

**UNIVERSIDAD EUROPEA
MIGUEL DE CERVANTES**

ESCUELA POLITÉCNICA SUPERIOR

**TITULACIÓN:
MÁSTER UNIVERSITARIO EN GESTIÓN Y ANÁLISIS
DE GRANDES VOLUMENES DE DATOS: BIG DATA**



TRABAJO FIN DE MÁSTER

Title of the project

AUTOR

Daniel García Teba

TUTOR

Miguel Angel Gomez Lopez

VALLADOLID, octubre de 2020

Índice de contenidos

uciones y planes de mej
uciones
o utilizado
ódigo Exploratory Data An
ódigo Red neuronal secuenc
ódigo Utilities
ías

Índice de figuras

3.1. Número de muestras de gas por Batch	10
3.2. Número de muestras de cada gas en total	11

Índice de tablas

3.1. Distribución de los lotes a lo largo del tiempo.	9
---	---

Capítulo 1

Objetivos del trabajo

Dados unos datos experimentales provenientes de una nariz electrónica, **insertar aquí enlace UCI data** se desea hacer un estudio de la viabilidad de clasificar correctamente de qué gas se trata, e intentar dar una estimación de la concentración de dicho gas.

Los datos provienen de cada medición de un gas con 16 sensores. De esta forma, cada medición genera 16 series temporales, de las cuales de cada una se han extraído 8 *features*. Esto hace un total de 128 componentes para cada medición. Para cada medición es conocido el gas que se ha ensayado y su nivel de concentración.

Con este tipo de información, se va a estudiar los resultados que pueden ofrecer las redes neuronales para resolver este problema.

En el caso que nos ocupa, los sensores derivan a lo largo del tiempo, lo cual implica que para el mismo gas, con la misma concentración, las series temporales obtenidas en un mes dado podrían ser diferentes a las realizadas con las mismas condiciones meses después.

En este trabajo se tomará como primera aproximación dividir la tarea de clasificación y la tarea de estimación.

Para la tarea de clasificación, se probarán las siguientes configuraciones, de más simple a más compleja:

- ▷ Perceptron
- ▷ **Long short-term memory Neural net**

Una vez conseguida la tarea de clasificación, se usará otra red neuronal para determinar, si es posible, la concentración del gas.

En este trabajo se utilizará Keras y TensorFlow para el diseño de las redes neuronales, en el entorno Python.

DRAFT 12/10/2020

Capítulo 2

Análisis de la situación

De los estudios realizados en *Chemical gas sensor drift compensation using classifier ensembles* ?? se deriva que los sensores utilizados derivan a lo largo del tiempo.

Esto presenta un reto para entrenar un algoritmo de clasificación. En este trabajo primero comprobaremos que existe esta deriva de los sensores, para posteriormente aplicar algoritmos que sean capaces de hacer buenas predicciones a partir de los datos disponibles.

Capítulo 3

Obtención, procesamiento y almacenamiento de los datos

Los datos provienen del artículo Chemical gas sensor drift compensation using classifier ensembles (?),

donde el objetivo era tratar de detectar el drift (la deriva) de los sensores a lo largo de los meses, y poder calibrarlos utilizando el mínimo número de experimentos posibles. Es decir, de la forma más rápida y eficiente posible. (ver (?))

Los datos están disponibles para su descarga desde UCI data repository. en 10 archivos formato *.dat*.

Cada lote cuenta con una estructura de 129 columnas, donde la primera nos informa del gas y la concentración, y el resto es la información obtenida del sensor.

El primer paso que se va a realizar es, dada las 128 componentes X, averiguar a qué tipo de gas pertenece la medición.

Una vez la red es capaz de inferir correctamente de qué gas se trata, alimentaremos otra red cuya función sea averiguar la concentración del mismo.

3.0.1 Procesado

La lectura de los archivos *.dat* se ha realizado utilizando el código adjunto en Apéndices ??

3.0.2 Exploratory Data Analysis

Los datos se nos presentan en 10 lotes, correspondientes a experimentos a lo largo de tres años, donde se ensayaron 6 diferentes gases a diferentes concentraciones.

Batch ID	Month IDs
Batch 1	Months 1 and 2
Batch 2	Months 3, 4, 8, 9 and 10
Batch 3	Months 11, 12, and 13
Batch 4	Months 14 and 15
Batch 5	Month 16
Batch 6	Months 17, 18, 19, and 20
Batch 7	Month 21
Batch 8	Months 22 and 23
Batch 9	Months 24 and 30
Batch 10	Month 36

Tabla 3.1: Distribución de los lotes a lo largo del tiempo.

Los gases que se estudiaron son los siguientes:

1. Ethanol
2. Ethylene
3. Ammonia
4. Acetaldehyde
5. Acetone
6. Toluene

Los lotes contienen una cantidad de muestras desigual, ni los 6 gases de estudio están presentes en todos los lotes.

La tabla ?? muestra el numero de ensayos para cada gas.

En la Tabla?? podemos ver que el número de muestras en cada lote es desigual. En los lotes 3, 4 y 5 el gas 6 no está presente. A la hora de crear un dataset de entrenamiento, convendría generar un lote donde haya un numero equitativo de muestras de todos los gases, y si las mediciones no son distantes en el tiempo podremos ver el efecto de la deriva si el algoritmo entrenado con los primeros lotes falla para cada vez más conforme nos alejamos en el tiempo.

Batch ID	1	2	3	4	5	6
1	90	98	83	30	70	74
2	164	334	100	109	532	5
3	365	490	216	240	275	0
4	64	43	12	30	12	0
5	28	40	20	46	63	0
6	514	574	110	29	606	467
7	649	662	360	744	630	568
8	30	30	40	33	143	18
9	61	55	100	75	78	101
10	600	600	600	600	600	600

La Figura 3.1 muestra la cantidad de gases ensayados por lote, mientras que la Figura3.2 muestra el numero de mediciones totales sobre cada gas.

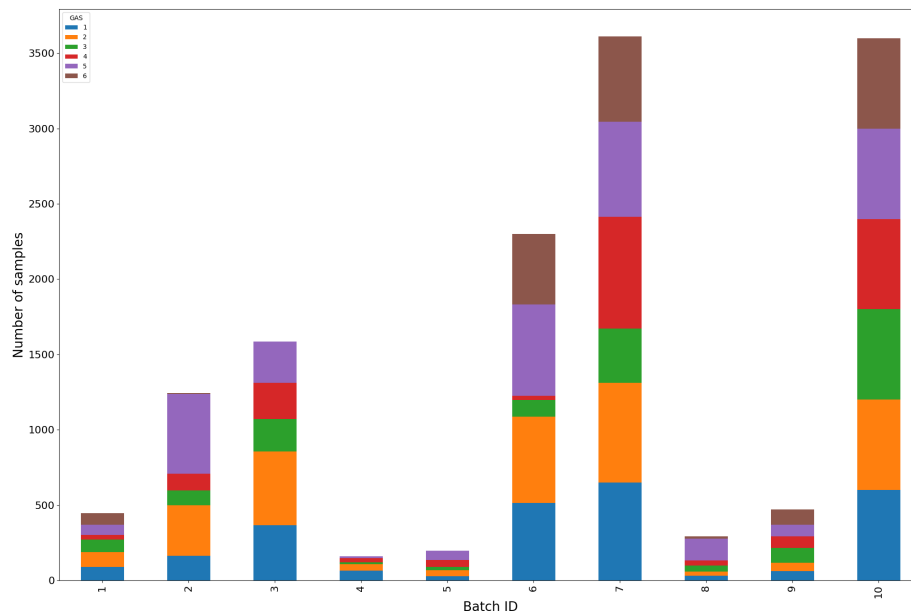


Figura 3.1: Número de muestras de gas por Batch

Si observamos los rangos de concentración para cada gas, han sido también diferentes.

Esta información es necesario tenerla en cuenta a la hora de entrenar nuestro

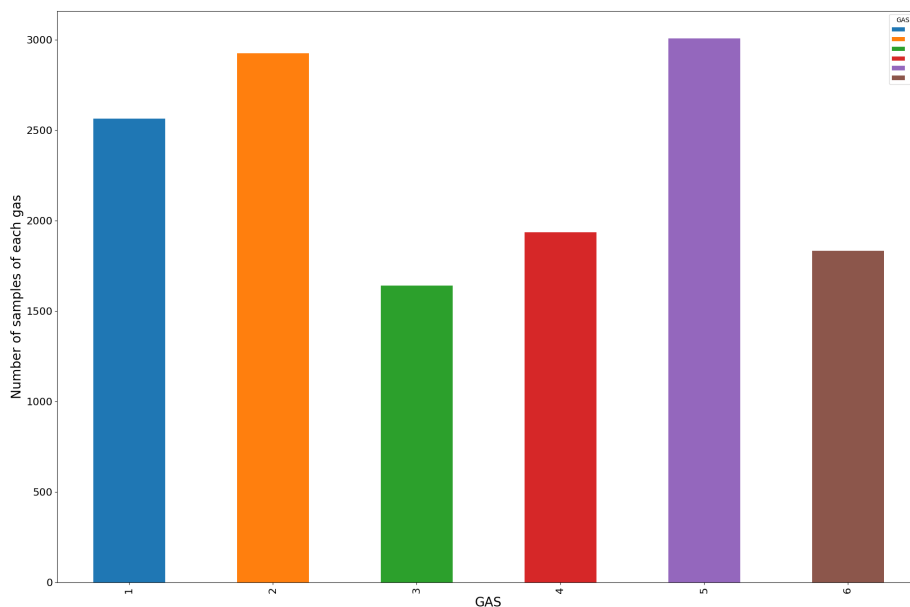


Figura 3.2: Número de muestras de cada gas en total

modelo, ya que si el rango de variación de los datos es dispar, será recomendable normalizar.

GAS	Minimo	Máximo	Media	StdDesv
1	2.5	600	114.95	86.64
2	2.5	300	116.1	79.89
3	2.5	1000	323.55	272.02
4	2.5	300	126.32	76.71
5	10	1000	228.57	217.38
6	1	230	47.66	32.58

Capítulo 4

Diseño e implementación de los modelos o técnicas necesarias

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Capítulo 5

Análisis de los resultados obtenidos con los modelos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Capítulo 6

Conclusiones y planes de mejora

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Capítulo 7

Conclusiones

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet

vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

DRAFT 12/10/2020

Apéndice A

Código utilizado

A.1 Código Exploratory Data Analysis

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 from python.LoadUciData import LoadDatFolder
5 from python.StandardFigure import save_figure
6 pd.set_option('display.max_columns', 10)
7
8
9 def plot_count_per_batch_and_gas(df_gas):
10     props = df_gas.groupby("Batch ID")['GAS'].value_counts(normalize=False).unstack()
11     ax = props.plot(kind='bar', stacked=True, figsize=(24, 16))
12     ax.set_ylabel('Number of samples')
13     save_figure(plt.gcf(), 'Step0_Count_Batch_Gas')
14
15
16 def plot_sample_count_per_gas(df_gas):
17     props = df_gas.groupby('GAS')['GAS'].value_counts(normalize=False).unstack()
18     ax = props.plot(kind='bar', stacked=True, figsize=(24, 16))
19     ax.set_ylabel('Number of samples of each gas')
20     save_figure(plt.gcf(), 'Step0_Count_Gas')
21
22
23 if __name__ == '__main__':
24
25     # Load .dat files
26     folder = r'data_uci/driftdataset'
27     df_gas = LoadDatFolder(folder).df
28
29     ## Tables
```

```
30 # Show samples per Batch and Gas
31 gas_batch_group = pd.crosstab(df_gas['Batch ID'], df_gas['GAS']).
sort_index()
32 print('\n', gas_batch_group.to_markdown())
33
34 # Show concentration range statistics per GAS
35 pivot = pd.pivot_table(df_gas,
36                         index=['GAS'],
37                         values='CONCENTRATION',
38                         aggfunc=['min', 'max', 'mean', 'std', '
count'])
39 pivot.round(2)
40 print('\n', pivot.round(2).to_markdown())
41
42 ## Plots
43 # Show samples count per GAS
44 plot_count_per_batch_and_gas(df_gas)
45 plot_sample_count_per_gas(df_gas)
```

Listing A.1: Realiza el analisis exploratorio de los datos disponibles

A.2 Código Red neuronal secuencial

```
1 import tensorflow as tf
2 from tensorflow import keras
3 from tensorflow.python.keras.callbacks import TensorBoard
4 from time import time
5 import os
6
7
8 class SeqModel:
9     """ Sequential Neutal Net."""
10     FOLDER = 'models/sequential'
11
12     def __init__(self):
13         pass
14
15     def save_model(self, name):
16         self.model.save(os.path.join(self.FOLDER, name))
17
18     def load_model(self, name):
19         self.model = keras.models.load_model(os.path.join(self.FOLDER
, name))
20
21     def _gen_model_seq(self):
22         model = keras.Sequential([
23             keras.layers.Flatten(input_shape=(self.x_size, 1)),
24             keras.layers.Dense(32, activation='relu'),
```

```

25         keras.layers.Dense(32, activation='relu'),
26         keras.layers.Dense(self.y_size)
27     ])
28     return model
29
30     def _gen_and_complile_model(self):
31         model = self._gen_model_seq()
32         print(model.summary())
33         model.compile(optimizer='adam',
34                       loss=tf.keras.losses.CategoricalCrossentropy(
35 from_logits=True),
36                       metrics=['accuracy'])
37         return model
38
39     def model_train(self, X_train, y_train):
40         # TensorFlow and tf.keras
41         self.x_size = X_train.shape[1]
42         self.y_size = y_train.shape[1]
43         model = self._gen_and_complile_model()
44         tb = TensorBoard(log_dir="logs/{}".format(time()))
45         model.fit(X_train, y_train, epochs=30, callbacks=[tb])
46         self.model = model
47
48     def model_evaluate(self, X_test, y_test):
49         model = self.model
50         test_loss, test_acc = model.evaluate(X_test, y_test, verbose
51 =2)
52         print('\nTest accuracy:', test_acc)
53         return test_loss, test_acc
54
55     def get_model(self):
56         return self.model

```

Listing A.2: Red neuronal secuencial

```

1 from abc import ABC, abstractmethod
2 import numpy as np
3 import pandas as pd
4
5 import tensorflow as tf
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import confusion_matrix
8
9 import matplotlib.pyplot as plt
10 import seaborn as sns
11
12 from python.LoadUciData import load_data
13 from python.StandardFigure import save_figure
14 from python.SeqModel import SeqModel

```

```

15
16
17 class AbstractSequentialModel(ABC):
18
19     @abstractmethod
20     def split_data(self):
21         return self.X_train, self.X_test, self.y_train, self.y_test
22
23     def train_and_save_model(self, modelname):
24         seq = SeqModel()
25         seq.model_train(self.X_train, self.y_train)
26         seq.save_model(modelname)
27
28
29 class SeqModelSimple(AbstractSequentialModel):
30
31     def __init__(self, df):
32         self.df = df
33
34     def split_data(self):
35         ## First, we will NOT use concentration data
36         gas_X = df_gas.drop(columns=['Batch ID', 'GAS', '
CONCENTRATION']).to_numpy()
37         gas_y = pd.get_dummies(df_gas['GAS'], drop_first=False)
38         self.X_train, self.X_test, self.y_train, self.y_test =
train_test_split(gas_X, gas_y,
39                                                         test_size
=0.33,
40
random_state=42)
41         return self.X_train, self.X_test, self.y_train, self.y_test
42
43
44 class SeqModelWithConcentration(AbstractSequentialModel):
45
46     def __init__(self, df):
47         self.df = df
48
49     def split_data(self):
50         ## First, we will NOT use concentration data
51         gas_X = df_gas.drop(columns=['Batch ID', 'GAS']).to_numpy()
52         gas_y = pd.get_dummies(df_gas['GAS'], drop_first=False)
53         self.X_train, self.X_test, self.y_train, self.y_test =
train_test_split(gas_X, gas_y, test_size=0.33, random_state=42)
54         return self.X_train, self.X_test, self.y_train, self.y_test
55
56
57 def plot_conf(model, X_test, y_test):
58

```

```

59 # Plot the confusion matrix
60 f = plt.figure(figsize=(8, 6))
61 # First add a Softmax layer to get probabilities.
62 probability_model = tf.keras.Sequential([model, tf.keras.layers.
Softmax()])
63 predictions = probability_model.predict(X_test)
64 y_test_values = np.argmax(y_test.values, axis=1)
65 gas_predictions = np.argmax(predictions, axis=1)
66 confusion = confusion_matrix(y_test_values, gas_predictions)
67
68 label_gas = ['Gas1', 'Gas2', 'Gas3', 'Gas4', 'Gas5', 'Gas6']
69 df_conf = pd.DataFrame(data=confusion, columns=label_gas, index=
label_gas)
70 sns.heatmap(df_conf, annot=True, fmt="d", cmap='Blues')
71 plt.title('Confusion Matrix')
72 plt.xticks(rotation=0)
73 plt.show()
74 return f
75
76
77 if __name__ == '__main__':
78
79 # Check the results for the sequential Neural Net
80 # Load data
81 df_gas = load_data()
82
83 mod1 = SeqModelSimple(df_gas)
84 X_train, X_test, y_train, y_test = mod1.split_data()
85 #mod1.train_and_save_model('ModelSimple')
86
87 mod2 = SeqModelWithConcentration(df_gas)
88 X_train2, X_test2, y_train2, y_test2 = mod2.split_data()
89 #mod2.train_and_save_model('ModelWithConcentration')
90
91
92 seq = SeqModel()
93 seq.load_model('ModelSimple')
94 f = plot_conf(seq.get_model(), X_test, y_test)
95 save_figure(f, 'ConfMatrix_ModelSimple')
96
97 seq = SeqModel()
98 seq.load_model('ModelWithConcentration')
99 f = plot_conf(seq.get_model(), X_test2, y_test2)
100 save_figure(f, 'ConfMatrix_ModelWithConcentration')

```

Listing A.3: Red neuronal secuencial para clasificar gases

```

1 from sklearn.model_selection import train_test_split
2

```

```
3 from python.LoadUciData import load_data
4
5 if __name__ == '__main__':
6
7     # Load data
8     df_gas = load_data()
9
10    df_train_reg = df_gas[df_gas['GAS'] == 1]
11
12    gas_X = df_train_reg.drop(columns=['Batch ID', 'GAS', '
13    CONCENTRATION']).to_numpy()
14    gas_y = df_train_reg['CONCENTRATION'].to_numpy()
15
16    X_train, X_test, y_train, y_test = train_test_split(gas_X, gas_y,
17                                                         test_size
18                                                         =0.33,
19                                                         random_state
20                                                         =42)
```

Listing A.4: Red neuronal secuencial para estimar la concentración

A.3 Código Utilities

```
1 import os
2 import pandas as pd
3 import re
4 import pickle
5 from python.FileUtils import get_list_of_files_with_extension
6
7
8 class LoadDatFile:
9     """
10     This class aims to load a .dat files from UCI
11     https://archive.ics.uci.edu/ml//datasets/Gas+Sensor+Array+Drift+
12     Dataset
13     , and returns a pandas.dataframe object
14
15     :arg .dat file
16     :return df
17     """
18
19     def __init__(self, file):
20         self.file = file
21
22     @property
23     def batch_number(self):
24         base = os.path.basename(self.file)
25         name, ext = os.path.splitext(base)
```

```

25         num = re.findall(r'\d+', name)[0]
26         # num = num.zfill(2)
27         return int(num)
28
29     @property
30     def df(self):
31         df = pd.read_table(self.file, engine='python', sep='\s+\d+: ',
32                             header=None)
33         df['Batch ID'] = self.batch_number
34         return df
35
36 class GasDataFrame:
37     """ Process the .dat file to get all the information contained:
38     - Gas, concentration and measures. """
39
40     def __init__(self, file):
41         self.file = file
42
43     @property
44     def df(self):
45         df_raw = LoadDatFile(self.file).df
46         return self._add_gas_info(df_raw)
47
48     @staticmethod
49     def _add_gas_info(df):
50         df[['GAS', 'CONCENTRATION']] = df.iloc[:, 0].str.split(";",
51                                     expand=True, )
52         df.drop(df.columns[0], axis=1, inplace=True)
53         df['GAS'] = df['GAS'].astype('int')
54         df['CONCENTRATION'] = df['CONCENTRATION'].astype('float')
55         return df
56
57 class LoadDatFolder:
58     """
59     This class aims to load all .dat files contained in a folder,
60     gives each file a GasDataframe format and concats all in a pandas
61     .dataframe object with
62
63     :inputs: folder with many .dat files
64     :return df
65     """
66
67     def __init__(self, folder):
68         self.folder = folder
69
70     @property
71     def df(self):
72         files = get_list_of_files_with_extension(self.folder, 'dat')
73         df_full = pd.DataFrame()
74         for f in files:

```

```
71         dftemp = GasDataFrame(f).df
72         df_full = df_full.append(dftemp)
73         return df_full
74
75
76 def load_data():
77     folder = r'data_uci/driftdataset/'
78     df_gas = LoadDatFolder(folder).df
79     return df_gas
80
81 if __name__ == '__main__':
82     file_data = r'data_uci/driftdataset/batch1.dat'
83     lf = LoadDatFile(file_data)
84     my_dataframe = lf.df
85
86     gdf = GasDataFrame(file_data)
87     my_dataframe_gas = gdf.df
88
89     folder = r'data_uci/driftdataset/'
90     ldf = LoadDatFolder(folder)
91     my_dataframe_full = ldf.df
```

Listing A.5: Clases para cargar datos en memoria

DRAFT 12/10/2020

Referencias

Finazzi, I. F. (2020). Latex template for master's thesis of UEMC. *UEMC*, 1(1), 1.