Calidad del aire CDMX

Proyecto Final de Aprendizaje de
Máquina Profundo
Presentan:
Jorge III Altamirano Astorga,
Luz Aurora Hernández Martínez,
Ita-Andehui Santiago Castillejos.

Índice

i. Introducción
ii. Trabajo relacionado
iii. Solución
iv. Resultado
v. Conclusión

Introducción

Esta sección se divide en:

- Fuentes de datos del prospecto
- Problemáticas
- Variables
- EDA

Fuente de datos

Tenemos los siguientes de fuentes de datos:

- *Sensor Bosch BME680: contamos aproximadamente con casi 1.3 millones de registros con lecturas del sensor cada 3 segundos.*
- *Datos Abiertos de la Calidad del Aire del Gobierno de la Ciudad de México: datos por hora de las estaciones de monitoreo del Gobierno.*
- *Datos de otras estaciones meteorológicas de la Ciudad.*

Problemáticas

- *Datos del Gobierno de la Ciudad de México: estos datos no se actualizan de manera cotidiana*
- *Datos meteorológicos de terceros: No encontramos.*
- *Precisión y manipulación de los datos de nuestras fuentes de datos.*
- *Estabilidad y precisión de la toma de registros en el sensor. Pudiéramos tener interrupciones del suministro eléctrico que no permitieran obtener ciertas lecturas.*
- *Algoritmo cerrado del sensor para convertir de la variable gasResistance a la variable IAQ; el cual es cerrado.*

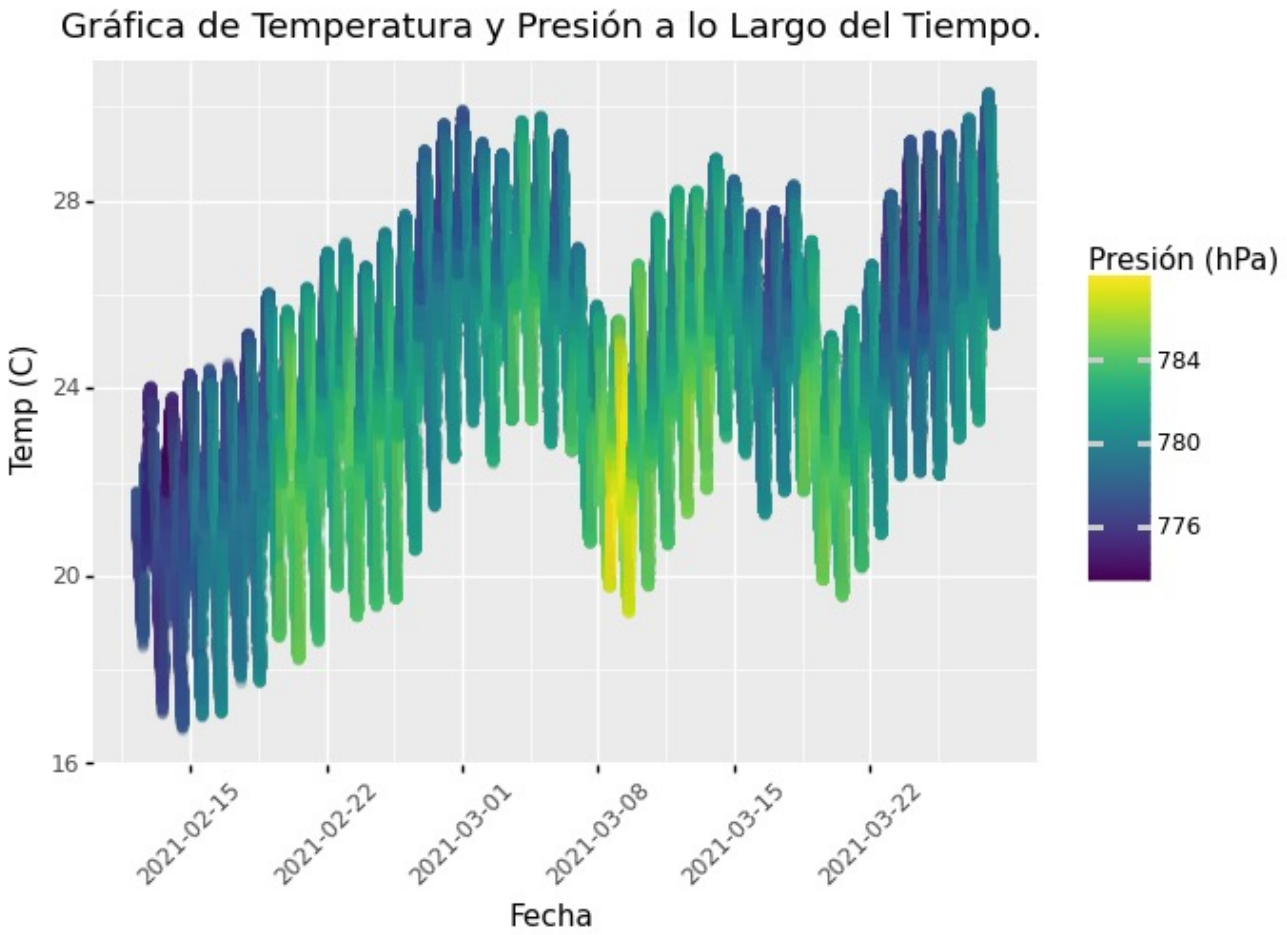
Variables

- ***Temperatura**: variable numérica en grados Celsius (C) con una resolución de 0.01C y una precisión de ±0.5C.*
- ***Presión**: variable numérica en hectopascales (hPa) con una resolución de 0.18 hPa y una precisión de ±0.12 hPa.*
- ***Humedad**: variable numérica en porcentaje de humedad relativa (%rH) con una resolución de 0.008%rH y una precisión de ±3%rH.*
- ***Resistencia del Gas**: variable numérica de la resistencia eléctrica opuesta al elemento sensible del sensor medida en Ohms.*

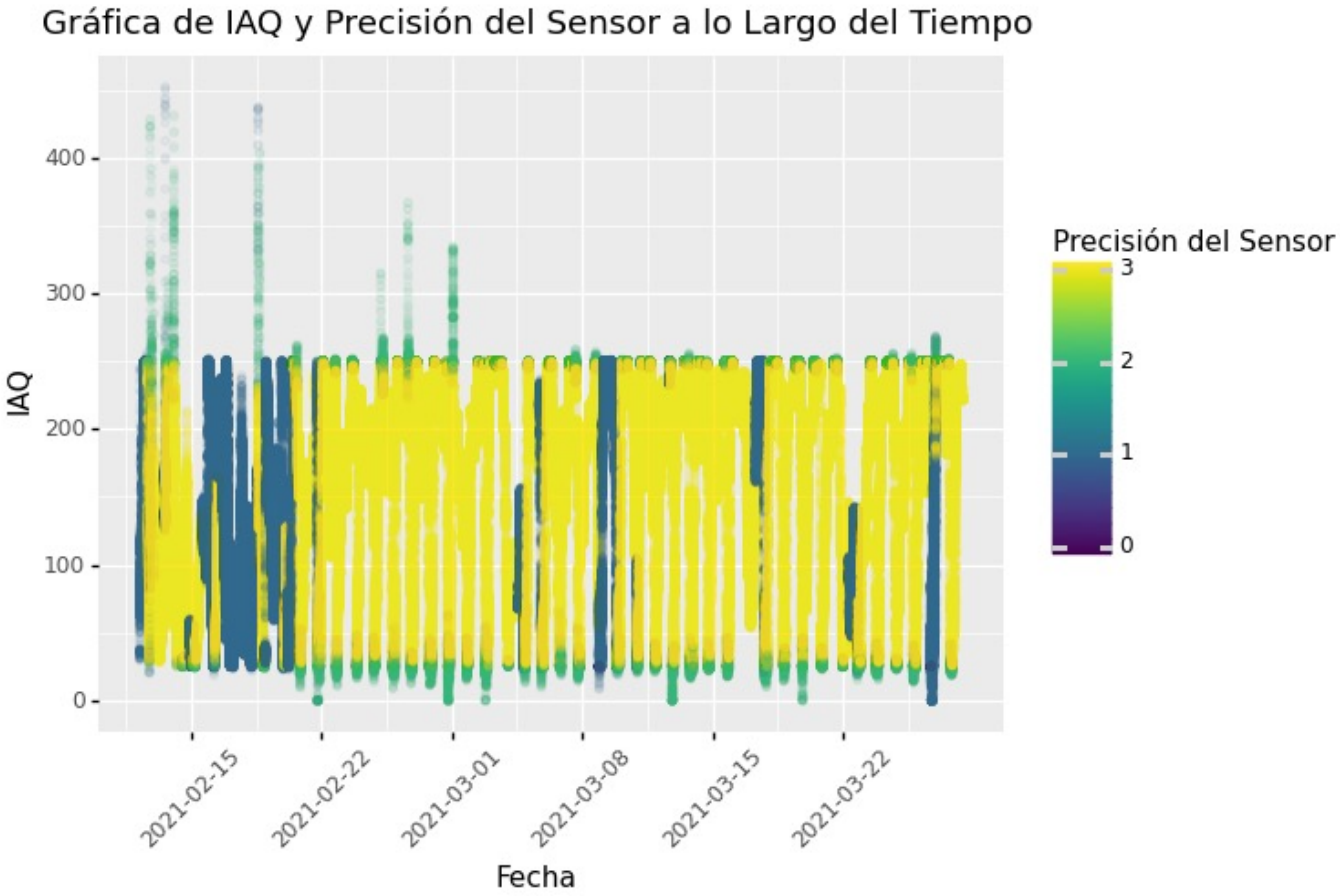
Variables

- ***IAQ**: variable numérica medida en el índice de calidad del aire americano en interior con una resolución de 1 IAQ. La precisión del sensor variable que no excede 5% se guarda en una variable independiente.*
- ***Precisión del sensor**: variable categórica ordinal con valores en el rango de [0,3]:*
  - *0: periodo de estabilización o no operativo.*
  - *1-2: periodo operativo.*
  - *3: precisión máxima y operación óptima.*
- ***Fecha y hora**: variable numérica basado en UNIX/POSIX epoch que denota el tiempo desde el 01/01/1970 00:00:00.0 UTC. El tiempo está sincronizado por NTP al Centro Nacional de Metrología de México (Hora Oficial del País).*

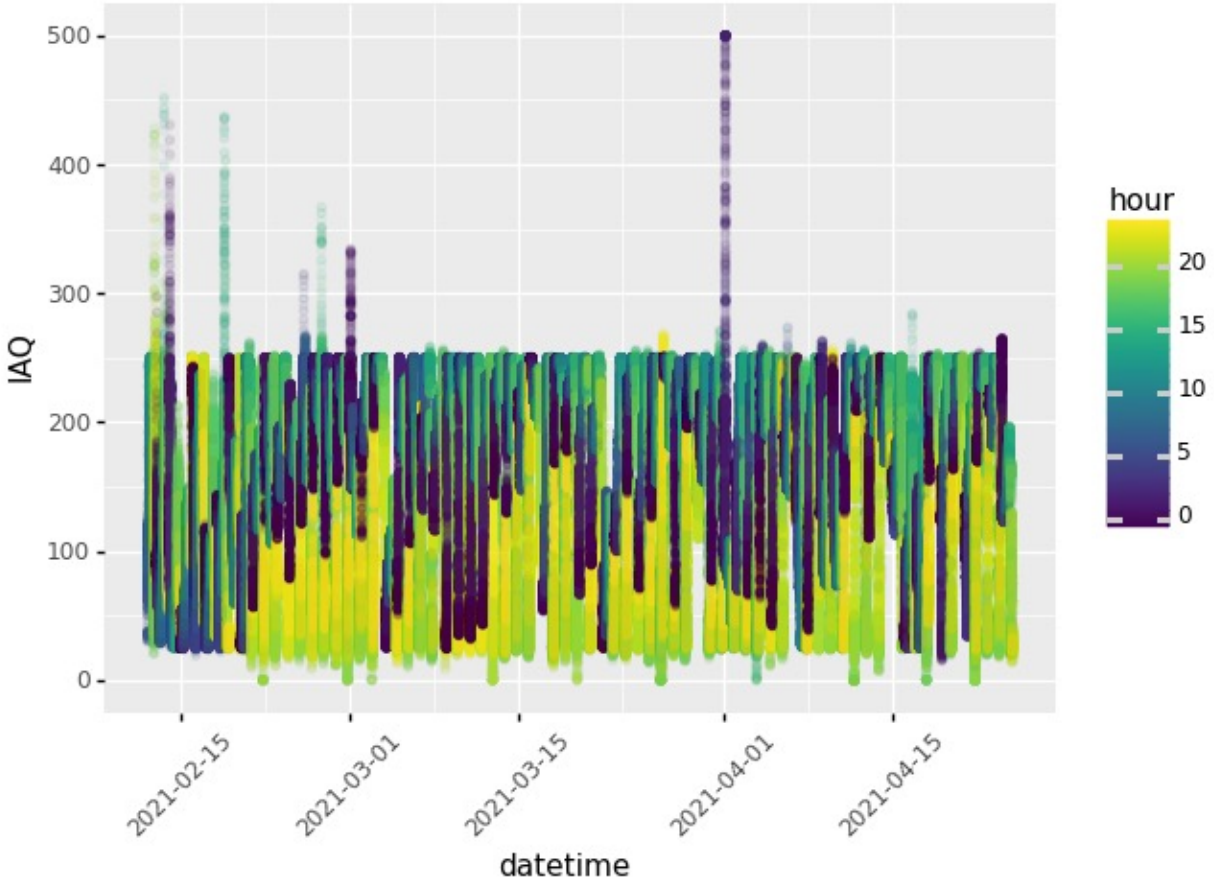
EDA


Gráfica de Temperatura y Presión a lo Largo del Tiempo.
Temp (C)
28
24
20
16
2021-02-15
2021-02-22
2021-03-01
2021-03-08
2021-03-15
2021-03-22
Fecha
Presión (hPa)
784
780
776

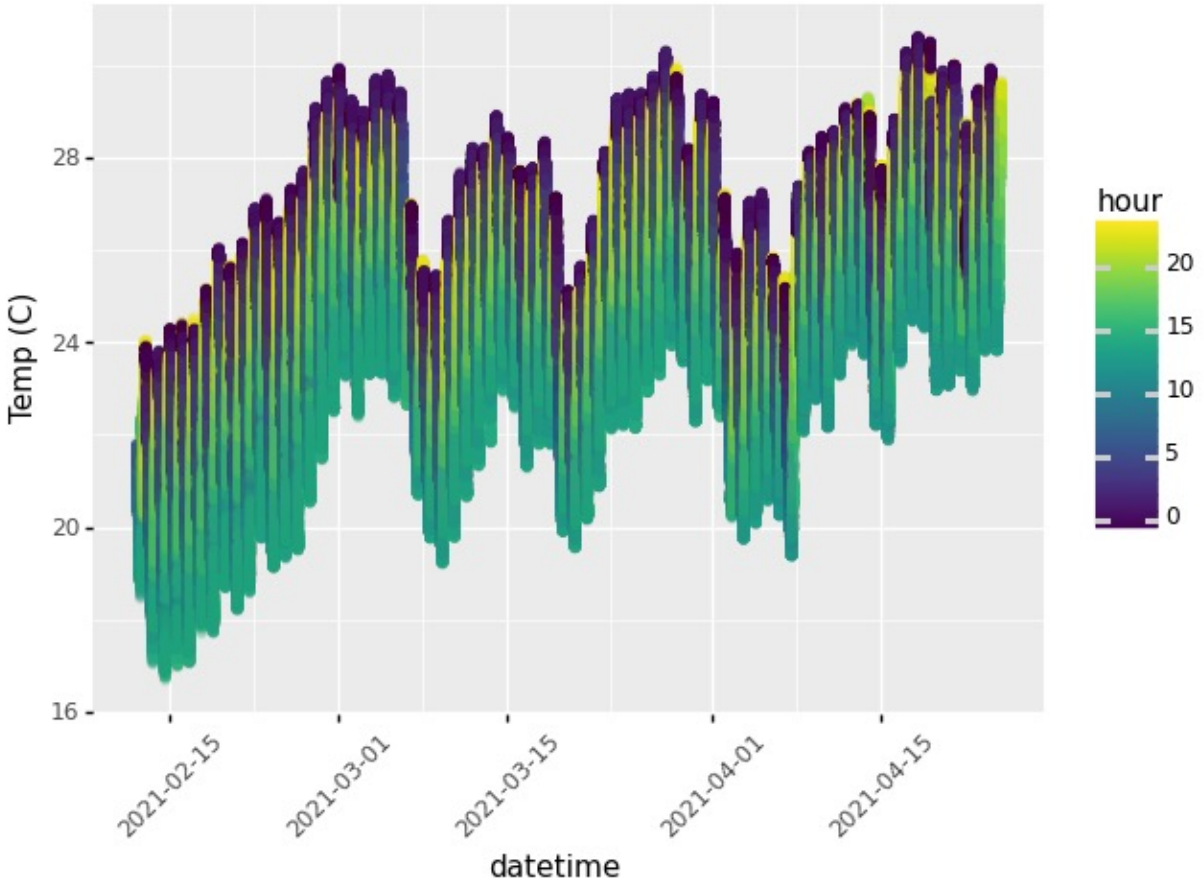
EDA


Gráfica de IAQ y Precisión del Sensor a lo Largo del Tiempo
IAQ
400
300
200
100
0
2021-02-15
2021-02-22
2021-03-01
2021-03-08
2021-03-15
2021-03-22
Fecha
Precisión del Sensor
3
2
1
0

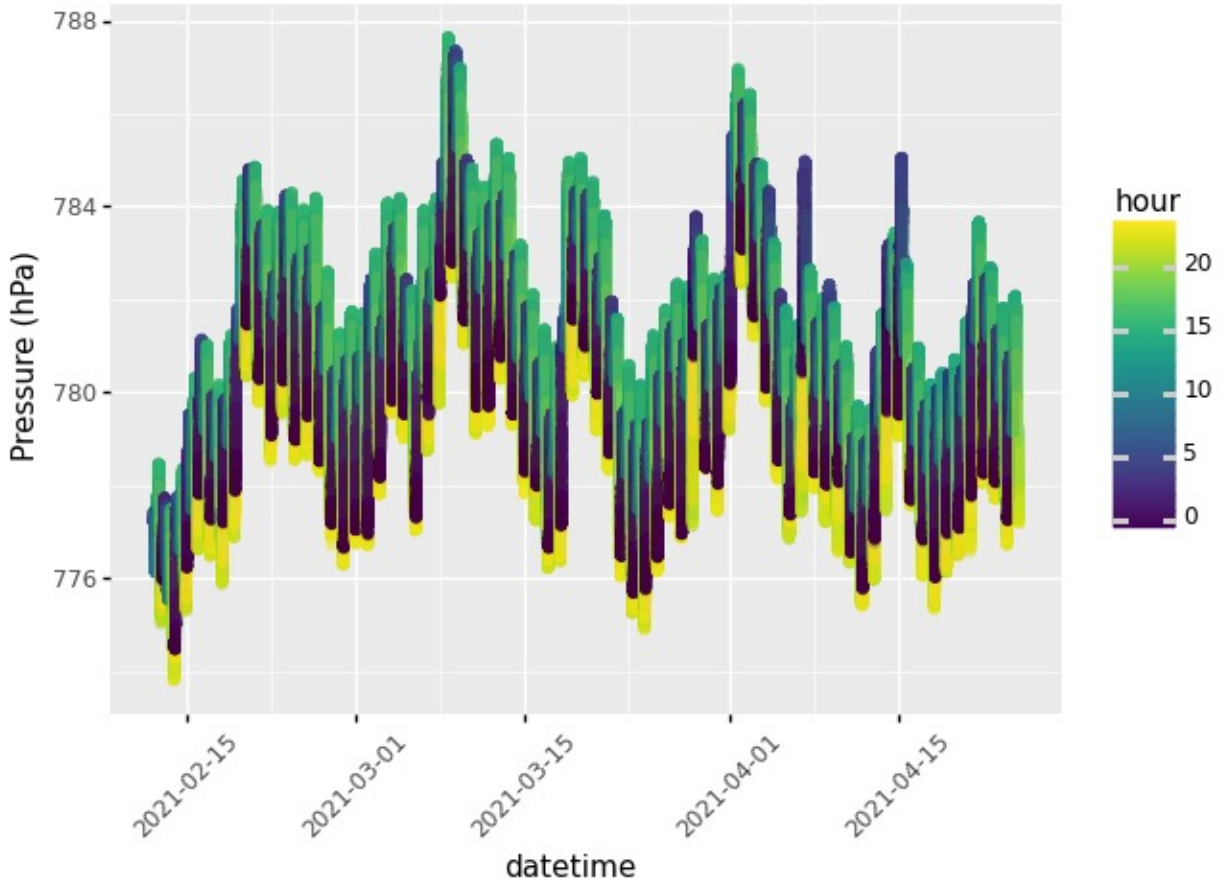
EDA


500
400
300
200
100
0
IAQ
2021-02-15
2021-03-01
2021-03-15
2021-04-01
2021-04-15
datetime
hour
20
15
10
5
0

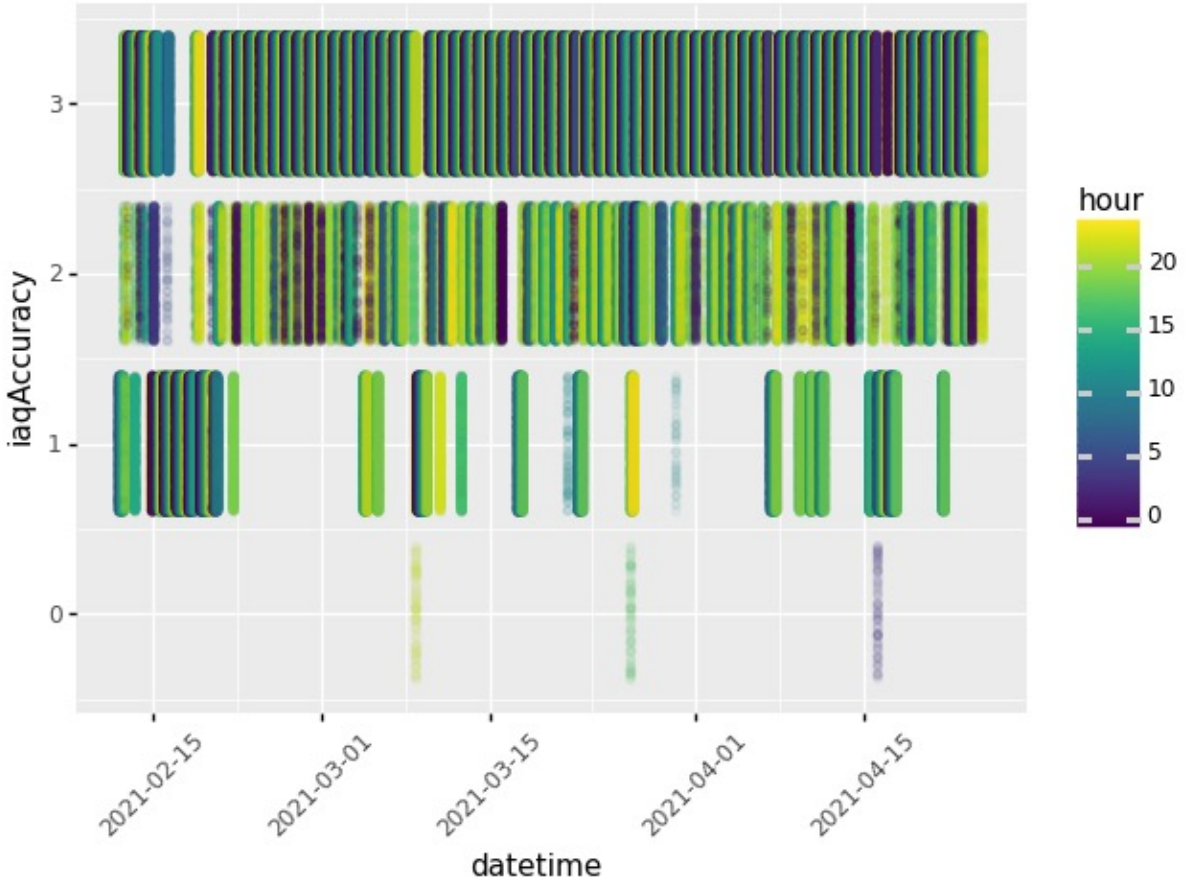
EDA


Temp (C)
28
24
20
16
2021-02-15
2021-03-01
2021-03-15
2021-04-01
2021-04-15
datetime
hour
20
15
10
5
0

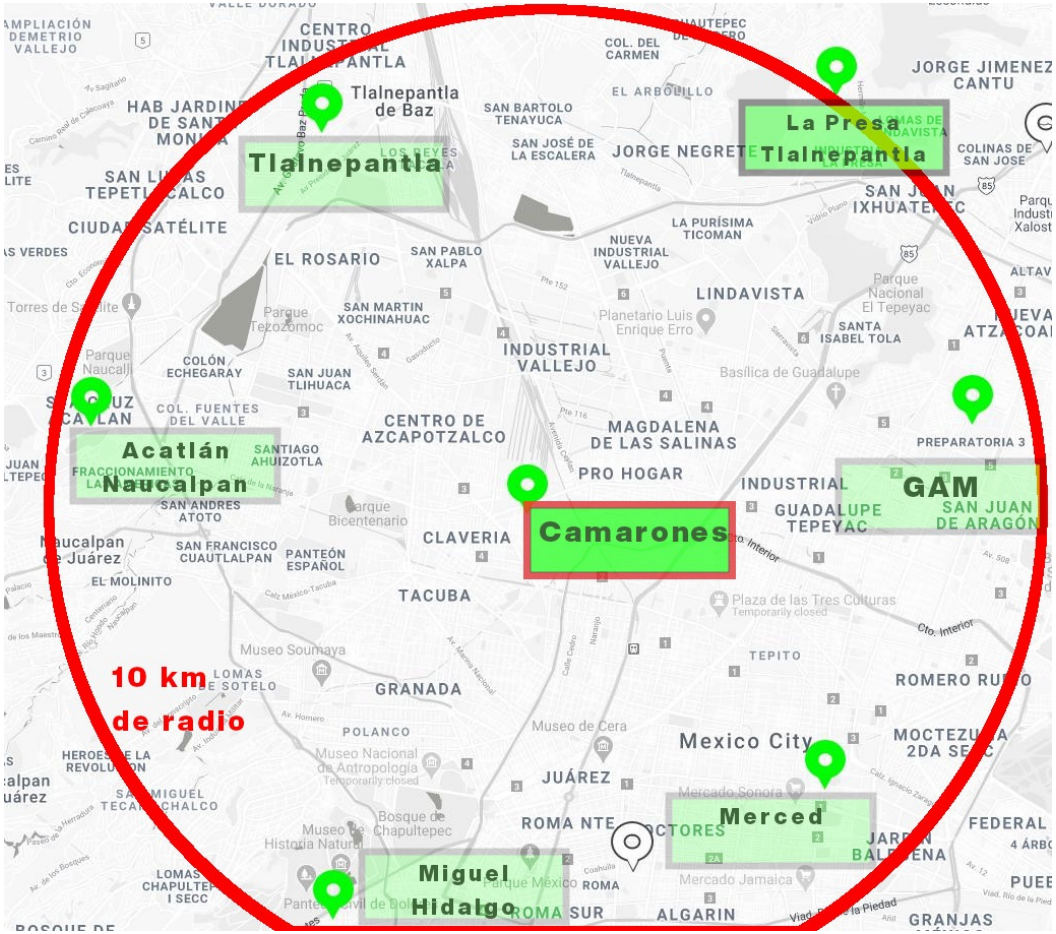
EDA


788
784
780
776
Pressure (hPa)
2021-02-15
2021-03-01
2021-03-15
2021-04-01
2021-04-15
datetime
hour
20
15
10
5
0

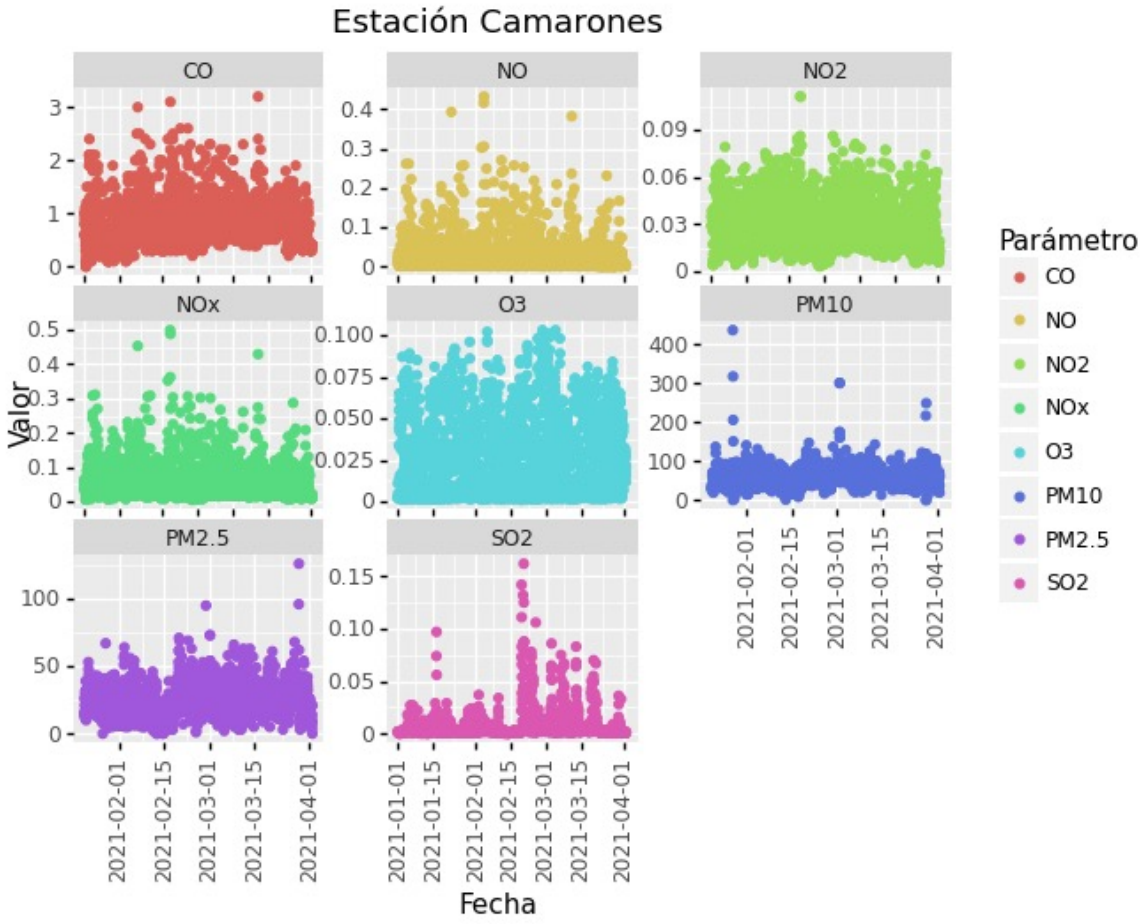
EDA


iaqAccuracy
3
2
1
0
2021-02-15
2021-03-01
2021-03-15
2021-04-01
2021-04-15
datetime
hour
20
15
10
5
0

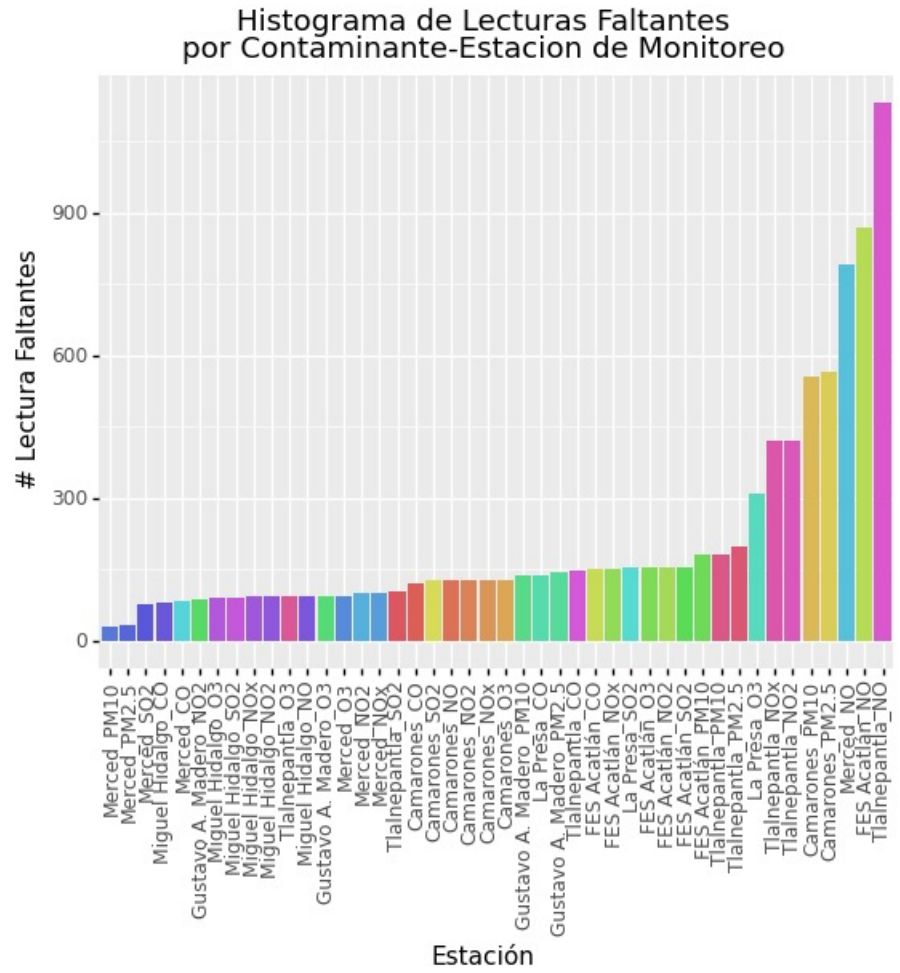
EDA: datos SINAICA

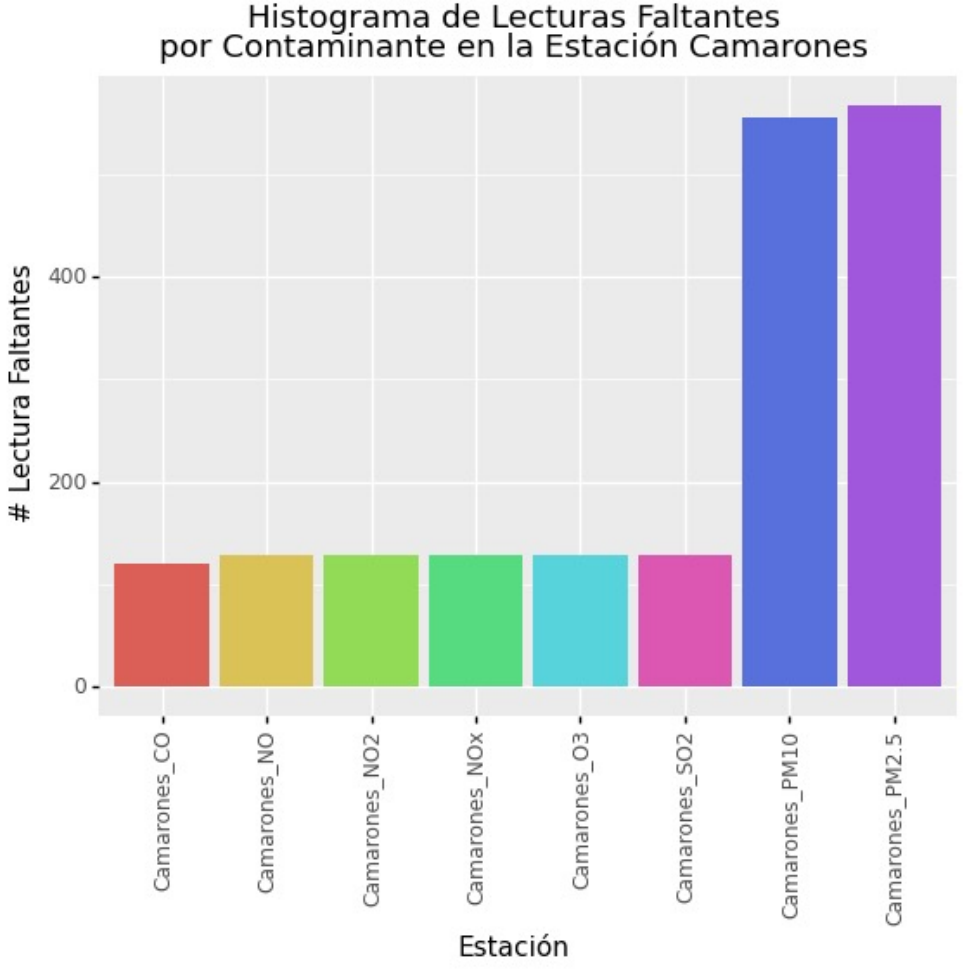


EDA: datos SINAICA



EDA: datos SINAICA





Trabajo relacionado

Hemos realizado algunos trabajos previos (1) y buscado artículos relacionados que describimos a continuación:

- *Examen final para la materia de "Modelos de Gran Escala" con la Prof. Liliana Millán, donde se estudiaron la relación de las estaciones de biciletas "Ecobici" con la calidad del aire en las inmediaciones.*

- *Development of indoor environmental index: Air quality index and thermal comfort index. Referido en la bibliografía.*

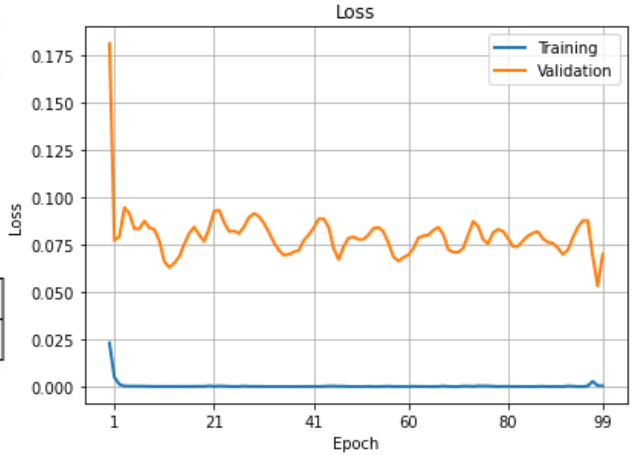
Solución

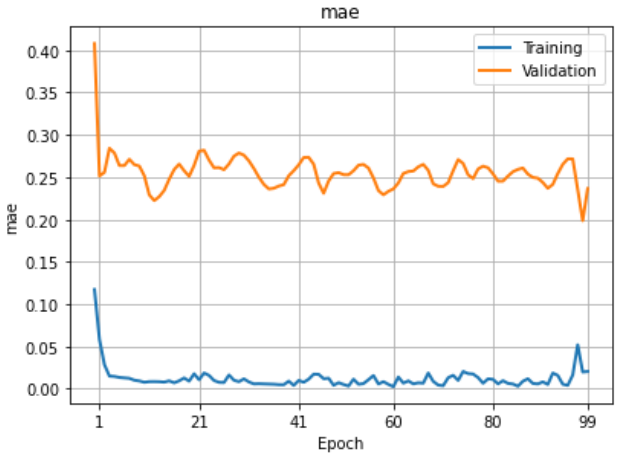
El uso de Deep Learning para el pronóstico de series temporales supera las desventajas tradicionales del aprendizaje automático con muchos enfoques diferentes. En este proyecto se presentan 5 arquitecturas de aprendizaje profundo diferentes para el pronóstico de nuestra serie temporale:

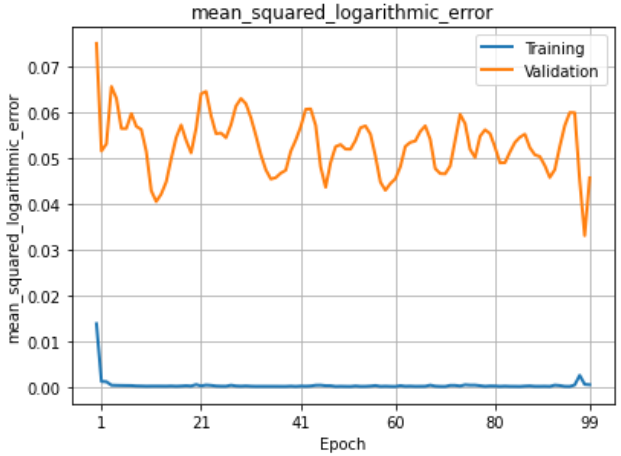
- *Redes neuronales recurrentes (**RNN**), que son la arquitectura más clásica y utilizada para problemas de predicción de series temporales;*
- *Long Short-Term Memory (**LSTM**), que son una evolución de las RNN desarrolladas para superar el problema del gradiente que desaparece;*
- *Redes neuronales convolucionales (**CNN**), aunque es popular en conjuntos de datos de imágenes, también se puede usar (y puede ser más práctico que los RNN) en datos de series de tiempo;*
- *Redes neuronales densas (**DNN**) Una red neuronal profunda (DNN) es una red neuronal artificial (ANN) con múltiples capas entre las capas de entrada y salida;*
- *Mezcla de los mejores modelos*

CNN: red neuronal convolucional. IAQ

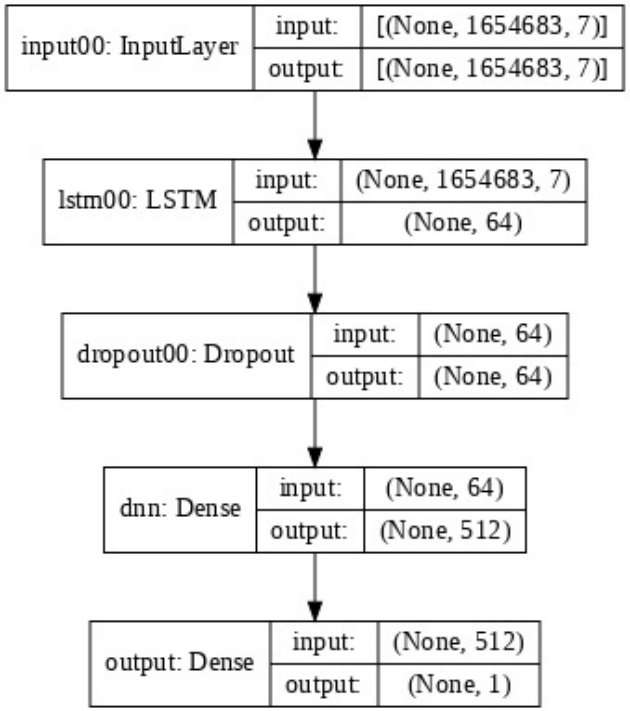
input00: InputLayer	input:	[(None, 1654683, 7)]
	output:	[(None, 1654683, 7)]
conv00: Conv1D	input:	(None, 1654683, 7)
	output:	(None, 1654677, 512)
dropout00: Dropout	input:	(None, 1654677, 512)
	output:	(None, 1654677, 512)
dnn: Dense	input:	(None, 1654677, 512)
	output:	(None, 1654677, 512)
dropout_1: Dropout	input:	(None, 1654677, 512)
	output:	(None, 1654677, 512)
dense_1: Dense	input:	(None, 1654677, 512)
	output:	(None, 1654677, 256)
output: Dense	input:	(None, 1654677, 256)
	output:	(None, 1654677, 1)

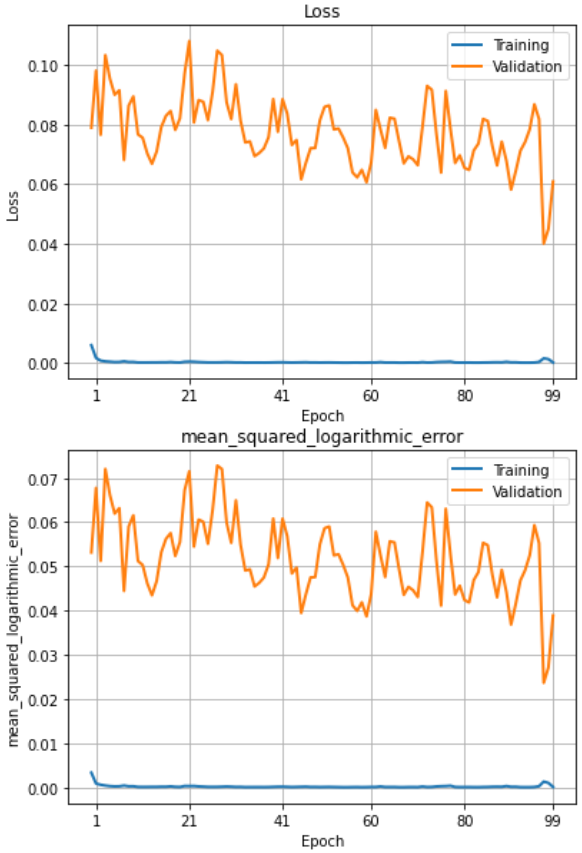


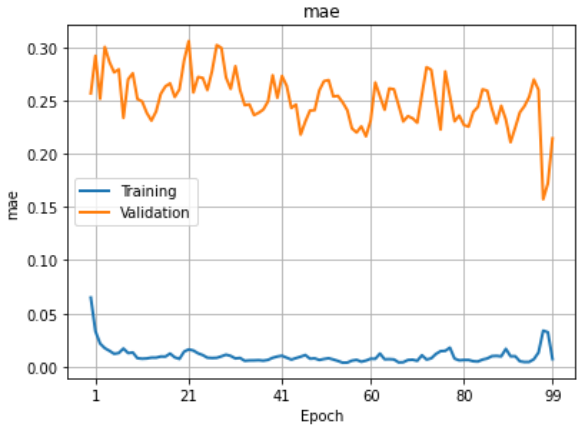




LSTM. IAQ


input00: InputLayer
input: [(None, 1654683, 7)]
output: [(None, 1654683, 7)]
lstm00: LSTM
input: (None, 1654683, 7)
output: (None, 64)
dropout00: Dropout
input: (None, 64)
output: (None, 64)
dnn: Dense
input: (None, 64)
output: (None, 512)
output: Dense
input: (None, 512)
output: (None, 1)


Loss
Training
Validation
Loss
Epoch
mean_squared_logarithmic_error
Training
Validation
mean_squared_logarithmic_error
Epoch


mae
Training
Validation
mae
Epoch

RNN: red neuronal recurrente. IAQ

Inputs: InputLayer	
input:	output:
[(None, None, 7)]	[(None, None, 7)]

→

RNN: SimpleRNN	
input:	output:
(None, None, 7)	(None, 512)

→

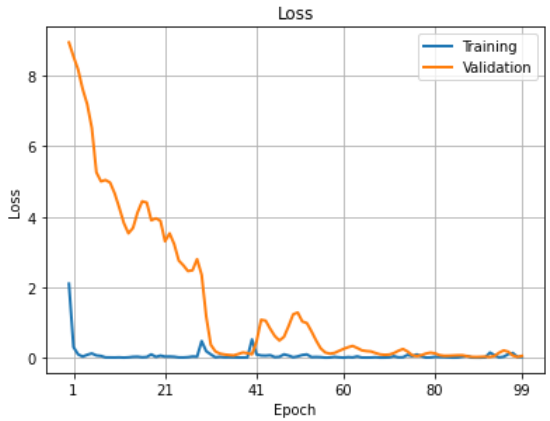
batch_normalization_1: BatchNormalization	
input:	output:
(None, 512)	(None, 512)

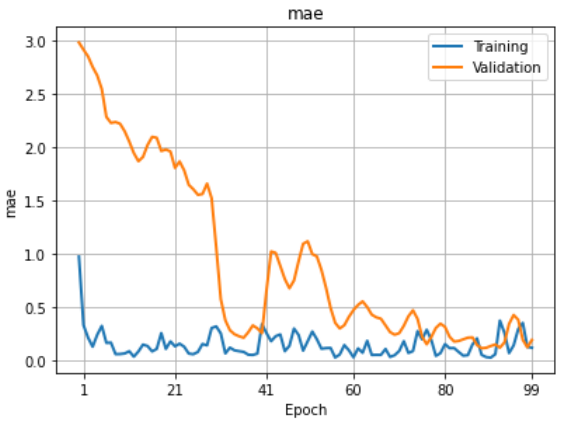
→

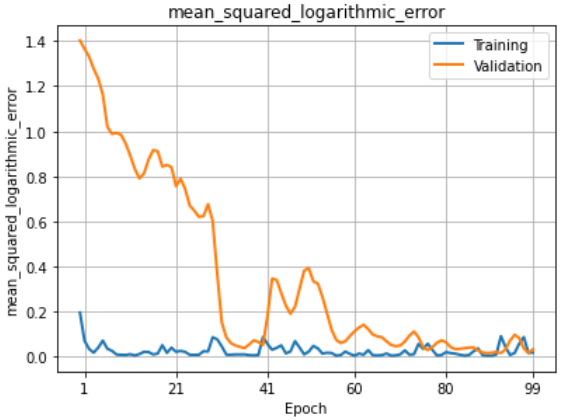
Dense: Dense	
input:	output:
(None, 512)	(None, 32)

→

Output: Dense	
input:	output:
(None, 32)	(None, 1)





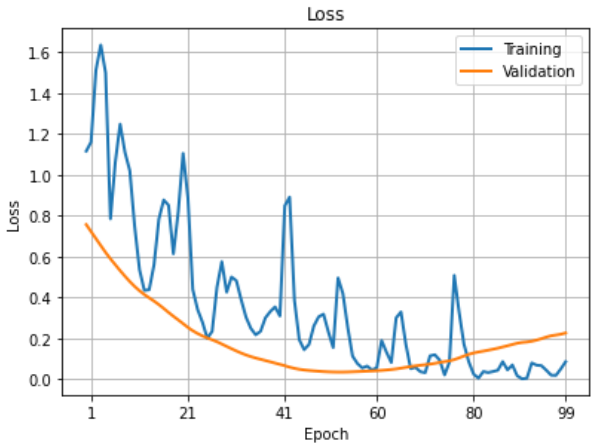


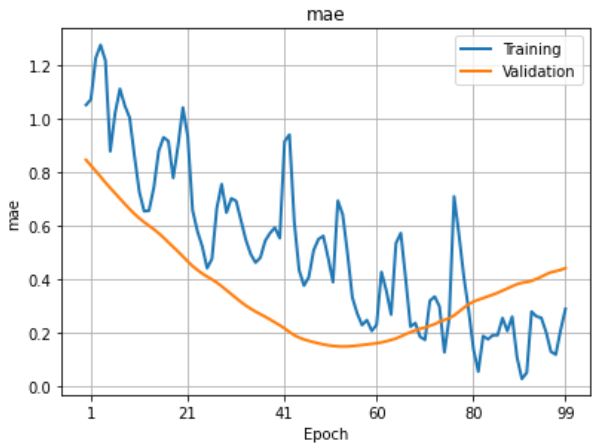
Modelo baseline. IAQ

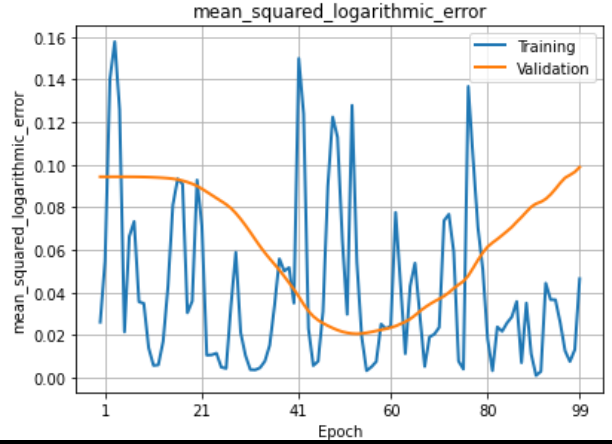
Inputs: InputLayer	
input:	output:
[(None, None, 7)]	[(None, None, 7)]

→

Output: Dense	
input:	output:
(None, None, 7)	(None, None, 1)







DNN: red neuronal datos secuenciales. IAQ

Inputs: InputLayer	
input:	output:
[(None, None, 7)]	[(None, None, 7)]

→

DNN: Dense	
input:	output:
(None, None, 7)	(None, None, 512)

→

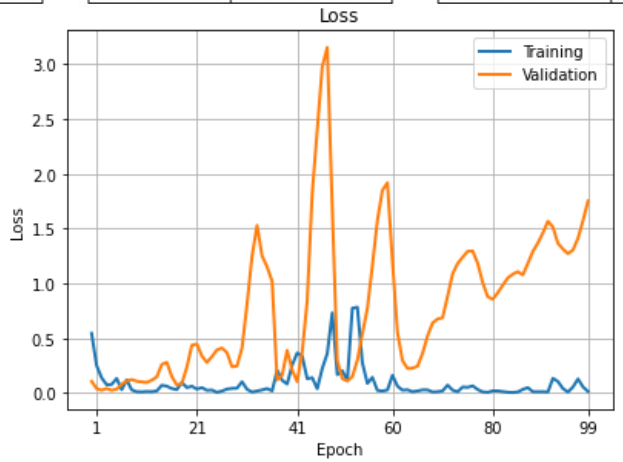
batch_normalization_1: BatchNormalization	
input:	output:
(None, None, 512)	(None, None, 512)

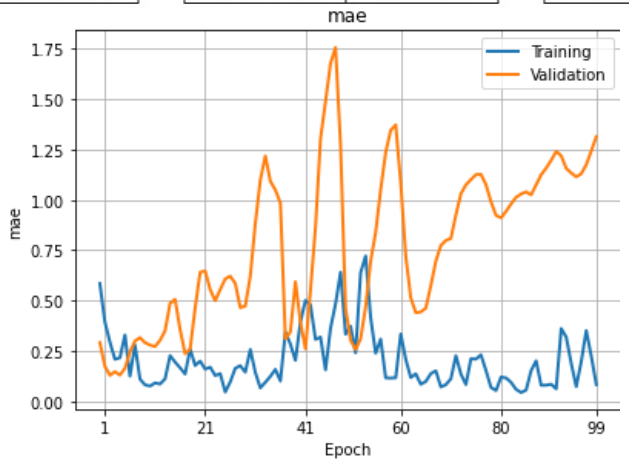
→

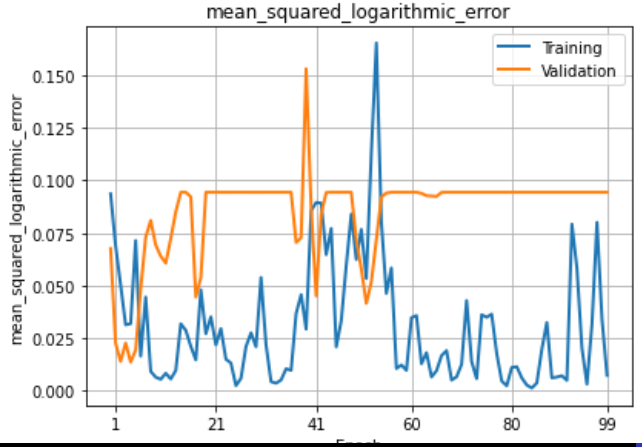
Dense: Dense	
input:	output:
(None, None, 512)	(None, None, 32)

→

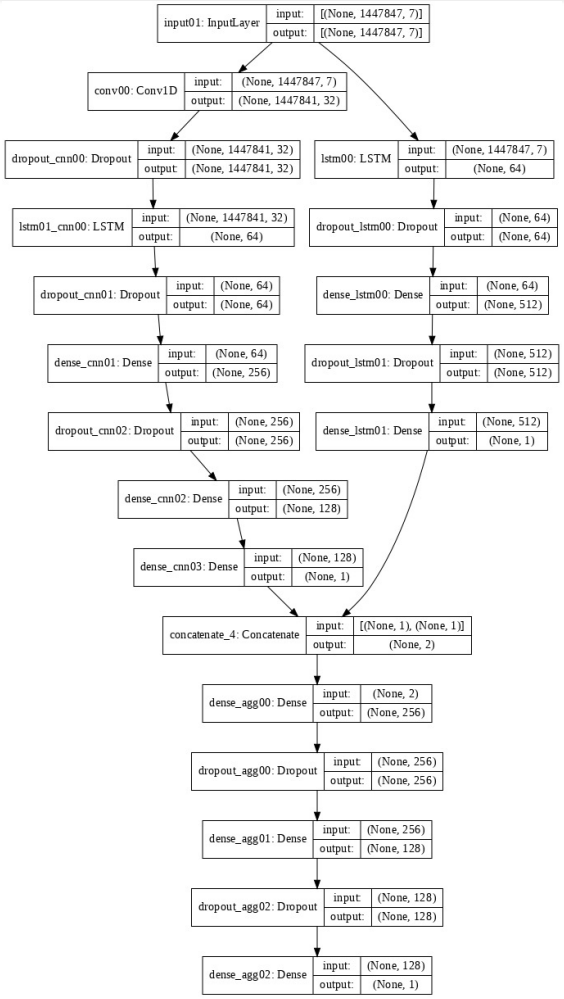
Output: Dense	
input:	output:
(None, None, 32)	(None, None, 1)

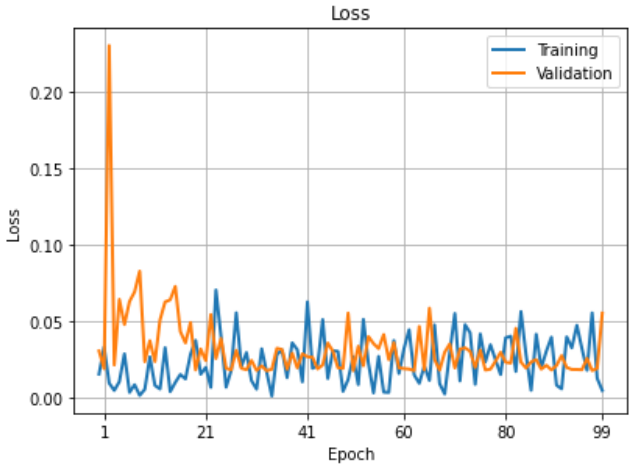


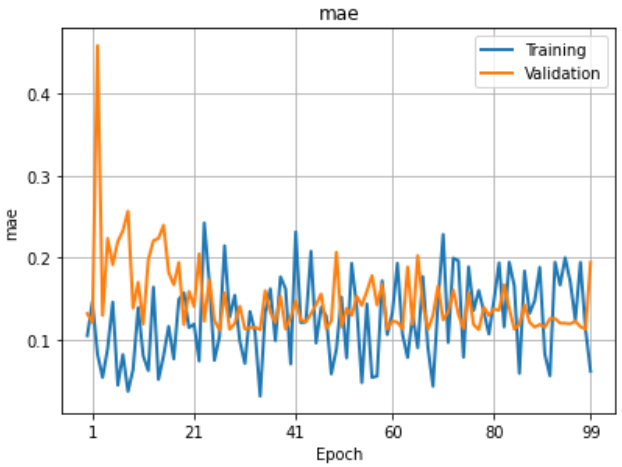


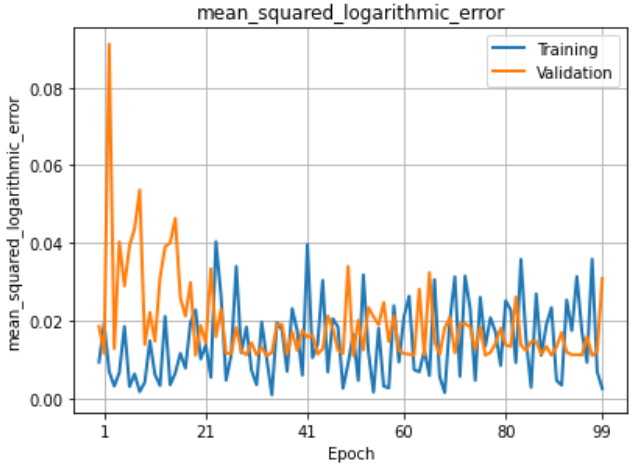


CNN + LSTM. IAQ


input01: InputLayer | input: [(None, 1447847, 7)] | output: [(None, 1447847, 7)]
conv00: Conv1D | input: (None, 1447847, 7) | output: (None, 1447841, 32)
dropout_cnn00: Dropout | input: (None, 1447841, 32) | output: (None, 1447841, 32)
lstm00: LSTM | input: (None, 1447847, 7) | output: (None, 64)
lstm01_cnn00: LSTM | input: (None, 1447841, 32) | output: (None, 64)
dropout_lstm00: Dropout | input: (None, 64) | output: (None, 64)
dropout_cnn01: Dropout | input: (None, 64) | output: (None, 64)
dense_lstm00: Dense | input: (None, 64) | output: (None, 512)
dense_cnn01: Dense | input: (None, 64) | output: (None, 256)
dropout_lstm01: Dropout | input: (None, 512) | output: (None, 512)
dropout_cnn02: Dropout | input: (None, 256) | output: (None, 256)
dense_lstm01: Dense | input: (None, 512) | output: (None, 1)
dense_cnn02: Dense | input: (None, 256) | output: (None, 128)
dense_cnn03: Dense | input: (None, 128) | output: (None, 1)
concatenate_4: Concatenate | input: [(None, 1), (None, 1)] | output: (None, 2)
dense_agg00: Dense | input: (None, 2) | output: (None, 256)
dropout_agg00: Dropout | input: (None, 256) | output: (None, 256)
dense_agg01: Dense | input: (None, 256) | output: (None, 128)
dropout_agg02: Dropout | input: (None, 128) | output: (None, 128)
dense_agg02: Dense | input: (None, 128) | output: (None, 1)


Loss
Training
Validation
Loss
Epoch
0.20
0.15
0.10
0.05
0.00
1
21
41
60
80
99


mae
Training
Validation
mae
Epoch
0.4
0.3
0.2
0.1
1
21
41
60
80
99


mean_squared_logarithmic_error
Training
Validation
mean_squared_logarithmic_error
Epoch
0.08
0.06
0.04
0.02
0.00
1
21
41
60
80
99

Resultado.

Resaltamos:

- **Usar los datos de la calidad del aire de la Ciudad de México mejoró nuestros modelos en todos los casos.**
- **Usar el paquete de keras:** tf.keras.preprocessing.timeseries_dataset_from_array **para interactuar con los modelos nos ayudó con el problema de *data leakage*.**
- **Los modelos más complejos no siempre fueron mejores.**
- **Mejora al utilizar técnicas de series de tiempo, estadística frecuentista y bayesiana para el análisis de los datos.**

Resultados. Comparación de modelos: IAQ

	Modelo	Tiempo	val_mae	mae
0	model_conv01	14m3.57s	62.76	62.73
1	model_best03b	9m1.76s	73.65	62.24
2	model_best03a	14m19.67s	76.24	53.65
5	model_lstm03	9m11.55s	87.76	21.71
7	model_lstm01	1m40.80s	125.40	5.02
8	model_conv03	6m33.58s	127.78	5.79
10	model_dnn01	1m40.90s	131.42	61.27
14	model_rnn01	7m56.31s	142.26	92.01
15	model_baseline01	0m57.19s	177.51	257.50
17	model_dnn03	4m13.19s	381.25	101.50
18	model_rnn03	7m32.10s	464.77	76.45

Conclusión

Las redes neuronales recurrentes son la técnica de aprendizaje profundo más popular para la predicción de series temporales, ya que permiten realizar predicciones fiables sobre series temporales en muchos problemas diferentes. El principal problema con los ***RNN*** *es que sufren el problema del gradiente de desaparición cuando se aplican a secuencias largas.*

LSTM *se creó para mitigar el problema del gradiente de desaparición de los RNN con el uso de puertas, que regulan el flujo de información a través de la cadena de secuencia. El uso de LSTM da resultados notables en aplicaciones como reconocimiento de voz, síntesis de voz, comprensión del lenguaje natural, etc.*

Conclusión

*El beneficio de usar **CNN-1D** para la clasificación de secuencias es que pueden aprender directamente de los datos de series de tiempo sin procesar y, a su vez, no requieren experiencia en el dominio para diseñar manualmente las características de entrada. El modelo aprendió una representación interna de los datos de la serie temporal y logró el mejor rendimiento comparable al de los modelos que se ajustan a una versión del conjunto de datos con características diseñadas.*

*La idea clave en el modelado del **DNN**: consideramos series de tiempo como modelo lineal: {X (i)… X (i + t)} ~ Y (i + t + 1). Usamos la series de tiempo de entrada de t pasos para predecir el siguiente paso, que es Y (i + t + 1).*

¡Gracias!

Preguntas