Pr?ctico_2_curaci?n

August 20, 2019

Universidad Nacional de Córdoba - Facultad de Matemática, Astronomía, Física y Computación

Diplomatura en Ciencia de Datos, Aprendizaje Automático y sus Aplicaciones

Práctico 2 - Curación

Análisis Exploratorio y Curación de Datos - 2019

Integrante:

- Tarletta Juan

Trabajaremos con un dataset que contiene muestras de aceite extraidas a camiones de minería. Estas muestras son extraíadas a diferentes componentes de los camiones (Motor, Diferencial, etc)

Las muestras se envían a un laboratorio que analiza los diferentes componentes químicos de las mismas (Hierro, Cromo, presencia de Agua, horas de utitización, etc), luego de analizarlas emiten un informe indicando el estado de la muesta, lo que permite a los analistas encontar posibles fallas en los equipos, previmiendo futuras roturas.

El dataset cuenta con X features, siendo las más importantes

- Componente: Indica a que componente pertenece la muestra
- Horas Funcionamiento: Indica la cantidad de horas de funcionamiento del camión (sería como el kilometraje de los camiones)
- Horas del Aceite: Representa la cantidad de horas de utilización del aceite (este dato es importante dado que a medida que, a mayor horas de uso del aceite, el mismo comienza a desgastarse)
- Resultado: (El laboratorio indica si la muestra de aceite está Bien = 1, Regular=2, Mal=3)
- St: Presencia de Hollin en el Aceite
- Al: Presencia de Aluminio en el Aceite
- Fe Presencia de Hierro en el Aceite
- Si Presencia de Silicio en el Aceite
- Na Presencia de Sodio en el Aceite
- Visco: Viscosidad del aceite

NOTA: se modifica le dataset y se incluyen el feature * Fecha de Análisis: Indica cuando fué analizada la muestra por el laboratorio

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn
from IPython.display import display, Markdown
```

[2]:
```
seaborn.__version__
```

[2]:
```
'0.9.0'
```

[3]:
```
dataset = pd.read_csv('OilDataSetCuracion.csv')
important_cols= ['Equipo', 'Componente','Resultado','Horas␣
 ↪Funcionamiento','Horas del Aceite', 'St', 'Al', 'Fe', 'Si', 'Na' ]

dataset[important_cols]
```

```
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\IPython\core\interactiveshell.py:3049: DtypeWarning: Columns (0) have
mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[3]:

	Equipo	Componente	Resultado	\
0	1355	Mando Final TD	1.0	
1	1355	Mando Final TI	3.0	
2	1355	Masa Derecha	3.0	
3	1355	Motor	3.0	
4	1355	Sist de Dirección	1.0	
5	1355	Sist. Hidráulico	1.0	
6	1355	Sist. Hidráulico	3.0	
7	1356	Mando Final TI	1.0	
8	1356	Masa Izquierda	1.0	
9	1356	Sist de Dirección	1.0	
10	1356	Sist. Hidráulico	3.0	
11	1356	Sist. Hidráulico	3.0	
12	1357	Motor	1.0	
13	1359	Convertidor	1.0	
14	1359	Convertidor	1.0	
15	1359	Convertidor	1.0	
16	1359	Diferencial Trasero	1.0	
17	1359	Diferencial Trasero	1.0	
18	1359	Diferencial Trasero	3.0	
19	1359	Mando Final TD	1.0	
20	1359	Mando Final TD	3.0	
21	1359	Mando Final TD	3.0	
22	1359	Mando Final TD	3.0	
23	1355	Diferencial Trasero	1.0	
24	1355	Diferencial Trasero	2.0	
25	1355	Masa Derecha	2.0	
26	1355	Masa Izquierda	1.0	
27	1355	Transmisión	1.0	
28	1356	Diferencial Trasero	1.0	
29	1356	Motor	3.0	
...	...	...	...	
21403	2061	Diferencial Trasero	2.0	

21404	2061	Diferencial Trasero	2.0	
21405	2061	Mando Final TD	2.0	
21406	2061	Mando Final TD	2.0	
21407	2061	Mando Final TD	2.0	
21408	2061	Mando Final TI	1.0	
21409	2061	Mando Final TI	1.0	
21410	2061	Mando Final TI	2.0	
21411	2061	Masa Derecha	2.0	
21412	2061	Masa Derecha	2.0	
21413	2061	Masa Derecha	2.0	
21414	2061	Masa Izquierda	2.0	
21415	2061	Masa Izquierda	2.0	
21416	2061	Masa Izquierda	2.0	
21417	2061	Masa Izquierda	2.0	
21418	2061	Motor	3.0	
21419	2061	Motor	3.0	
21420	2061	Motor	2.0	
21421	2061	Motor	2.0	
21422	2061	Sist de Dirección	1.0	
21423	2061	Sist de Dirección	1.0	
21424	2061	Sist de Dirección	1.0	
21425	2061	Sist. Hidráulico	2.0	
21426	2061	Sist. Hidráulico	2.0	
21427	2061	Sist. Hidráulico	2.0	
21428	2061	Transmisión	1.0	
21429	2061	Transmisión	2.0	
21430	2061	Transmisión	1.0	
21431	2061	Transmisión	2.0	
21432	(21432 row(s) affected)	NaN	NaN	

	Horas Funcionamiento	Horas del Aceite	St	Al	Fe	Si	Na
0	21950.0	409.0	0.0	0.0	15.0	7.0	8.0
1	23153.0	424.0	0.0	0.0	55.0	9.0	9.0
2	22349.0	399.0	NaN	28.0	117.0	109.0	16.0
3	21950.0	409.0	57.0	3.0	74.0	6.0	15.0
4	22349.0	808.0	NaN	2.0	3.0	6.0	4.0
5	21950.0	409.0	NaN	1.0	8.0	22.0	4.0
6	28295.0	NaN	NaN	1.0	164.0	18.0	3.0
7	28295.0	NaN	0.0	0.0	11.0	8.0	5.0
8	27624.0	663.0	NaN	0.0	9.0	15.0	0.0
9	27624.0	258.0	NaN	1.0	2.0	4.0	4.0
10	27624.0	187.0	NaN	1.0	407.0	13.0	19.0
11	28377.0	82.0	NaN	1.0	58.0	11.0	0.0
12	27876.0	263.0	24.0	1.0	12.0	2.0	8.0
13	43742.0	448.0	NaN	1.0	6.8	9.1	3.6
14	47386.0	512.0	NaN	2.7	7.2	7.1	2.6
15	48300.0	914.0	NaN	0.1	5.6	8.3	3.7

16	44460.0	NaN	NaN	0.0	20.0	5.0	7.0
17	43505.0	2618.0	NaN	0.1	43.0	4.3	6.7
18	48626.0	326.0	NaN	0.1	15.1	3.3	5.8
19	44006.0	3119.0	NaN	0.1	39.9	0.8	8.1
20	46670.0	2374.0	NaN	0.1	39.1	5.3	8.2
21	48626.0	326.0	NaN	0.1	16.1	3.0	5.6
22	48940.0	640.0	NaN	0.1	22.7	1.4	5.5
23	21950.0	409.0	0.0	0.0	16.0	7.0	9.0
24	22729.0	1188.0	0.0	0.0	35.0	10.0	8.0
25	21950.0	409.0	NaN	11.0	42.0	55.0	11.0
26	23153.0	424.0	NaN	0.0	15.0	10.0	5.0
27	22729.0	380.0	NaN	1.0	4.0	6.0	1.0
28	27624.0	258.0	NaN	1.0	9.0	10.0	2.0
29	28295.0	NaN	77.0	4.0	90.0	9.0	13.0
...	...	...	...	...	...	...	...
21403	30057.0	1037.0	NaN	0.0	50.0	1.0	4.0
21404	3315.0	1295.0	NaN	0.0	46.0	6.0	6.0
21405	30057.0	1037.0	NaN	0.0	49.0	1.0	4.0
21406	4131.0	2111.0	NaN	0.0	65.0	8.0	7.0
21407	4422.0	291.0	NaN	0.0	28.0	3.0	6.0
21408	477.0	221.0	NaN	0.0	17.0	8.0	6.0
21409	2275.0	255.0	NaN	0.0	30.0	6.0	5.0
21410	30057.0	1037.0	NaN	0.0	50.0	1.0	4.0
21411	256.0	256.0	NaN	0.0	28.0	36.0	6.0
21412	2275.0	255.0	NaN	0.0	33.0	4.0	4.0
21413	3872.0	285.0	NaN	0.0	38.0	2.0	4.0
21414	3587.0	530.0	NaN	1.0	52.0	8.0	4.0
21415	4131.0	544.0	NaN	0.0	48.0	5.0	5.0
21416	4422.0	291.0	NaN	0.0	25.0	3.0	5.0
21417	5164.0	496.0	NaN	0.0	41.0	3.0	3.0
21418	2020.0	511.0	40.0	2.0	65.0	5.0	17.0
21419	2541.0	521.0	53.0	2.0	69.0	6.0	9.0
21420	2733.0	192.0	18.0	3.0	23.0	8.0	7.0
21421	4668.0	537.0	72.0	2.0	47.0	6.0	6.0
21422	758.0	502.0	NaN	1.0	2.0	3.0	5.0
21423	1781.0	272.0	NaN	1.0	2.0	2.0	5.0
21424	2020.0	272.0	NaN	1.0	2.0	3.0	5.0
21425	256.0	256.0	NaN	1.0	11.0	21.0	3.0
21426	3872.0	3395.0	NaN	2.0	11.0	22.0	8.0
21427	4668.0	4191.0	NaN	2.0	10.0	13.0	6.0
21428	758.0	281.0	NaN	1.0	4.0	3.0	4.0
21429	2020.0	511.0	NaN	1.0	4.0	2.0	3.0
21430	30057.0	516.0	NaN	0.0	4.0	0.0	1.0
21431	4915.0	247.0	NaN	1.0	4.0	2.0	2.0
21432	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[21433 rows x 10 columns]

Observamos que la última fila es inutil, entonces la eliminamos

2) Obtener el tipo de dato de cada feature

- Verificar si las fechas son tomasdas como fechas y en caso contrario convertir a fecha
- Verificar si los números son tomados como números)

[4]:
```
dataset.dtypes
```

[4]:
```
Equipo                  object
Componente              object
Id                     float64
Resultado              float64
Horas Funcionamiento   float64
Horas del Aceite       float64
Fecha de Análisis       object
B                      float64
Nit                    float64
Oxi                    float64
Sul                    float64
St                     float64
V                      float64
Al                     float64
Cr                     float64
Cu                     float64
Fe                     float64
Pb                     float64
Mo                     float64
Ni                     float64
Sn                     float64
Si                     float64
K                      float64
Na                     float64
W                       object
F                       object
A                       object
ISO                     object
PQI                    float64
Ba                     float64
Ca                     float64
Mg                     float64
Mn                     float64
P                      float64
Zn                     float64
Ag                     float64
Ti                     float64
V40                    float64
V100                   float64
TBN                    float64
```

```
TAN                     float64
ISO14                   float64
ISO4                    float64
ISO6                    float64
dtype: object
```

Observamos que los números son números con decimales, exceptuando "equipo", "componente", que son objetos. Y la fecha de análisi que es tomada como objeto y no como fecha. (No tenemos en cuenta W,F,A)

[5]:
```
# Convertimos a la "Fecha de Análisis" en formato datatime

dataset2 = pd.read_csv('OilDataSetCuracion.csv', parse_dates= ["Fecha de␣
 ↪Análisis"])
dataset2.dtypes
```

[5]:
```
Equipo                          object
Componente                      object
Id                             float64
Resultado                      float64
Horas Funcionamiento           float64
Horas del Aceite               float64
Fecha de Análisis       datetime64[ns]
B                              float64
Nit                            float64
Oxi                            float64
Sul                            float64
St                             float64
V                              float64
Al                             float64
Cr                             float64
Cu                             float64
Fe                             float64
Pb                             float64
Mo                             float64
Ni                             float64
Sn                             float64
Si                             float64
K                              float64
Na                             float64
W                               object
F                               object
A                               object
ISO                             object
PQI                            float64
Ba                             float64
Ca                             float64
Mg                             float64
Mn                             float64
P                              float64
```

```
Zn                                  float64
Ag                                  float64
Ti                                  float64
V40                                 float64
V100                                float64
TBN                                 float64
TAN                                 float64
ISO14                               float64
ISO4                                float64
ISO6                                float64
dtype: object
```

3) Verificar entre que fechas hay muestras de aceite

[6]:
```
primer_analisis = dataset2[['Fecha de Análisis']].min()
ultimo_analisis = dataset2[['Fecha de Análisis']].max()
display('El primer análisis se registra en: ',primer_analisis[0], 'El último␣
 ↪análisis se registra en: ',ultimo_analisis[0])
```

```
'El primer análisis se registra en: '


Timestamp('2017-10-29 00:00:00')


'El último análisis se registra en: '


Timestamp('2019-06-03 00:00:00')
```

4) Verificar si existen registros duplicados

[7]:
```
dataset2[dataset2.duplicated()]
```

[7]:
```
Empty DataFrame
Columns: [Equipo, Componente, Id, Resultado, Horas Funcionamiento, Horas del
Aceite, Fecha de Análisis, B, Nit, Oxi, Sul, St, V, Al, Cr, Cu, Fe, Pb, Mo, Ni,
Sn, Si, K, Na, W, F, A, ISO, PQI, Ba, Ca, Mg, Mn, P, Zn, Ag, Ti, V40, V100, TBN,
TAN, ISO14, ISO4, ISO6]
Index: []

[0 rows x 44 columns]
```

[8]:
```
dataset2[dataset2.index.duplicated()]
```

[8]:
```
Empty DataFrame
Columns: [Equipo, Componente, Id, Resultado, Horas Funcionamiento, Horas del
Aceite, Fecha de Análisis, B, Nit, Oxi, Sul, St, V, Al, Cr, Cu, Fe, Pb, Mo, Ni,
```

```
Sn, Si, K, Na, W, F, A, ISO, PQI, Ba, Ca, Mg, Mn, P, Zn, Ag, Ti, V40, V100, TBN,
TAN, ISO14, ISO4, ISO6]
Index: []

[0 rows x 44 columns]
```

No se observan datos duplicados

5) Despersonalización de Datos.

Vamos a suponer que los id de los camiones que se presetan en el datast son datos que permiten identificar a los mismos y por un tema de confidencialidad necestiamos despersonalizar este dato para hacer público su análisis. Aplicar dos técnicas de despersonalización de datos sobre el dataset

[9]:
```
#Datos sin anonimizar
display(dataset2[['Id']].head(5))
```

```
         Id
0  273615.0
1  294763.0
2  278700.0
3  273586.0
4  278549.0
```

[10]:
```
#Creamos una funcion para anonimizar
def anoni1 (column):
    return ((column/3333)-np.mod(13,7))
```

[11]:
```
#Anonimizamos los datos
dataset2[['Id']].apply(anoni1).head(5)
```

[11]:
```
           Id
0  76.092709
1  82.437744
2  77.618362
3  76.084008
4  77.573057
```

[12]:
```
#Creamos otra funcion para anonimizar
def anoni2 (column):
    return ((column/2222)-np.mod(14,6))
```

[13]:
```
dataset2[['Id']].apply(anoni2).head(5)
```

[13]:
```
            Id
0  121.139064
1  130.656616
2  123.427543
3  121.126013
```

```
4  123.359586
```

[14]:
```
#Le aplicamos la anonimidad a todo el data set
dataset2[['Id']] = dataset2[['Id']].apply(anoni1)
```

Vale aclarar que existen diferentes (infinitas) formas de "Anonimizar" los datos, ya que a fin de cuenta lo que se aplica es una 'función' que transforme los mismos. Ej: MD5,SHAx,HILL...

6) Revisar las etiquetas de los feature y renombrar aquellas que contengan caracteres "no ascci" que pudieran traer probleams en el procesamiento

[15]:
```
columnas_antes = dataset2.columns.astype('unicode')
columnas_antes.values
```

[15]:
```
array(['Equipo', 'Componente', 'Id', 'Resultado', 'Horas Funcionamiento',
       'Horas del Aceite', 'Fecha de Análisis', 'B', 'Nit', 'Oxi', 'Sul',
       'St', 'V', 'Al', 'Cr', 'Cu', 'Fe', 'Pb', 'Mo', 'Ni', 'Sn', 'Si',
       'K', 'Na', 'W', 'F', 'A', 'ISO', 'PQI', 'Ba', 'Ca', 'Mg', 'Mn',
       'P', 'Zn', 'Ag', 'Ti', 'V40', 'V100', 'TBN', 'TAN', 'ISO14',
       'ISO4', 'ISO6'], dtype=object)
```

[16]:
```
#Modificamos las columnas a "ascii", remplazando los caracteres que nos traerán␣
 ↪problemas con un signo de pregunta '?'
columnas_despues = columnas_antes.str.encode("ascii", errors = "replace")
columnas_despues.values
```

[16]:
```
array([b'Equipo', b'Componente', b'Id', b'Resultado',
       b'Horas Funcionamiento', b'Horas del Aceite', b'Fecha de An?lisis',
       b'B', b'Nit', b'Oxi', b'Sul', b'St', b'V', b'Al', b'Cr', b'Cu',
       b'Fe', b'Pb', b'Mo', b'Ni', b'Sn', b'Si', b'K', b'Na', b'W', b'F',
       b'A', b'ISO', b'PQI', b'Ba', b'Ca', b'Mg', b'Mn', b'P', b'Zn',
       b'Ag', b'Ti', b'V40', b'V100', b'TBN', b'TAN', b'ISO14', b'ISO4',
       b'ISO6'], dtype=object)
```

7)Valores faltantes

Determinar que porcentaje de datos faltantes existen en el feature Fe y Na Completar los datos faltantes con la media

[17]:
```
dataset[['Id','Fe','Na']].count()
```

[17]:
```
Id    21432
Fe    20693
Na    20690
dtype: int64
```

[18]:
```
dataset[dataset.duplicated(['Id'])]
```

[18]:
```
Empty DataFrame
Columns: [Equipo, Componente, Id, Resultado, Horas Funcionamiento, Horas del
Aceite, Fecha de Análisis, B, Nit, Oxi, Sul, St, V, Al, Cr, Cu, Fe, Pb, Mo, Ni,
Sn, Si, K, Na, W, F, A, ISO, PQI, Ba, Ca, Mg, Mn, P, Zn, Ag, Ti, V40, V100, TBN,
TAN, ISO14, ISO4, ISO6]
Index: []
```

```
[0 rows x 44 columns]
```

[19]:
```
total_datos = dataset[['Id']].count().values[0]+1
total_Fe = dataset[['Fe']].dropna().count().values[0]
total_Na = dataset[['Na']].dropna().count().values[0]

porc_falt_Fe = abs(((100*total_Fe)/total_datos)-100)
porc_falt_Na = abs(((100*total_Na)/total_datos)-100)
display('Total de datos',total_datos, 'Total de datos de Hiero', total_Fe,␣
 ↪'Porcentaje faltante de datos de Hierro (Fe)', porc_falt_Fe, 'Total de datos␣
 ↪de Sodio',total_Na, 'Porcentaje faltante de datos de Sodio␣
 ↪(Na)',porc_falt_Na)
```

```
'Total de datos'
```

```
21433
```

```
'Total de datos de Hiero'
```

```
20693
```

```
'Porcentaje faltante de datos de Hierro (Fe)'
```

```
3.452619791909669
```

```
'Total de datos de Sodio'
```

```
20690
```

```
'Porcentaje faltante de datos de Sodio (Na)'
```

```
3.46661689917417
```

[20]:
```
#Completamos los datos faltantes de hierro y Sodio con sus respectivas medias
display(dataset[['Na']].fillna(dataset[['Na']].mean()).dropna().
 ↪count(),dataset[['Fe']].fillna(dataset[['Fe']].mean()).dropna().count())
dataset2[['Na']] = dataset[['Na']].fillna(dataset[['Na']].mean()).dropna()
dataset2[['Fe']] = dataset[['Fe']].fillna(dataset[['Fe']].mean()).dropna()
```

```
Na    21433
dtype: int64


Fe    21433
dtype: int64
```

[21]:
```
#'Eliminamos' la última fila y guardamos
dataset2 = dataset2.iloc[:21432,]
dataset2.to_csv('dataset_intro.csv')
```

Eliminamos la última fila (solo nos aporta ruido), y guardamos nuestro dataset para utilizarlo en el próximo práctico