In [8]:

```
%reload_ext watermark
%watermark
```

```
2019-05-30T21:35:03+02:00

CPython 3.6.5
IPython 6.4.0

compiler   : GCC 7.2.0
system     : Linux
release    : 5.1.5-arch1-2-ARCH
machine    : x86_64
processor  :
CPU cores  : 4
interpreter: 64bit
```

Procesado de DataSet

Con lo visto en el apartado anterior de procesado de variables vamos a transformar un dataset en otro que pueda ser utilizado para entrenar modelos de predicción.

Ingesta de datos

Cargamos las librerías que vamos a utilizar y el dataset original

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import feature_extraction
```

In [2]:

```
datos = pd.read_csv("../../RESOURCES/datos_procesamiento.csv")
datos.head()
```

Out[2]:

	col_inexistente1	col2	col3	col_outliers	col_outliers2	col_categorica	col_ordinal	col_texto
0	59.0	52.0	2.232832	-50	0.771666	ratón	muy bien	Tenía en su casa una ama que pasaba de los cua...
1	31.0	74.0	0.906147	-5	1.068558	elefante	regular	El resto della concluían sayo de velarte, calz...
2	81.0	28.0	0.626750	-32	0.846396	ratón	muy mal	El resto della concluían sayo de velarte, calz...
3	34.0	16.0	0.816738	-84	0.637381	gato	mal	Una olla de algo más vaca que carnero, salpicó...
4	32.0	28.0	0.571131	65	4.540614	gato	bien	Tenía en su casa una ama que pasaba de los cua...

Transformación del DataSet

Separación de variables

In [3]:

```
col_numericas =  ['col_inexistente1', 'col2', 'col3', 'col_outliers', 'col_outliers2']
col_categorica = ['col_categorica']
col_texto = ['col_texto']
```

Variables numéricas

In [4]:

```
imputador = preprocessing.Imputer(strategy="mean")
escalador = preprocessing.StandardScaler()
var_numericas_imputadas_escalado_standard = escalador.fit_transform(
                                                imputador.fit_transform(datos[col_numericas])
                                            )
df_numerico_procesado = pd.DataFrame(var_numericas_imputadas_escalado_standard,
                                                 columns=col_numericas)
```

```
/opt/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:58: DeprecationWarning: Cl
ass Imputer is deprecated; Imputer was deprecated in version 0.20 and will be removed in 0.22.
Import impute.SimpleImputer from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

Variables Categóricas

In [5]:

```
label_codificador_categorico = preprocessing.LabelEncoder()
categorias_codificadas = label_codificador_categorico.fit_transform(datos[col_categorica])
oh_codificador = preprocessing.OneHotEncoder(sparse=False)
categorias_oh_codificadas = oh_codificador.fit_transform(categorias_codificadas.reshape(1000,1))

df_categorico_procesado = pd.DataFrame(categorias_oh_codificadas,
                                       columns=label_codificador_categorico.classes_)
```

```
/opt/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:235:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
e shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/opt/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/_encoders.py:371: FutureWarning:
The handling of integer data will change in version 0.22. Currently, the categories are determined
based on the range [0, max(values)], while in the future they will be determined based on the
unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, t
hen you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

Variables de Texto

In [6]:

```
vectorizador_tfidf = feature_extraction.text.TfidfVectorizer()
texto_vectorizado = vectorizador_tfidf.fit_transform(datos.col_texto)
df_texto_procesado =  pd.DataFrame(texto_vectorizado.toarray(),
columns=vectorizador_tfidf.get_feature_names())
```

Exportación y muestra final

In [7]:

```
datos_procesados = pd.concat([
    df_numerico_procesado,
    df_categorico_procesado,
    df_texto_procesado
], axis=1)

label_codificador_ordinal = preprocessing.LabelEncoder()
datos_procesados['col_ordinal'] = label_codificador_ordinal.fit_transform(datos.col_ordinal)
datos_procesados.head()
```

Out[7]:

	col_inexistente1	col2	col3	col_outliers	col_outliers2	elefante	gato	perro	ratón	acordarme	...	vaca	
0	0.399217	0.082807	0.442819	-0.694600	-0.038365	0.0	0.0	0.0	1.0	0.0	...	0.000000	0.2
1	-0.653605	0.861333	-0.323390	-0.118466	-0.038278	1.0	0.0	0.0	0.0	0.0	...	0.000000	0.0

	col_inexistente1	col2	col3	col_outliers	col_outliers2	elefante	gato	perro	ratón	acordarme	...	vaca	
2	[illegible]	-0.766494	-0.484752	[illegible]	[illegible]	0.0	0.0	0.0	1.0	0.0	...	0.000000	0.0
3	-0.540803	-1.191145	-0.375028	-1.129901	-0.038405	0.0	1.0	0.0	0.0	0.0	...	0.194272	0.0
4	-0.616004	-0.766494	-0.516874	0.777743	-0.037257	0.0	1.0	0.0	0.0	0.0	...	0.000000	0.2

5 rows × 144 columns

In [17]:

```
datos_procesados.to_csv("../../RESOURCES/dataset_procesado.csv")
```