```
SANDBOX_NAME =  'fesc' # Sandbox Name
DATA_PATH = "/data/sandboxes/"+SANDBOX_NAME+"/data/"


#para leer desde drive se hace así:
from google.colab import drive



drive.mount('/content/drive',force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4

Enter your authorization code:
··········
Mounted at /content/drive
```

Workshop de Python

Cargamos datasets con información de admisiones a hospitales de enfermos de diabetes. El objetivo es, una vez limpiado el dataset, estudiarlo para extraer el máximo número de insights de los datos.

Cargar librerías

```
%matplotlib inline

import re
import random
from collections import Counter

import pandas as pd
pd.set_option('display.max_colwidth', -1)
import numpy as np
import matplotlib.pyplot as plt
#import seaborn as sns
#from scipy.stats import kstest
#sns.set(color_codes=True)
```

Lectura de datos

```
#diabetes = spark.read.csv(DATA_PATH+'diabetic_data.csv', sep=',', header=True, inferSchema=True).toPandas()
#admission_source = spark.read.csv(DATA_PATH+'admission_source_id.csv', sep=',', header=True, inferSchema=True).toPandas()
#admission_type = spark.read.csv(DATA_PATH+'admission_type_id.csv', sep=',', header=True, inferSchema=True).toPandas()
#discharge_disposition = spark.read.csv(DATA_PATH+'discharge_disposition_id.csv', sep=',', header=True, inferSchema=True).toPandas()
diabetes = pd.read_csv('/content/drive/My Drive/WS 20190411/data/diabetic_data.csv', sep=',')
admission_source = pd.read_csv('/content/drive/My Drive/WS 20190411/data/admission_source_id.csv', sep=',')
admission_type = pd.read_csv('/content/drive/My Drive/WS 20190411/data/admission_type_id.csv', sep=',')
discharge_disposition = pd.read_csv('/content/drive/My Drive/WS 20190411/data/discharge_disposition_id.csv', sep=',')
```

Comprobar correcta lectura de los datos

```
diabetes.head()
```

```
diabetes.columns
```

```
diabetes.dtypes
```

```
diabetes.shape
```

```
diabetes.info()
```

```
diabetes['admission_type_id'].unique()
```

```
diabetes['admission_source_id'].unique()
```

```
#admission_source
admission_source.head()
```

```
admission_source.shape
```

```
admission_source.dtypes
```

```
admission_source.info()
```

```
admission_source['admission_source_id'].unique()
```

```
admission_source['admission_source_id'].nunique()
```

```
#ADMISSION_TYPE
admission_type.head()
```

```
admission_type.dtypes
```

```
admission_type.info()
```

```
admission_type['admission_type_id'].nunique()
```

```
admission_type['admission_type_id'].unique()
```

```
admission_type.tail()
```

```
#discharge_disposition
```

```
discharge_disposition.head()
```

```
discharge_disposition.dtypes
```

```
discharge_disposition.info()
```

```
discharge_disposition['discharge_disposition_id'].unique()
```

```
discharge_disposition['discharge_disposition_id'].nunique()
```

Juntar todos los datos en el mismo DataFrame

```
#Junto la tabla diabetes con admission_source
paso1=diabetes.merge(admission_source, on='admission_source_id', how='left')
paso1.shape #diabetes tenía (101766, 50) y al añadir la columna de admission_source tenemos (101766, 51)


#diabetes
#admission_type_id           int64
#discharge_disposition_id    int64
#admission_source_id         int64

#admission_source
#admission_source_id    int64
#description
```

```
#renombro la columna descripcion para saber qué es
paso1.rename(columns= {'description' : 'descrip_adm_source'}, inplace=True)
paso1.head(2)
```

```
#Junto la tabla que tiene diabetes y admission_source con
paso2=paso1.merge(admission_type, on='admission_type_id', how='left')
paso2.head(2)
#paso1.shape #diabetes tenía (101766, 50) y al añadir la columna de admission_source tenemos (101766, 51)

#diabetes
#admission_type_id           int64
#discharge_disposition_id    int64
#admission_source_id         int64

#admission_type
#admission_type_id    int64
#description          object
#dtype: object
```

```
paso2.shape #(101766, 51) y al juntar los datos de la tercera tabla pasamos a 52 columnas
```

```
paso2.rename(columns= {'description' : 'descrip_adm_type'}, inplace=True)
paso2.head(2)
```

```
#Junto la tabla que tiene diabetes y admission_source con
data_all=paso2.merge(discharge_disposition, on='discharge_disposition_id', how='left')
data_all.head(2)


#diabetes
#admission_type_id           int64
#discharge_disposition_id    int64
#admission_source_id
#discharge_disposition_id    int64
#description                 object
#dtype: object
```

```
data_all.rename(columns= {'description' : 'descrip_discharge_dispo'}, inplace=True)
data_all.head(2)
```

Estudiar las dimensiones del dataset

```
data_all.shape
```

Data Wrangling

Cambiar los nombres de las columnas para que cumplan buenas prácticas

```
data_all.columns
```

```
#data_all_lo=data_all.rename(columns = lambda x: x.lower())
#data_all_lo.head(2)
data_all_re=data_all.rename(columns = lambda x: x.lower().replace('-','_'))
data_all_re.columns
```

Estudiar el formato de las variables, ver cuáles se deberían modificar y modificarlas cuando se considere oportuno

```
data_all_re.head(2)
```

```
data_all_re.dtypes
```

```
#Las variables id que sean numéricas, las convertimos en categóricas y también patient_nbr que es el identificador de cada paciente.
data_all_re['encounter']=data_all_re['encounter_id'].apply(lambda x: str(x))
data_all_re['admission_type']=data_all_re['admission_type_id'].apply(lambda x: str(x))
data_all_re['discharge_disposition']=data_all_re['discharge_disposition_id'].apply(lambda x: str(x))
data_all_re['admission_source']=data_all_re['admission_source_id'].apply(lambda x: str(x))
data_all_re['patient_nbr_str']=data_all_re['patient_nbr'].apply(lambda x: str(x))
#data_all_re.head(2)
data_all_re.dtypes
#encounter_id
#admission_type_id
#discharge_disposition_id
#admission_source_id
```

```
#Eliminamos las vbles que hemos convertido a categóricas
data_all_re2=data_all_re.drop(columns=['encounter_id','admission_type_id','discharge_disposition_id','admission_source_id','patient_nb
```

```
data_all_re['weight'].unique()
```

Estudiar si hay registros repetidos

```
data_all_re2.duplicated().any() #No hay registros duplicados
```

Estudiar si hay variables que siempre o prácticamente siempre toman el mismo valor: tomar siempre el mismo valor no aporta información al modelo

```
list(set(data_all_re2.dtypes.tolist())) #creamos una lista con los formatos diferentes que tiene el dataframe
```

```
#crear un dataframe para las variables cuantitativas
numerical=data_all_re2.select_dtypes(include=['int64']) #dataframe con los valores numéricos
numerical.head(2)
```

```
#vemos como se distribuyen los valores
numerical.hist(figsize=(15, 20), bins=60, xlabelsize=10, ylabelsize=10); #esto no es obligatorio
```

```
#Calculamos para cada columna los valores que contiene y qué % tiene cada valor
columnas_cuanti=numerical.columns
for i in columnas_cuanti:
  print((data_all_re2[i].value_counts()/data_all_re2.shape[0])*100)
```

```
#crear dataframe para las variables cualitativas
cualitativas=data_all_re2.select_dtypes(include=['object']) #dataframe con los valores numéricos
cualitativas.head(2)
```

```
#Guardamos las columnas en una variable para utilizarlas más adelante
columnas_cuali=cualitativas.columns
```

```
#Calculamos para cada columna los valores que contiene y qué % tiene cada valor
for i in columnas_cuali:
  print((data_all_re2[i].value_counts()/data_all_re2.shape[0])*100)

#Analizamos el % de nulos para cada campo
#Eliminamos el peso porque el 97% de los registros no están informados.
#payer_code, hay un 40% sin informar
#medial_speciality, hay un 49% sin informar
#max_glu_serum tiene un 95% de None
#a1cresult tiene un 86% de None
```

Estudiar los nulos en el dataset

```
data_all_re2.isnull().any() #No hay ningún registro con nulos
```

```
#Tras el análisis de las variables categóricas decimimos eliminar lo siguiente:

#categóricas
#weigh porque tiene el 97% de ?
#payer_code porque tiene el 40% de ?
#medical_speciality poruqe tiene el 50% de ?
#max_glu_serum porque tiene None en un 95%
#A1crsult en un 86%
```

```
data_all_sincolum=data_all_re2.drop(columns=['weight','a1cresult','payer_code','medical_specialty','max_glu_serum'])
data_all_sincolum.shape
data_all_sincolum.head(2)
```

Estudiar los outliers

```
#Hacemos análisis de outliers para las variables cuantitativas
columnas_cuanti
```

```
#Obtenemos un histograma para ver la distribución de las vbles cuantitativas a grandes rasgos

for i in columnas_cuanti:
  plt.hist(data_all_sincolum[i])
  plt.xlabel(i)
  plt.ylabel('quantity')
  str_title='Analysis of: ' + i.upper()
  plt.title(str_title)
  plt.show()
```

```
#Función que se queda con los outliers
def tukey_outliers(df,column,extreme=False):
    q1, q3 = np.percentile(df[column],[25,75])#q1 es 25 y q3 es 75 y df[column] es la columna
    iqr = q3 - q1
    constant = 1.5 if not extreme else 3
    return df[((df[column]>(q3+constant*iqr)) | (df[column]<(q1-constant*iqr)))]


columnas_cuanti
```

```
#Creamos un diccionario con las columnas cuantitativas y los outliers que tiene cada una de ellas
di = {}
for i in columnas_cuanti:
  di[i] = tukey_outliers(data_all_sincolum,i,extreme=False).shape[0]

di
#sacar los valores de los outliers
```

```
data_all_sincolum.shape[0]
```

```
#En este análisis vemos:
#- Las variables que tienen outlierslas
#- Vemos cuántos outliers tiene cada variable
#- También vemos 2 variables posibles candidatas a eliminar los outliers del data frame compelto, pero antes tenemos que ver si los
#valores que son outliers tienen sentido o no


#Creamos un diccionario con las columnas cuantitativas y los outliers que tiene cada una de ellas
di = {}
di2 = {}
for i in columnas_cuanti:
  di[i] = tukey_outliers(data_all_sincolum,i,extreme=False).shape[0]
  di2[i] = tukey_outliers(data_all_sincolum,i,extreme=False)[i].value_counts()

di2
#sacar los valores de los outliers
```

```
#Una vez analizados los valores outliers concluimos que no eliminamos los procedimientos de laboratio y el número de diagnósticos porqu
```

Crear variables dummy en caso que sea conveniente

```
# Creamos un dataframe con los dummis
df_dummy = pd.get_dummies(data_all_sincolum['age'])
df_dummy
```

```
# Creamos un dataframe con los dummis
df_dummy = pd.get_dummies(data_all_sincolum['age'])
# Y los concatenamos con nuestro dataframe
df_total = pd.concat([data_all_sincolum, df_dummy], axis=1)
df_total.head(2)
```

Estudiar que todas las variables ahora sí tengan el formato y contenido adecuado. si todavía no, arreglarlas.

```
data_all_sincolum.head(2)
```

```
data_all_sincolum.dtypes
```

```
data_all_sincolum.info()
```

```
data_all_sincolum.describe()
```

Antes de estudiar el comportamiento de las variables, veremos si hay alguna variable derivada a crear

```
#La variable derivada a crear es la edad
#vemos los tramos de edad
data_all_sincolum['age'].unique()
```

```
def func_tramos_edad(x):
  tramos_edad='0'
  if x == '[0-10)':
    tramos_edad='niños'
  elif x == '[10-20)':
    tramos_edad='adolescentes'
  elif x == '[20-30)':
    tramos_edad='jóvenes'
  elif x == '[30-40)':
    tramos_edad='nuevos jóvenes'
  elif x == '[40-50)':
    tramos_edad='adultos'
  elif x == '[50-60)':
    tramos_edad='prejubilables'
  elif x == '[60-70)':
    tramos_edad='jubilables'
  elif x=='[70-80)':
```

```
    tramos_edad='jubilados'
  elif x=='[80-90)':
    tramos_edad='octogenarios'
  else:
    tramos_edad='nonagenarios'
  return tramos_edad
```

```
data_all_sincolum['periodo_vital']=data_all_sincolum['age'].apply(func_tramos_edad)
data_all_sincolum.head(2)
```

Exploratory Data Analysis

Crear la matriz de correlación

```
matriz_correlacion=data_all_sincolum.corr()
```

Pintar la matriz de correlación

```
matriz_correlacion
```

Determinar qué par de variables están correlacionadas

```
corr_matrix = data_all_sincolum.corr().abs()

sol = (corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
                  .stack())
sol[sol > 0.25]
```

```
#Tabla Final: Nos quedamos con los mismos registros iniciales y hemos añadido 3 columnas para las descripciones, hemos añadido una vble
#y hemos eliminado 5 variables que tenían un alto porcentaje de valores sin informar
data_all_sincolum.shape
```