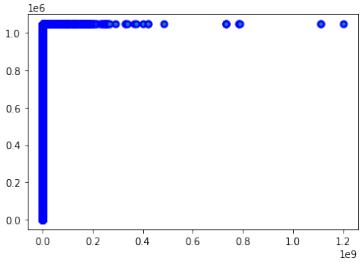


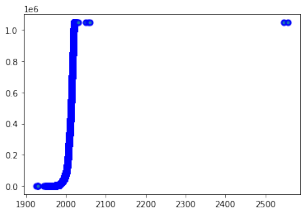


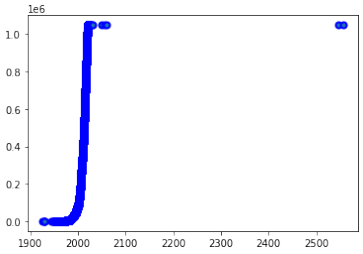


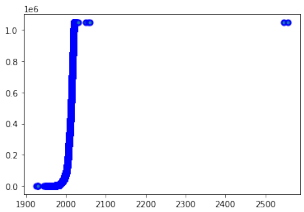












0.10 Balance de la base de datos

[55]:
```
CantMax = max(NoFaltantes)
for idx, element in enumerate(Atributos):
  Proporcion = NoFaltantes[idx] * 100 / CantMax
  print('El porcentaje de carcaterísticas presentes en el atributo', str(idx +␣
 ↪1), 'es igual a', str(Proporcion) + '.')
```

```
El porcentaje de carcaterísticas presentes en el atributo 1 es igual a 0.0.
El porcentaje de carcaterísticas presentes en el atributo 2 es igual a
13.774688683118187.
El porcentaje de carcaterísticas presentes en el atributo 3 es igual a 100.0.
El porcentaje de carcaterísticas presentes en el atributo 4 es igual a 0.0.
El porcentaje de carcaterísticas presentes en el atributo 5 es igual a 0.0.
El porcentaje de carcaterísticas presentes en el atributo 6 es igual a 0.0.
El porcentaje de carcaterísticas presentes en el atributo 7 es igual a 0.0.
El porcentaje de carcaterísticas presentes en el atributo 8 es igual a 0.0.
El porcentaje de carcaterísticas presentes en el atributo 9 es igual a 0.0.
```

0.11 Normalización de los datos

No siempre es necesario normalizar, sin embargo se recomienda tratar de mantener los datos entre 0 y 1.

[38]:
```
DBNorm = []
MaximoNormalizado = 1
MinimoNormalizado = 0
RangoNormalizado = MaximoNormalizado - MinimoNormalizado
for idx, element in enumerate(Atributos):
  CaractNorm = []
  if str(type(Atributos[idx]))[8 : -2] != 'str':
    RangodeDatos = MaximoDeAtributos[idx] - MinimoDeAtributos[idx]
    for idx2, element2 in enumerate(Instancias):
      if str(DB.T[idx][idx2]) != 'nan':
        D = DB.T[idx][idx2] - MinimoDeAtributos[idx]
        DPct = D / RangodeDatos
        dNorm = RangoNormalizado * DPct
        Normalizado = MinimoNormalizado + dNorm
        CaractNorm.append(Normalizado)
      else:
        CaractNorm.append(DB.T[idx][idx2])
  else:
    for idx2, element2 in enumerate(Instancias):
      CaractNorm.append(DB.T[idx][idx2])
  DBNorm.append(CaractNorm)
```

NOTA:
* En los atributos categóricos no se realiza normalización de los ordinales porque es una base de datos mixta (con atrinutos numéricos y categóricos) y existe más de un atributo categórico, porlo que no se podrían normalizar todos y dejar cada instancia con las características que le corresponde, adecuadamente.
* Cómo se maneja la normalización en estos casos donde la base de datos se considera mixta?, y cómo se normaliza cuando existe más de un atributo categórico?.

0.12 Imputación por vecino más cercano (por sus siglas en inglés NN o K-NN)

[ ]:
```
import numpy as np
for idx, element in enumerate(Instancias):
  for idx2, element2 in enumerate(Atributos):
    if str(DB[idx][idx2]) == 'nan':

      for idx3, element3 in enumerate(Atributos):
        if str(DB[idx][idx3]) != 'nan':
          break

      d = []
      posd = []
      for idx4, element4 in enumerate(Instancias):
        if idx4 < idx3:
          d.append((((idx3 - idx3) ** 2) + (idx - idx4) ** 2) ** (1 / 2))
          posd.append(idx4)
        elif idx4 == idx3:
          d.append(0)
          posd.append(idx4)
        else:
          d.append((((idx3 - idx3) ** 2) + (idx4 - idx) ** 2) ** (1 / 2))
          posd.append(idx4)

      dNN = min(d)
      posNN = np.where(DB.T[idx3] == dNN)
      DB[idx][idx2] = DB[posNN[0][0]][idx2]
```

NOTA:
* El algoritmo se ve limitado por la velocidad del internet al realizarse en colab y al manejar valores grandes en las magnitudes la base de datos, y los ciclos for anidados, juntos con las operaciones requeridas, es necesario buscar una alternativa para mejorarlo.

0.13 Conclusión

Se realizaron algoritmos que permiten identificar las características de los datos de cualquier base de datos. Generando información que permitirá un análisis de los datos con los que se plantea trabajar y además se generaron al final un par de algoritmos que ayudan a la base de datos a estar completa y en mejor forma para su utilización y sobre todo en algortimos de inteligencia artificial.

Se destacó la importancia de estas últimas correciones mencionadas, para eliminar valores faltantes, los cuales generaron complicaciones incluso desde esta etapa del manejo de los datos.

Se plantea realizar pruebas diversas sobre los algoritmos creados para robustecerlos con cada error encontrado y cada base de datos desglozada.

0.14 Referencias

[1] M. A. Aceves Fernández, Inteligencia Artificial para programadores con prisa. UNIVERSO de LETRAS, 2021.

[2] W. Navidi, Estadística para ingenieros. México: Mc Graw Hill, 2006.