Redes neuronales

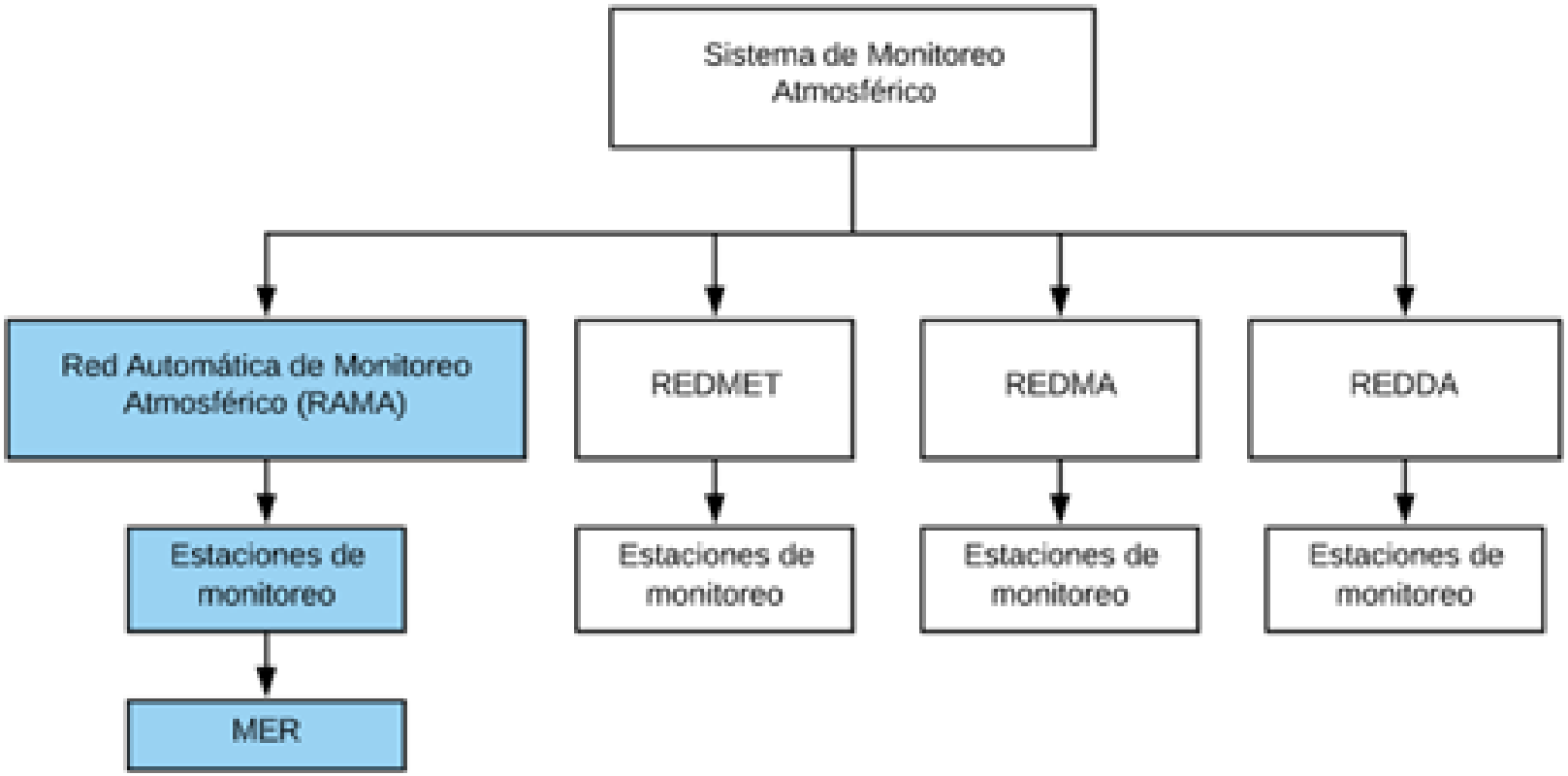
Análisis de partículas PM 2.5 mediante redes LSTM

Zúñiga González Daniel Iván

15 de junio, 2020

Introducción

Introducción



Objetivos

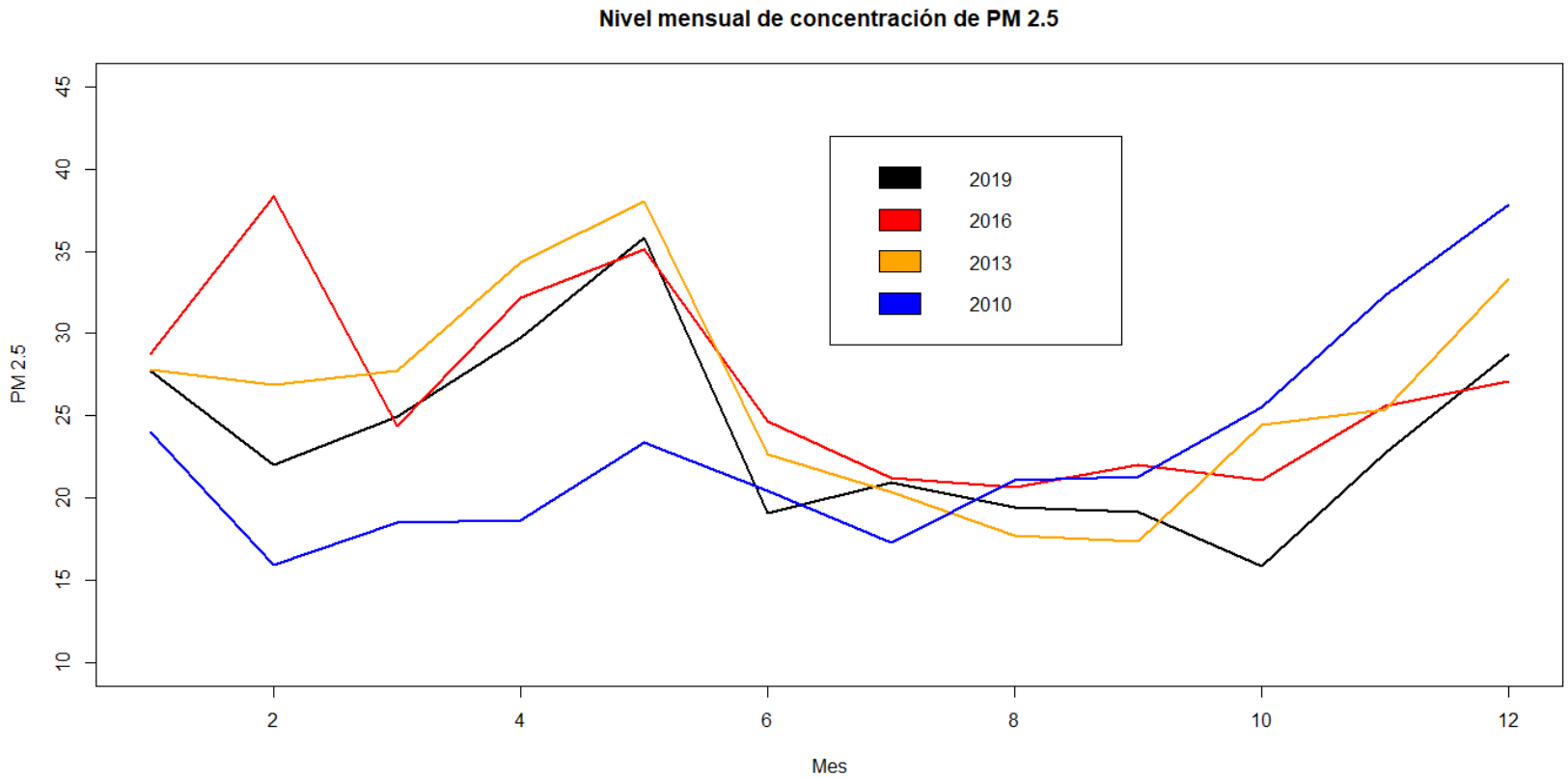
- Uso de redes neuronales artificiales (LSTM) como alternativa a los modelos estadísticos clásicos;
- Predicción año 2020

Datos (2010-2019)

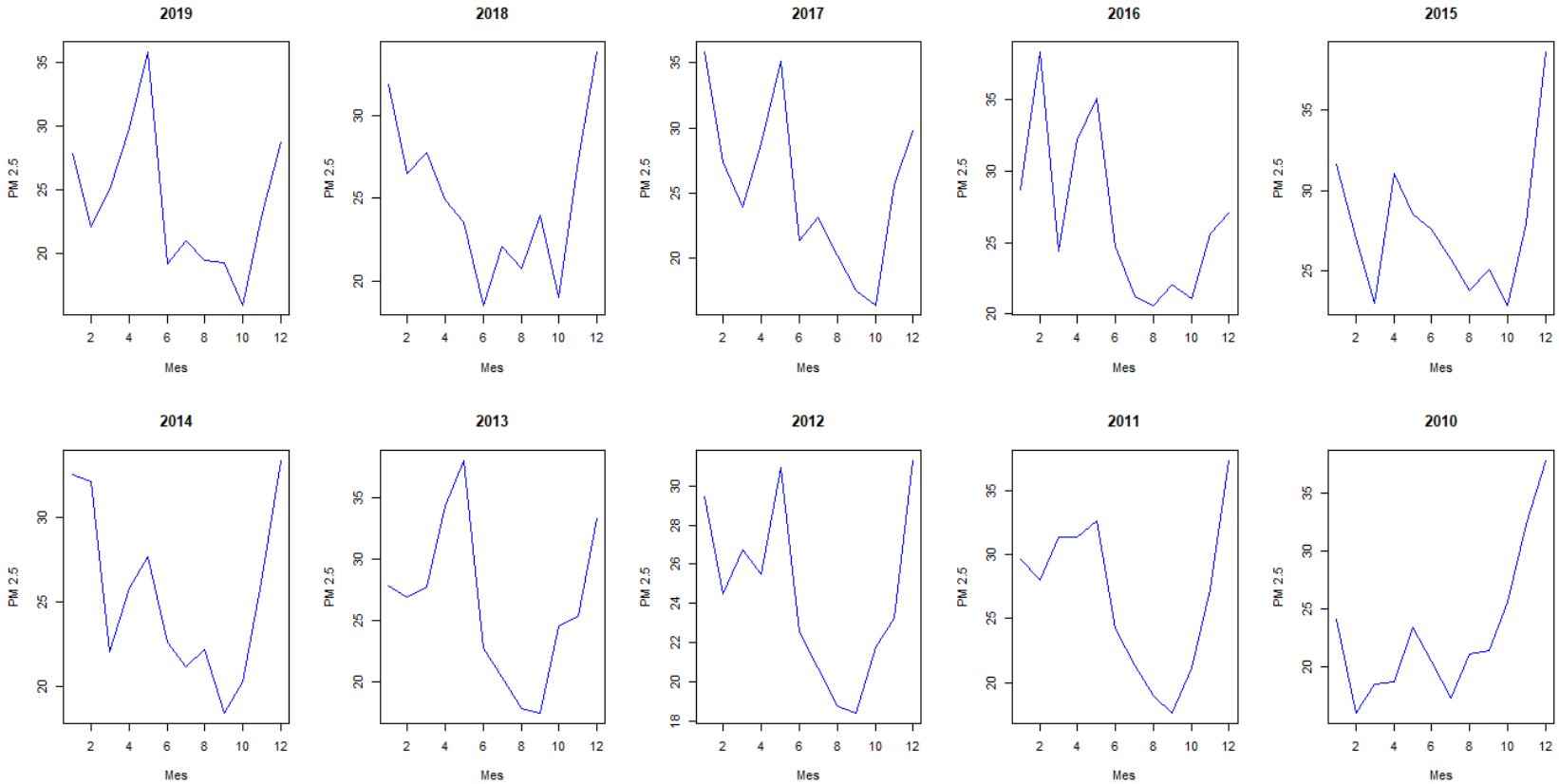
	A	B	C	D	E	F	G	H	I	J
1	FECHA	HORA	AJM	AJU	BJU	CAM	CCA	COY	GAM	HGM
2	01/01/2019	1	19	35	62	90	66	-99	-99	5
3	01/01/2019	2	17	24	88	104	84	-99	-99	6
4	01/01/2019	3	14	20	107	140	95	-99	-99	7
5	01/01/2019	4	6	15	101	162	97	-99	-99	9
6	01/01/2019	5	4	8	121	133	88	-99	-99	9
7	01/01/2019	6	7	7	93	106	77	-99	-99	10
8	01/01/2019	7	12	8	84	98	51	-99	-99	11
9	01/01/2019	8	15	7	101	82	39	-99	-99	9
10	01/01/2019	9	24	3	89	54	26	-99	-99	9
11	01/01/2019	10	24	-99	88	76	26	-99	-99	9
12	01/01/2019	11	49	-99	102	77	66	-99	-99	9

Fuente: Red automática de Monitoreo Atmosférico del Gobierno de la CDMX (http://www.aire.cdmx.gob.mx/default.php?opc=%27aKBh%27)

Análisis exploratorio



Análisis exploratorio



Código

```
df = pd.read_csv(r'C:/')
```

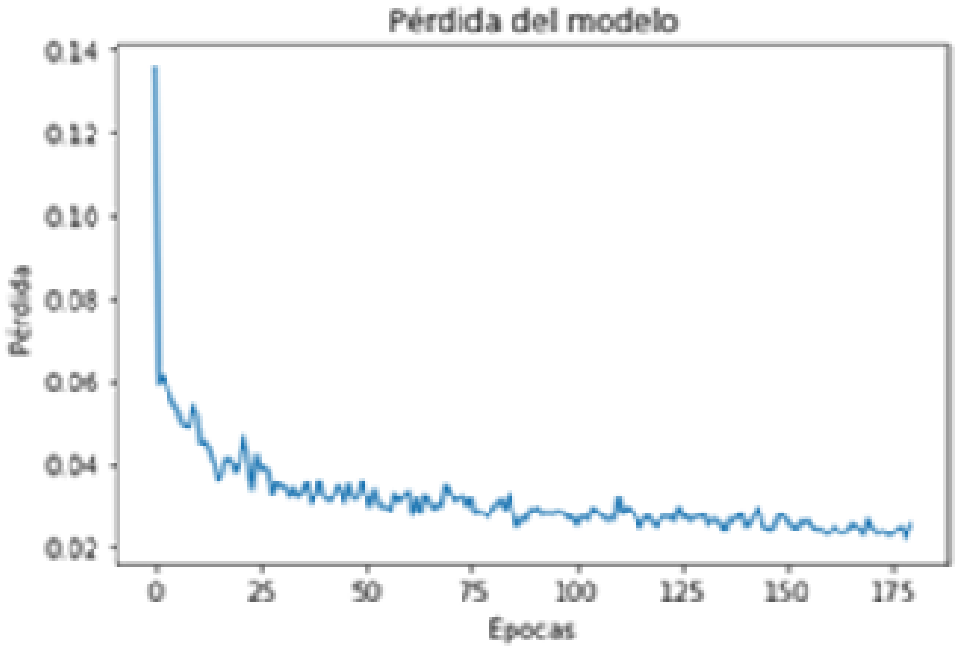
```
train, test = df[:-12], df[-12:]
```

```
scaler = MinMaxScaler()
scaler.fit(train)
train = scaler.transform(train)
test = scaler.transform(test)
```

```
n_input = 12
n_features = 1

generator = TimeseriesGenerator(train,train,length = n_input,
                                batch_size = 6)

model = Sequential()
model.add(LSTM(200, activation='relu', input_shape= (n_input,n_features)))
model.add(Dropout(0.15))
model.add(Dense(1))
model.compile(optimizer='Adam', loss='mse')

H=model.fit_generator(generator, epochs=180)
```

Código

```
Epoch 176/180
16/16 [==============================] - 1s 51ms/step - loss: 0.0232
Epoch 177/180
16/16 [==============================] - 1s 52ms/step - loss: 0.0242
Epoch 178/180
16/16 [==============================] - 1s 52ms/step - loss: 0.0249
Epoch 179/180
16/16 [==============================] - 1s 52ms/step - loss: 0.0220
Epoch 180/180
16/16 [==============================] - 1s 53ms/step - loss: 0.0254
```

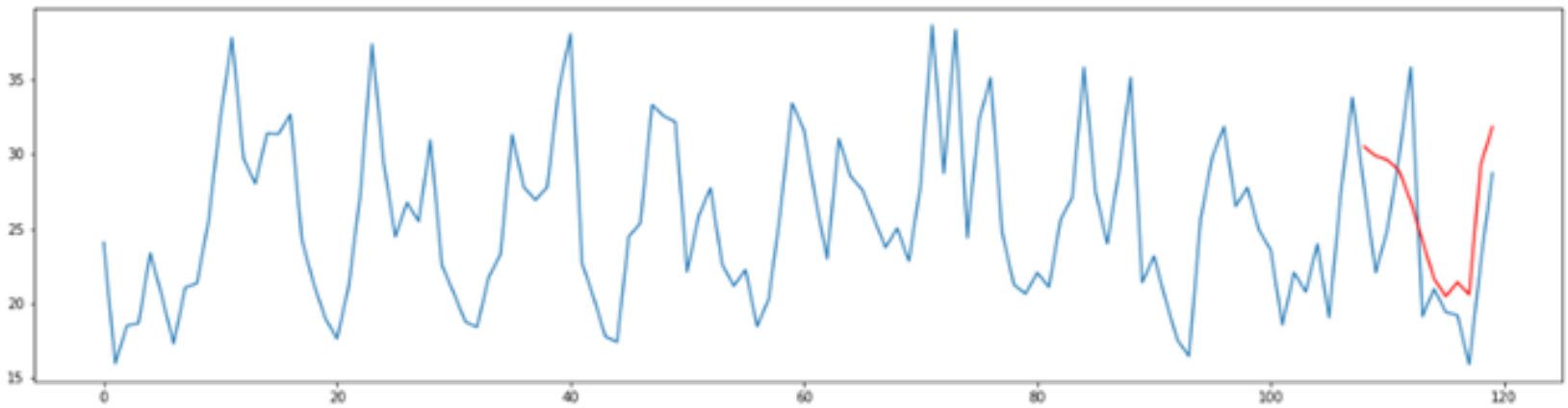


Código

```
pred_list = []

batch = train[-n_input:].reshape((1, n_input, n_features))

for i in range(n_input):
    pred_list.append(model.predict(batch)[0])
    batch = np.append(batch[:, 1:,:], [[pred_list[i]]], axis=1)
```

```
df_predict = pd.DataFrame(scaler.inverse_transform(pred_list),
                          index=df[-n_input:].index, columns=['Predicciones'])

df_test = pd.concat([df, df_predict], axis=1)
```

Código



Código

```
train = df

scaler.fit(train)
train = scaler.transform(train)

n_input = 12
n_features = 1

generator = TimeseriesGenerator(train,train,length = n_input,
                                batch_size = 6)

model.fit_generator(generator, epochs=180)

pred_list = []

batch = train[-n_input:].reshape((1, n_input, n_features))

for i in range(n_input):
    pred_list.append(model.predict(batch)[0])
    batch = np.append(batch[:, 1:,:], [[pred_list[i]]], axis=1)
```

Código

```
 df_predict = pd.DataFrame(scaler.inverse_transform(pred_list),
                           index=future_dates[-12:].index,
                           columns=['Predicciones'])
df_proj = pd.concat([df,df_predict],axis=1)
```

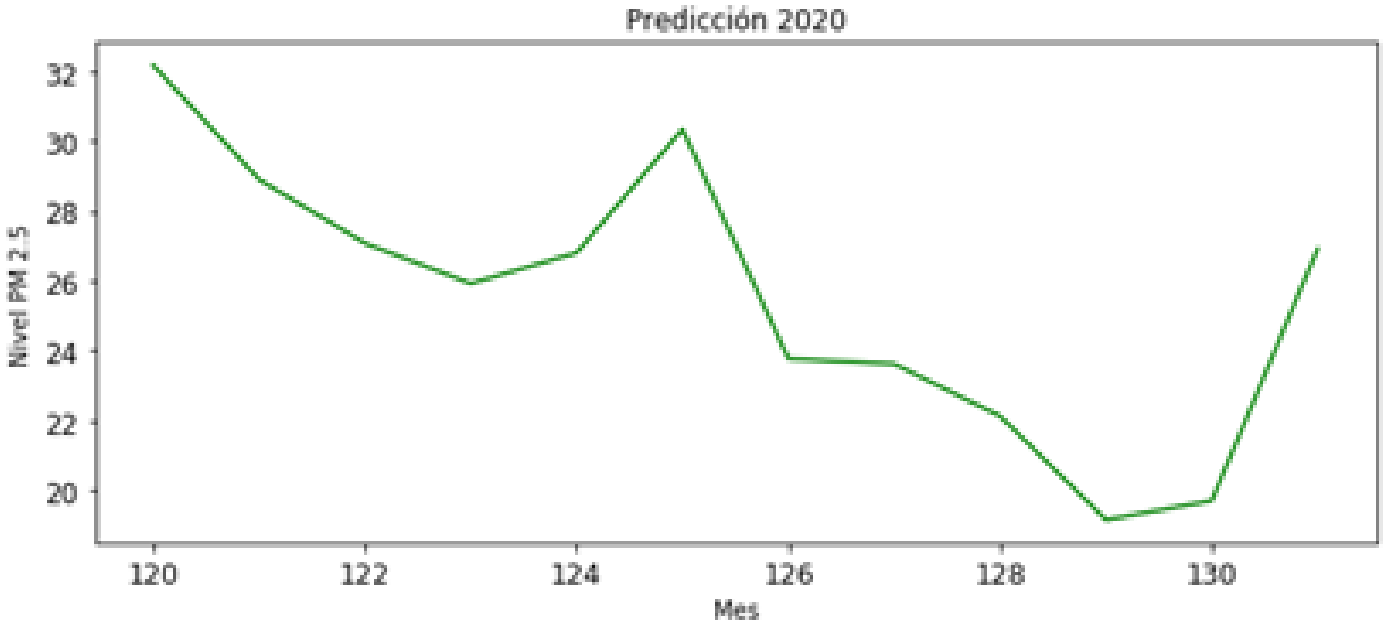
```
df_predict = pd.DataFrame(scaler.inverse_transform(pred_list),
                          index=future_dates[-n_input:].index,
                          columns=['Prediccion'])
df_proj = pd.concat([df,df_predict], axis=1)
```

Predicción mensual 2020

Mes	Enero	Febrero	Marzo	Abril	Mayo	Junio
PM 2.5	32.1	28.9	27.0	25.9	26.7	30.3

Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
23.7	23.6	22.1	19.1	19.6	26.9

Predicción mensual 2020



Predicción mensual 2020