DATA DE TELEMETRIA PARA IOT

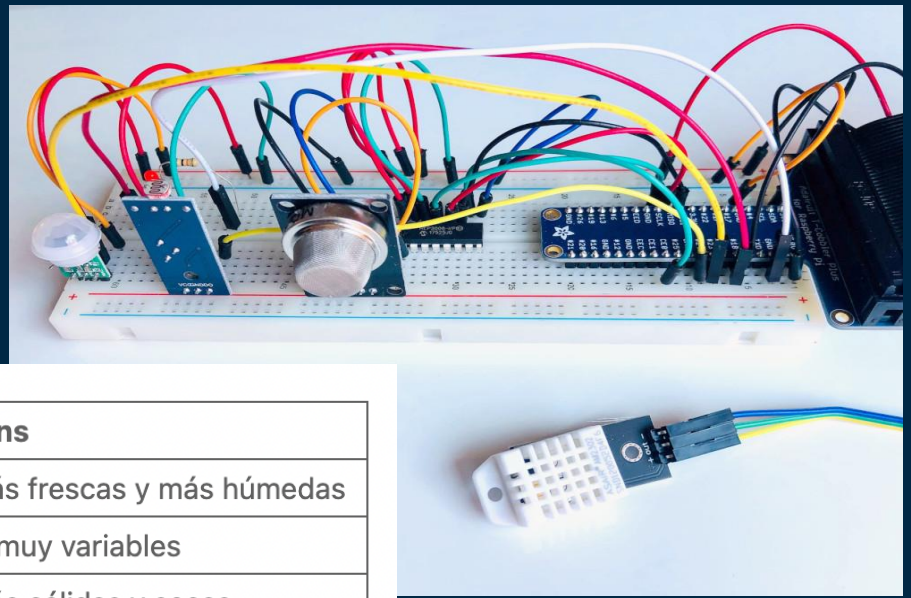
Jorge Esneider Henao Gonzalez

Aprendizaje Automático de Maquina I.

Profesor Oscar Samuel Fernandez

CONTEXTO DE LA DATA

Los datos se generaron a partir de una serie de conjuntos de sensores idénticos, hechos a medida. Cada sensor se conectó a una Raspberry Pi. Cada uno de los dispositivos IoT se colocaron a tomar datos en 3 diferentes ubicaciones físicas con diversas condiciones ambientales.

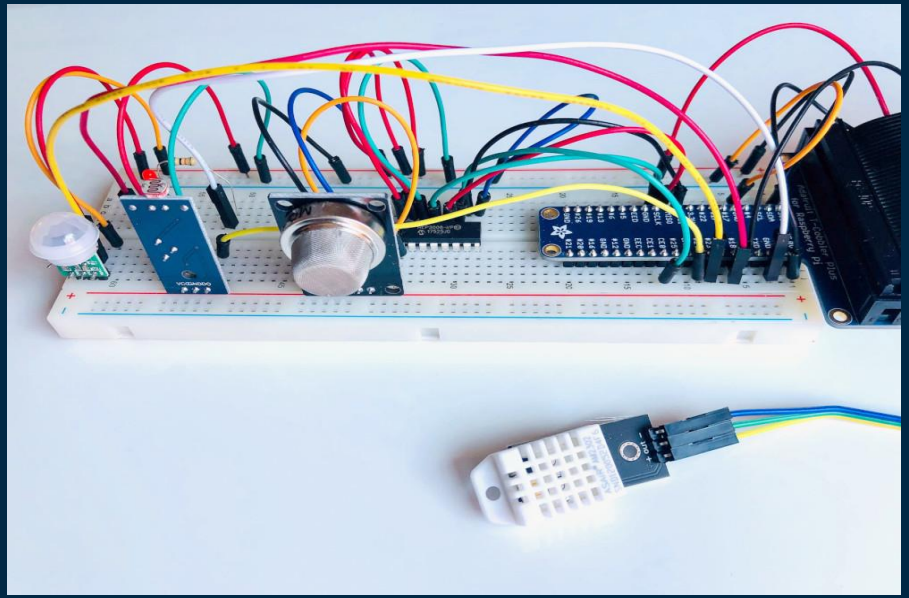


device	environmental conditions
00:0f:00:70:91:0a	condiciones estables, más frescas y más húmedas
1c:bf:ce:15:ec:4d	temperatura y humedad muy variables
b8:27:eb:bf:9d:51	condiciones estables, más cálidas y secas

CONTEXTO DE LA DATA

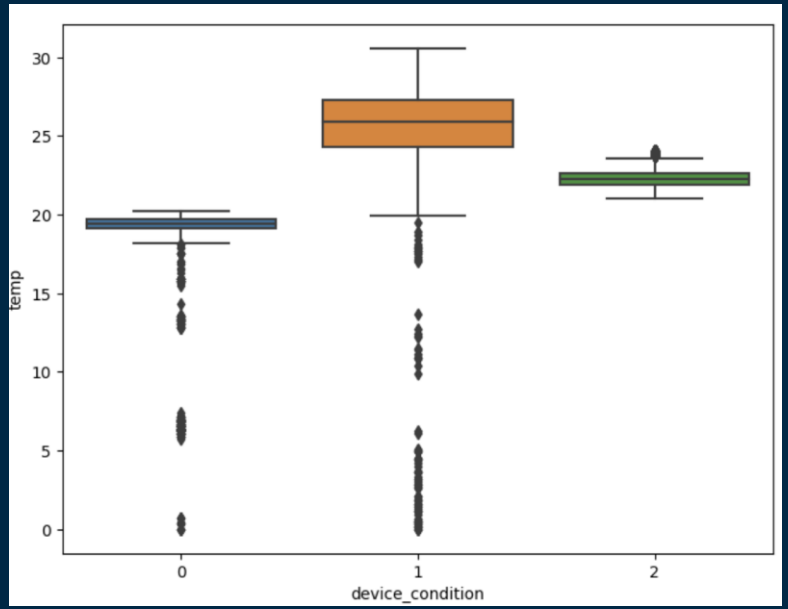
Cada dispositivo IoT recolectó un total de siete lecturas diferentes en un intervalo regular. Las lecturas provienen de los siguientes sensores que incluyen temperatura, humedad, monóxido de carbono (CO), gas licuado de petróleo (GLP), humo, luz y movimiento. Los datos abarcan el período comprendido entre el 12/07/2020 00:00:00 UTC y el 19/07/2020 23:59:59 UTC . Hay un total de 405.184 filas de datos.

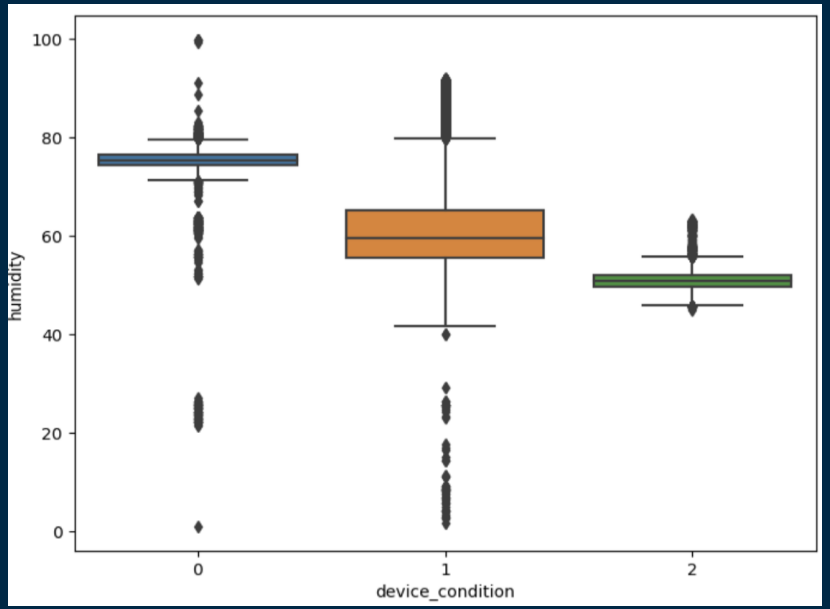
column	description	units
ts	timestamp of event	epoch
device	unique device name	string
co	carbon monoxide	ppm (%)
humidity	humidity	percentage
light	light detected?	boolean
lpg	liquid petroleum gas	ppm (%)
motion	motion detected?	boolean
smoke	smoke	ppm (%)
temp	temperature	Fahrenheit



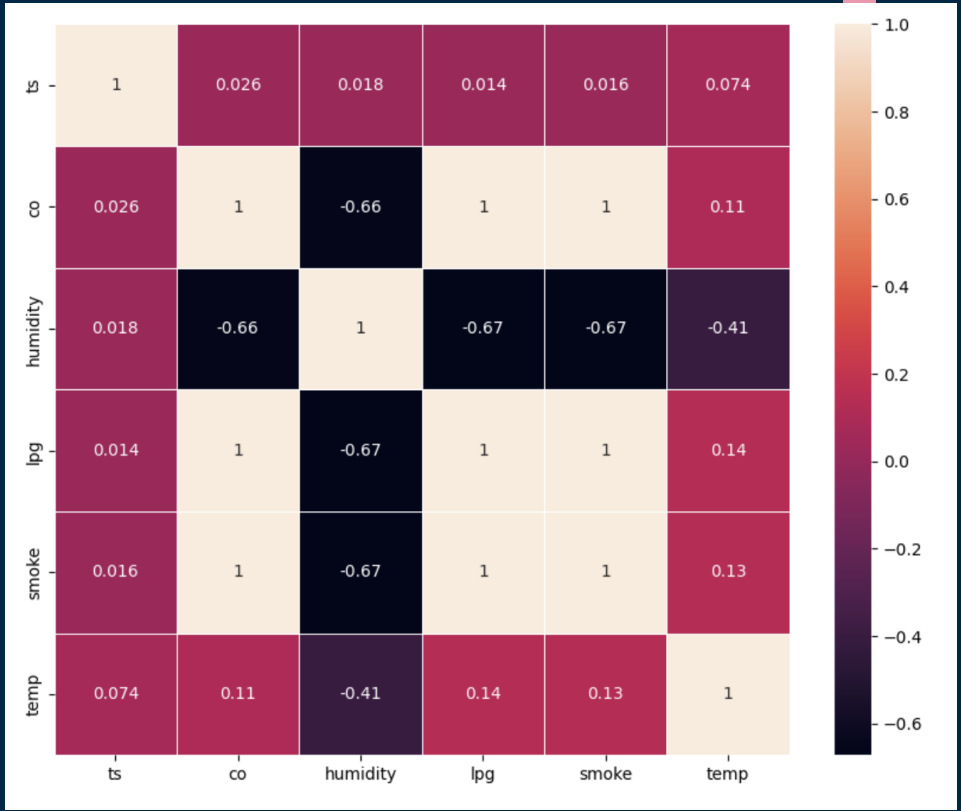
PREPROCESAMIENTO DE LOS DATOS

PREPROCESAMIENTO DE LOS DATOS





PREPROCESAMIENTO DE LOS DATOS



MATRIZ DE CORRELACION

PROCESAMIENTO DE LOS DATOS

	ts	device	co	humidity	light	lpg	motion	smoke	temp
0	1.594512e+09	b8:27:eb:bf:9d:51	0.004956	51.000000	False	0.007651	False	0.020411	22.700000
1	1.594512e+09	00:0f:00:70:91:0a	0.002840	76.000000	False	0.005114	False	0.013275	19.700001
2	1.594512e+09	b8:27:eb:bf:9d:51	0.004976	50.900000	False	0.007673	False	0.020475	22.600000
3	1.594512e+09	1c:bf:ce:15:ec:4d	0.004403	76.800003	True	0.007023	False	0.018628	27.000000
4	1.594512e+09	b8:27:eb:bf:9d:51	0.004967	50.900000	False	0.007664	False	0.020448	22.600000

Una variable x, variable predictora. La otra variable, y, variable de criterio o variable de respuesta

	ts	humidity	temp	device_condition	lights
0	1.594512e+09	51.000000	22.700000	2	0
1	1.594512e+09	76.000000	19.700001	0	0
2	1.594512e+09	50.900000	22.600000	2	0
3	1.594512e+09	76.800003	27.000000	1	1
4	1.594512e+09	50.900000	22.600000	2	0

MATRIZ DE CORRELACION

PROCESAMIENTO DE LOS DATOS- Entrenamiento modelo LinearRegresion

Entranamiento del modelo sin estandarizar para el modelo de LinearRegresion

In [87]:
```
X = data_limpia[X_cols].values
y = data_limpia[y_cols].values

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=42)


model_sin_standar = LinearRegression()
model_sin_standar.fit(X_train, y_train)

y_pred_sin_estandarizar = model_sin_standar.predict(X_test)
```

In [88]:
```
X_test
```

Out[88]:
```
array([[ 1.        , 28.79999924, 55.5       ],
       [ 0.        , 21.8       , 51.8       ],
       [ 1.        , 28.        , 65.5       ],
       ...,
       [ 1.        , 24.89999962, 55.40000153],
       [ 0.        , 19.10000038, 77.80000305],
       [ 1.        , 29.5       , 49.90000153]])
```

In [89]:
```
y_pred_sin_estandarizar
```

Out[89]:
```
array([[1.5102485 ],
       [1.78869408],
       [0.86001262],
       ...,
       [1.37449376],
       [0.07551161],
       [1.88355372]])
```

In [93]:
```
mse = metrics.mean_squared_error(y_test, y_pred_sin_estandarizar)
r2 = metrics.r2_score(y_test, y_pred_sin_estandarizar)
print("="*82)
print('r2 = ',r2)
print('mse = ',mse)
print("="*82)
```

```
==================================================================================
r2 =  0.8232370984673473
mse =  0.12422341986537934
==================================================================================
```

DATA LISTA PARA EL PROCESO DE APLICACION DE LOS DIFERENTES MODELOS

In [111…
```
data_limpia_input = dataProcessed.copy()
removedCol = ['lpg', 'smoke', 'co']

data_limpia = data_limpia_input.drop(removedCol, axis=1)
data_limpia.head()
```

Out[111…

	ts	humidity	temp	device_condition	lights
0	1.594512e+09	51.000000	22.700000	2	0
1	1.594512e+09	76.000000	19.700001	0	0
2	1.594512e+09	50.900000	22.600000	2	0
3	1.594512e+09	76.800003	27.000000	1	1
4	1.594512e+09	50.900000	22.600000	2	0

```
X_cols = list(set(data_limpia.columns)-set(['ts','device_condition']))
y_cols = ['device_condition']
```

PROCESAMIENTO DE LOS DATOS- Entrenamiento modelo Ridge

In [9]:
```
X_cols = list(set(data_limpia.columns)-set(['ts','device_condition','motion']))
y_cols = ['device_condition']

X = data_limpia[X_cols].values
y = data_limpia[y_cols].values

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=42)

sc_x = StandardScaler().fit(X)
sc_y = StandardScaler().fit(y)

X_train = sc_x.transform(X_train)
X_test = sc_x.transform(X_test)
y_train = sc_y.transform(y_train)
y_test = sc_y.transform(y_test)
```

In [10]:
```
modelo_ridge = Ridge(alpha=6).fit(X_train, y_train)
```

In [11]:
```
X_test
```

Out[11]:
```
array([[ 2.35181755, -0.44091891,  1.61269072],
       [-0.24236622, -0.76643759, -0.6200817 ],
       [ 2.05533966,  0.43886129,  1.61269072],
       ...,
       [ 0.90648658, -0.44971658,  1.61269072],
       [-1.24297993,  1.52099121, -0.6200817 ],
       [ 2.61123624, -0.93359569,  1.61269072]])
```

In [12]:
```
y_predict_ridge = modelo_ridge.predict(X_test)
y_predict_ridge
```

Out[12]:
```
array([[ 0.38572588],
       [ 0.71762297],
       [-0.38936783],
       ...,
       [ 0.22388142],
       [-1.32452708],
       [ 0.83071433]])
```

PROCESAMIENTO DE LOS DATOS - Metricas

```
================================================================================
Metricas para el modelo de LinearRegression

Coeficiente de Determinacion R2 =  0.8232507139737062
Error Cuadratico Medio (Mean Squared Error - MSE) =  0.17650359352984085
Raiz Cuadrada del Error Cuadratico Medio (RMSE) =  0.4201233075298737
================================================================================
Metricas para el modelo de Lasso

Coeficiente de Determinacion R2 =  0.8191629184098187
Error Cuadratico Medio (Mean Squared Error - MSE) =  0.18058570680374764
Raiz Cuadrada del Error Cuadratico Medio (RMSE) =  0.42495377019594455
================================================================================
Metricas para el modelo de Ridge

Coeficiente de Determinacion R2 =  0.8232507130473051
Error Cuadratico Medio (Mean Squared Error - MSE) =  0.1765035944549542
Raiz Cuadrada del Error Cuadratico Medio (RMSE) =  0.4201233086308759
```

MATRIZ DE CORRELACION

PROCESAMIENTO DE LOS DATOS – predicción

device	environmental conditions	device_condition
00:0f:00:70:91:0a	condiciones estables, más frescas y más húmedas	0
1c:bf:ce:15:ec:4d	temperatura y humedad muy variables	1
b8:27:eb:bf:9d:51	condiciones estables, más cálidas y secas	2

In [129…

```
lights = int(input("escriba 1 o 0 para luz"))
temp = float(input("escriba la temperatura"))
humidity = float(input("escriba la humedad"))
predicion_sin_standar = model_sin_standar.predict(np.array([[lights, temp, humidity]]))
s = float(predicion_sin_standar[0])
pre_redondeada =round(s)

if pre_redondeada == 0:
    print('condiciones estables, más frescas y más húmedas')
    print(pre_redondeada)
    print(s)
elif pre_redondeada == 1:
    print('temperatura y humedad muy variables ')
    print(pre_redondeada)
    print(s)
elif pre_redondeada == 2:
    print('condiciones estables, más cálidas y secas ')
    print(pre_redondeada)
    print(s)
elif pre_redondeada >= 4:
    print('no existe una condicion para estas variables')
```

```
condiciones estables, más frescas y más húmedas
0
0.20915333105493517
```

MATRIZ DE CORRELACION

PROCESAMIENTO DE LOS DATOS – predicción

Modelo de RidgeCV - Codigo de Prediciones dados valores estandarizados te da la respuesta sin estandarizar

datos probados en el codigo de prediccion datos estandarizados - estos dados estan dados por X_test temp', 'humidity', 'lights' condicion 2.35181755, -0.44091891, 1.61269072 0.38572601

In [31]:

```
lights = float(input("escriba 1 o 0 para luz"))
temp = float(input("escriba la temperatura"))
humidity = float(input("escriba la humedad"))
predicion_sin_standar = modelo_ridge.predict(np.array([[temp, humidity, lights]]))
a = sc_y.inverse_transform(predicion_sin_standar)
s = float(a[0])
pre_redondeada =round(s)

if pre_redondeada == 0:
    print('condiciones estables, más frescas y más húmedas')
    print(pre_redondeada)
    print(s)
elif pre_redondeada == 1:
    print('temperatura y humedad muy variables ')
    print(pre_redondeada)
    print(s)
elif pre_redondeada == 2:
    print('condiciones estables, más cálidas y secas ')
    print(pre_redondeada)
    print(s)
elif pre_redondeada >= 4:
    print('no existe una condicion para estas variables')
```

```
escriba 1 o 0 para luz1.61269072
escriba la temperatura2.35181755
escriba la humedad-0.44091891
condiciones estables, más cálidas y secas
2
1.5102548340954518
```

el anterior codigo lo ejecute introduciendo los valoresde luz temperatura humedad estandarizados y salida la estoy dando sin estandarizar. escriba 1 o 0 para luz = 1.61269072 escriba la temperatura = 2.35181755 escriba la humedad = -0.44091891 array([[1.51025483]])

CONCLUSIONES

Teniendo en cuenta lo anteriormente expuesto, se logra obtener un modelo de predicción tanto para Ridge y linear regresión, en el que ambos obtuvieron un comportamiento muy similar.

El coeficiente de determinación tanto para Ridge como para linear regresión son muy idénticos

Tanto el Error Cuadrático Medio (Mean Squared Error - MSE) como la Raíz Cuadrada del Error Cuadrático Medio de Ridge como de linear regresión fueron de igualmente cercano.

Los coeficientes estimados por Ridge pueden verse alterados si las variables no se estandarizan antes de llevar a cabo el ajuste.

La ventaja con la que cuenta el Ridge respecto al método de mínimos cuadrados reside en el equilibrio bias-varianza: conforme λ aumenta, la flexibilidad del ajuste por Ridge Regression disminuye, lo cual disminuye la varianza, pero aumenta el bias.

Un pequeño cambio en el set de datos de entrenamiento puede hacer variar los coeficientes de manera sustancial en Linear Regresión, mientras que Ridge puede reducir considerablemente la varianza a expensas de un pequeño aumento en el bias (conforme λ aumenta).

Bibliografía

https://www.kaggle.com/datasets/garystafford/environmental-sensor-data-132k

https://www.kaggle.com/code/helloedi/ml-telemetrysensordata/notebook

MATRIZ DE CORRELACION