

# **Instituto Tecnológico Superior de Huatusco**



## **Integrantes:**

**América Guadalupe Salazar Huerta**

**Miguel Arath Tamburrino Méndez**

**David Quintero Caballero**

**Juan Angel Vargas Martínez**

**Ingenieria en sistemas computacionales**

**Inteligencia Artificial**

**Docente Luis Humberto Sánchez Medel**

**Concepto de SmartWatch usando un modelo de Machine Learning**

**12/06/2023**

# App de Adquisición de Datos

El primer paso para realizar este modelo es recabar la informacion de algún tipo de sensor de inercia, en especifico de algun sensor que contenga un acelerómetro y giroscopio.

Como sabemos, todos nuestros teléfonos inteligentes contienen sensores para proporcionarnos utilidades y herramientas, en este caso usaremos los sensores acelerómetro y giroscopio mediante una aplicación desarrollada con el lenguaje de programacion Java en el IDE Android Studio.

Esta aplicación lee los valores que entregan estos sensores en tiempo real y los guarda en una archivo CSV.

Aquí dejo el link al Branch del proyecto en GitHub:

<https://github.com/rootAngeeeel/AdquisicionDatos.git>

Al final el archivo generado debe quedar como el siguiente:

	A	B	C	D	E	F	G	H	
1	Hora	Ax	Ay	Az	Gx	Gy	Gz	L	
2	21:41.8	-0.12705001	0.90705	10.26795	-0.259875	0.2580875	0.21505	BR	
3	21:41.8	-0.12705001	0.90705	10.26795	-0.12045	-0.091575	0.2332	BR	
4	21:41.8	0.10605001	0.96000004	10.222051	-0.12045	-0.091575	0.2332	BR	
5	21:41.8	0.40605003	1.05795	10.258051	-0.12045	-0.091575	0.2332	BR	
6	21:41.8	0.40605003	1.05795	10.258051	-0.0204875	-0.253825	0.2437875	BR	
7	21:41.9	0.57795	1.1580001	10.231951	-0.0204875	-0.253825	0.2437875	BR	
8	21:41.9	0.60405004	1.23	10.15005	-0.0204875	-0.253825	0.2437875	BR	
9	21:41.9	0.60405004	1.23	10.15005	0.0253	-0.0790625	0.2129875	BR	
10	21:41.9	0.51105005	1.2709501	9.922951	0.0253	-0.0790625	0.2129875	BR	
11	21:41.9	0.42705002	1.29495	9.733951	0.0253	-0.0790625	0.2129875	BR	
12	21:41.9	0.42705002	1.29495	9.733951	0.0603625	0.082225	0.109725	BR	
13	21:41.9	0.372	1.29495	9.622951	0.0603625	0.082225	0.109725	BR	
14	21:41.9	0.37905002	1.28505	9.540001	0.0603625	0.082225	0.109725	BR	
15	21:41.9	0.37905002	1.28505	9.540001	0.0551375	0.0622875	0.0394625	BR	
16	21:41.9	0.432	1.26405	9.505951	0.0551375	0.0622875	0.0394625	BR	
17	21:41.9	0.57600003	1.2400501	9.66195	0.0551375	0.0622875	0.0394625	BR	
18	21:41.9	0.57600003	1.2400501	9.66195	-0.0045375	0.027775	0.0182875	BR	
19	21:41.9	0.62100005	1.2379501	9.778951	-0.0045375	0.027775	0.0182875	BR	
20	21:41.9	0.55200005	1.21095	9.741	-0.0045375	0.027775	0.0182875	BR	
21	21:41.9	0.55200005	1.21095	9.741	-0.064075	0.0539	-0.018975	BR	
22	21:42.0	0.46995002	1.1659501	9.754951	-0.064075	0.0539	-0.018975	BR	
23	21:42.0	0.38895002	1.1149501	9.813001	-0.064075	0.0539	-0.018975	BR	
24	21:42.0	0.38895002	1.1149501	9.813001	-0.062975	0.0518375	-0.07865	BR	
25	21:42.0	0.36	1.07505	9.784051	-0.062975	0.0518375	-0.07865	BR	
26	21:42.0	0.42495003	1.0170001	9.753	-0.062975	0.0518375	-0.07865	BR	
27	21:42.0	0.42495003	1.0170001	9.753	-0.0385	-0.0518375	-0.11165	BR	
28	21:42.0	0.45105	0.97905004	9.69495	-0.0385	-0.0518375	-0.11165	BR	
29	21:42.0	0.42705002	0.94305	9.622951	-0.0385	-0.0518375	-0.11165	BR	
30	21:42.0	0.42705002	0.94305	9.622951	-0.015125	-0.17215	-0.1285625	BR	
31	21:42.0	0.40305	0.91395	9.559051	-0.015125	-0.17215	-0.1285625	BR	
32	21:42.0	0.31500003	0.90495	9.51795	-0.015125	-0.17215	-0.1285625	BR	
33	21:42.0	0.31500003	0.90495	9.51795	-0.0204875	-0.2360875	-0.1584	BR	
34	21:42.0	0.23295002	0.94500005	9.52995	-0.0204875	-0.2360875	-0.1584	BR	
35	21:42.0	0.13005	0.95295006	9.678	-0.0204875	-0.2360875	-0.1584	BR	
36	21:42.0	0.13005	0.95295006	9.678	-0.08855	-0.2140875	-0.1860375	BR	
37	21:42.1	0.039	0.94995004	9.84105	-0.08855	-0.2140875	-0.1860375	BR	
38	21:42.1	-0.07995001	0.94305	10.01895	-0.08855	-0.2140875	-0.1860375	BR	
39	21:42.1	-0.07995001	0.94305	10.01895	-0.1854875	-0.12815	-0.199925	BR	

	A	B	C	D	E	F	G	H	
3483	49:47.0	0.44895002	1.8750001	13.144051	0.4035625	0.8063	0.2061125	WLK	
3484	49:47.0	-0.55395	2.5429502	13.776001	0.4035625	0.8063	0.2061125	WLK	
3485	49:47.0	-0.55395	2.5429502	13.776001	0.2695	0.20019999	0.1794375	WLK	
3486	49:47.0	-0.72405005	2.80095	13.312051	0.2695	0.20019999	0.1794375	WLK	
3487	49:47.0	-0.72405005	2.80095	13.312051	0.12045	-0.26345	0.2039125	WLK	
3488	49:47.0	-0.62595004	2.8210502	12.940051	0.12045	-0.26345	0.2039125	WLK	
3489	49:47.0	-0.62595004	2.8210502	12.940051	-0.112475	-0.7115625	0.0251625	WLK	
3490	49:47.0	-0.726	3.1630502	12.294001	-0.112475	-0.7115625	0.0251625	WLK	
3491	49:47.0	-0.726	3.1630502	12.294001	-0.14025	-1.1983125	-0.3345375	WLK	
3492	49:47.1	-1.45395	3.903	11.092051	-0.14025	-1.1983125	-0.3345375	WLK	
3493	49:47.1	-1.45395	3.903	11.092051	0.0301125	-1.41185	-0.674025	WLK	
3494	49:47.1	-2.11095	4.87305	9.79605	0.0301125	-1.41185	-0.674025	WLK	
3495	49:47.1	-2.11095	4.87305	9.79605	0.09075	-1.1187	-0.71555	WLK	
3496	49:47.1	-2.3719501	5.2110004	9.336	0.09075	-1.1187	-0.71555	WLK	
3497	49:47.1	-2.3719501	5.2110004	9.336	-0.130625	-0.529375	-0.443025	WLK	
3498	49:47.1	-2.26095	4.3510504	9.537001	-0.130625	-0.529375	-0.443025	WLK	
3499	49:47.1	-2.26095	4.3510504	9.537001	-0.33385	-0.2341625	-0.0875875	WLK	
3500	49:47.1	-1.95	3.4669502	9.37005	-0.33385	-0.2341625	-0.0875875	WLK	
3501	49:47.1	-1.95	3.4669502	9.37005	-0.2222	-0.5209875	0.0347875	WLK	
3502	49:47.2	-2.01	3.4789503	8.91405	-0.2222	-0.5209875	0.0347875	WLK	
3503	49:47.2	-2.01	3.4789503	8.91405	0.1024375	-0.9596125	-0.054725	WLK	
3504	49:47.2	-2.05995	3.5130002	8.365951	0.1024375	-0.9596125	-0.054725	WLK	
3505	49:47.2	-2.05995	3.5130002	8.365951	0.455675	-1.082125	-0.0556875	WLK	
3506	49:47.2	-1.5169501	3.084	7.5370502	0.455675	-1.082125	-0.0556875	WLK	
3507	49:47.2	-1.5169501	3.084	7.5370502	0.815375	-1.0967	0.1326875	WLK	
3508	49:47.2	-0.62295	2.6869502	6.5599504	0.815375	-1.0967	0.1326875	WLK	
3509	49:47.2	-0.62295	2.6869502	6.5599504	1.111275	-1.1364375	0.26565	WLK	
3510	49:47.2	0.04905	2.4130502	6.0019503	1.111275	-1.1364375	0.26565	WLK	
3511	49:47.2	0.04905	2.4130502	6.0019503	1.2368125	-1.1218625	0.23485	WLK	
3512	49:47.3	0.02205	2.1790502	6.18195	1.2368125	-1.1218625	0.23485	WLK	
3513	49:47.3	0.02205	2.1790502	6.18195	1.26775	-1.012	0.1134375	WLK	
3514	49:47.3	-0.19500001	2.2339501	6.4879503	1.26775	-1.012	0.1134375	WLK	
3515	49:47.3	-0.19500001	2.2339501	6.4879503	1.36675	-0.7921375	0.07095	WLK	
3516	49:47.3	0.12495001	2.60295	6.38805	1.36675	-0.7921375	0.07095	WLK	
3517	49:47.3	0.12495001	2.60295	6.38805	1.4806	-0.5011875	0.1742125	WLK	
3518	49:47.3	0.70005006	3.2899501	6.49095	1.4806	-0.5011875	0.1742125	WLK	
3519	49:47.3	0.70005006	3.2899501	6.49095	1.3794	0.0044	0.322025	WLK	
3520	49:47.3	0.72705	3.489	7.3120503	1.3794	0.0044	0.322025	WLK	
3521	49:47.3	0.72705	3.489	7.3120503	0.9857375	0.913	0.5465625	WLK	

Donde Ax, Ay y Az son datos del acelerómetro en sus 3 magnitudes al igual que Gx, Gy y Gz del giroscopio. La ultima columna llamada L es una columna de clasificacion, cada actividad hecha se debe etiquetar para que el modelo de ML sepa que cosa se esta haciendo. Más adelante se cambiara este valor (por ejemplo si la actividad es caminar la etiqueta será “WLK”).

# Preprocesamiento de datos

Una vez teniendo los datos guardados es momento de preprocesarlos, este paso es muy importante, debido a que usamos sensores físicos siempre habrá cierto ruido eléctrico en la toma de los datos lo que provoca cierta imprecisión, lo que debemos hacer es suavizar esta señal utilizando una tecnica llamada Windowing (Ventaneo).

## Codigo en Matlab

```
%Cargamos los datos del archivo de adquisicion de datos usando el telefono movil
datos = readtable('C:\Users\solda\OneDrive\Escritorio\SmartWatch\nano33_Preprocesado.csv');

%Descomponemos el archivo en vectores
Vec_Ax = table2array(datos(:, 1));
Vec_Ay = table2array(datos(:, 2));
Vec_Az = table2array(datos(:, 3));
Vec_Gx = table2array(datos(:, 4));
Vec_Gy = table2array(datos(:, 5));
Vec_Gz = table2array(datos(:, 6));
Vec_L = table2array(datos(:, 7));

%Seteamos la frecuencia de actualizacion y el tamaño de la ventana
fhz = 996;
V = 0.1;

%Calculamos el tamaño del ventaneo
ventana = fix(fhz * V - 1);

%Seteamos el tamaño de los vectores
V_preprocesado = size(Vec_Ax,1);

%Creamos 7 vectores para los 3 ejes de cada sensor más la etiqueta
VPP_Ax = zeros(V_preprocesado-ventana,1);
VPP_Ay = zeros(V_preprocesado-ventana,1);
VPP_Az = zeros(V_preprocesado-ventana,1);
VPP_Gx = zeros(V_preprocesado-ventana,1);
VPP_Gy = zeros(V_preprocesado-ventana,1);
VPP_Gz = zeros(V_preprocesado-ventana,1);
VPP_L = zeros(V_preprocesado-ventana,1);

%Empezamos la etapa de pre procesamiento en paralelo
parfor n = 1:(size(datos,1)-ventana)
    VPP_Ax(n) = mean(Vec_Ax(n : n + ventana,1));
    VPP_Ay(n) = mean(Vec_Ay(n : n + ventana,1));
    VPP_Az(n) = mean(Vec_Az(n : n + ventana,1));
    VPP_Gx(n) = mean(Vec_Gx(n : n + ventana,1));
    VPP_Gy(n) = mean(Vec_Gy(n : n + ventana,1));
    VPP_Gz(n) = mean(Vec_Gz(n : n + ventana,1));
end

%Preprocesamos la etiqueta
VPP_L = Vec_L(1:size(datos,1)-ventana,1);

%Cargamos los valores en una nueva variable
M_PP = [VPP_Ax, VPP_Ay, VPP_Az, VPP_Gx, VPP_Gy, VPP_Gz, VPP_L];
surf(M_PP);

%Limpiamos nuestro Workspace
clear("n", "fhz", "V", "V_preprocesado", "Vec_Ay", "Vec_Gx", "Vec_Gz", "Vec_Ax", "Vec_Az", "Vec_Gy", "Vec_L", "ventana", "VPP_Ax", "VPP_Ay", "VPP_Az", "VPP_Gx", "VPP_Gy", "VPP_Gz", "VPP_L");
```

El codigo anterior descompone el archivo CSV generado en 7 vectores diferentes, agregamos la frecuencia y la ventana, calculamos el tamaño de la ventana y creamos nuevos vectores que contendrán los datos preprocesados, al final con un ciclo for paralelo preprocesamos cada vector por separado. Al final cargamos los nuevos valores a una variable con los vectores ya preprocesados más la etiqueta de actividad. Cabe recalcar que la etiqueta debe ser remplazada

por un valor numerico ya que al momento de cargar los nuevos valores marcara un error de concatenación.

Esto se hace usando una relación parecida a esta:

Actividad	Etiqueta	Valor numerico
Caminar	WLK	0
Correr	RN	1
Brincar	BR	2

■	A	B	C	D	E	F	G	H
2633	22:01.5	0.39105	1.4320501	9.280951	0.2316875	0.2014375	-0.0818125	0
2634	22:01.5	0.79800004	1.5460501	9.23505	0.2316875	0.2014375	-0.0818125	0
2635	22:01.5	0.79800004	1.5460501	9.23505	0.2519	0.113575	-0.0669625	0
2636	22:01.6	0.85095006	1.4770501	9.1710005	0.2519	0.113575	-0.0669625	0
2637	22:01.6	0.85095006	1.4770501	9.1710005	0.293425	0.011	-0.1276	0
2638	22:01.6	0.86100006	1.45095	9.063001	0.293425	0.011	-0.1276	0
2639	22:01.6	0.86100006	1.45095	9.063001	0.292325	-0.016225	-0.1628	0
2640	22:01.6	1.3759501	1.8649501	9.348001	0.292325	-0.016225	-0.1628	0
2641	22:01.6	1.3759501	1.8649501	9.348001	0.1912625	0.2140875	-0.170225	0
2642	22:01.6	1.11705	1.5460501	9.504001	0.1912625	0.2140875	-0.170225	0
2643	22:01.6	1.11705	1.5460501	9.504001	0.1645875	0.523875	-0.285175	0
2644	22:01.6	0.32895002	1.9009501	9.142051	0.1645875	0.523875	-0.285175	0
2645	22:01.6	0.32895002	1.9009501	9.142051	0.1859	0.588775	-0.3595625	0
2646	22:01.7	-0.25695002	2.1739502	8.47395	0.1859	0.588775	-0.3595625	0
2647	22:01.7	-0.25695002	2.1739502	8.47395	0.17847499	0.3491125	-0.333025	0
2648	22:01.7	-1.07295	1.8870001	8.44695	0.17847499	0.3491125	-0.333025	0
2649	22:01.7	-1.07295	1.8870001	8.44695	0.015675	-0.070675	-0.1808125	0
2650	22:01.7	0.051	2.0689502	9.264001	0.015675	-0.070675	-0.1808125	0
2651	22:01.7	0.051	2.0689502	9.264001	-0.25575	-0.147125	0.1522125	0
2652	49:38.6	1.4280001	3.3120003	9.21105	-0.0518375	0.0704	0.06985	1
2653	49:38.6	1.4280001	3.3120003	9.21105	0.130075	-0.530475	-0.034375	1
2654	49:38.6	1.35195	3.6760502	8.13795	0.130075	-0.530475	-0.034375	1
2655	49:38.6	1.35195	3.6760502	8.13795	0.0438625	0.0829125	-0.201575	1
2656	49:38.7	0.64305	4.01595	9.54405	0.0438625	0.0829125	-0.201575	1
2657	49:38.7	0.64305	4.01595	9.54405	-0.10615	1.20725	-0.22495	1
2658	49:38.7	-0.31395	3.9529502	11.155951	-0.10615	1.20725	-0.22495	1
2659	49:38.7	-0.31395	3.9529502	11.155951	-0.0284625	1.76715	-0.16115	1
2660	49:38.7	-1.18095	3.49095	11.6640005	-0.0284625	1.76715	-0.16115	1
2661	49:38.7	-1.18095	3.49095	11.6640005	0.18755	1.319175	-0.085525	1
2662	49:38.7	-1.00905	3.3790503	11.41005	0.18755	1.319175	-0.085525	1
2663	49:38.7	-1.00905	3.3790503	11.41005	0.3716625	0.5330875	-0.0386375	1
2664	49:38.7	-0.17805001	3.65595	10.465951	0.3716625	0.5330875	-0.0386375	1
2665	49:38.7	-0.17805001	3.65595	10.465951	0.554675	0.2943875	-0.0493625	1
2666	49:38.8	0.11595	4.0230002	9.23295	0.554675	0.2943875	-0.0493625	1

■	A	B	C	D	E	F	G	H
4952	20:39.9	0.693	0.32700002	10.893001	-0.494175	0.0727375	-0.0286	2
4953	20:39.9	0.693	0.32700002	10.893001	-0.4057625	0.1909875	0.0287375	2
4954	20:40.0	0.81795	0.56205004	10.79205	-0.4057625	0.1909875	0.0287375	2
4955	20:40.0	0.81795	0.56205004	10.79205	-0.3069	0.0999625	0.1299375	2
4956	20:40.0	1.2529501	0.87600005	10.79205	-0.3069	0.0999625	0.1299375	2
4957	20:40.0	1.2529501	0.87600005	10.79205	-0.1706375	-0.06435	0.1522125	2
4958	20:40.0	1.2199501	1.0390501	10.737	-0.1706375	-0.06435	0.1522125	2
4959	20:40.0	1.2199501	1.0390501	10.737	-0.09295	0.003575	0.0735625	2
4960	20:40.0	0.897	1.0870501	10.732051	-0.09295	0.003575	0.0735625	2
4961	20:40.0	0.897	1.0870501	10.732051	-0.036575	0.1376375	-0.0265375	2
4962	20:40.0	0.822	1.1029501	10.984051	-0.036575	0.1376375	-0.0265375	2
4963	20:40.0	0.822	1.1029501	10.984051	0.0518375	0.253825	-0.12979999	2
4964	20:40.1	1.095	0.89295006	10.785001	0.0518375	0.253825	-0.12979999	2
4965	20:40.1	1.095	0.89295006	10.785001	0.1582625	0.49981248	-0.2701875	2
4966	20:40.1	0.64305	1.05795	9.757951	0.1582625	0.49981248	-0.2701875	2
4967	20:40.1	0.64305	1.05795	9.757951	0.2838	0.616	-0.445775	2
4968	20:40.1	-0.906	1.233	8.689051	0.2838	0.616	-0.445775	2
4969	20:40.1	-0.906	1.233	8.689051	0.3614875	0.19415	-0.5564625	2
4970	20:40.1	-2.082	1.02705	8.380051	0.3614875	0.19415	-0.5564625	2
4971	20:40.1	-2.082	1.02705	8.380051	0.2943875	-0.62865	-0.384175	2
4972	20:40.1	-0.82905006	0.64395005	9.571051	0.2943875	-0.62865	-0.384175	2
4973	20:40.1	-0.82905006	0.64395005	9.571051	-0.0631125	-0.9112125	0.076725	2
4974	19:36.6	-0.156	1.422	9.946951	0.195525	-0.470525	-0.0488125	3
4975	19:36.6	-0.156	1.422	9.946951	0.27005	0.0204875	0.076725	3
4976	19:36.7	0.039	1.64505	8.467051	0.27005	0.0204875	0.076725	3
4977	19:36.7	0.039	1.64505	8.467051	0.2242625	0.006875	0.2332	3
4978	19:36.7	0.29805002	1.5820501	7.9110003	0.2242625	0.006875	0.2332	3
4979	19:36.7	0.29805002	1.5820501	7.9110003	0.00935	-0.2413125	0.24915	3
4980	19:36.7	0.43005002	1.7380501	8.61495	0.00935	-0.2413125	0.24915	3
4981	19:36.7	0.43005002	1.7380501	8.61495	-0.25465	-0.43285	0.1555125	3
4982	19:36.7	0.62595004	1.45305	9.501	-0.25465	-0.43285	0.1555125	3
4983	19:36.7	0.62595004	1.45305	9.501	-0.4195125	-0.2633125	0.0661375	3
4984	19:36.7	0.17595	1.4790001	9.85995	-0.4195125	-0.2633125	0.0661375	3
4985	19:36.7	0.17595	1.4790001	9.85995	-0.5174125	0.134475	0.05335	3
4986	19:36.8	-0.123	1.5630001	10.15005	-0.5174125	0.134475	0.05335	3

Como sabemos la mayoría de modelos de inteligencia artificial funciona mediante estimaciones estadísticas, para este caso usamos 14 características estadísticas:

- Promedio
- Media armonica
- Curtosis
- Mediana
- Moda
- Valor mínimo y máximo
- Rango
- Valor RMS
- Asimetria

- Desviacion estándar
- Suma
- Media recortada
- Varianza

### Codigo Matlab

```
%Generamos un vector
VPP_L = table2array(datos(:, 7));

%Configuramos la frecuencia y la ventana
fhz = 996;
v = 0.1;

%Calculamos el tamaño del ventaneo
ventEstats = fix(fhz * v - 1);

%Creamos el vector tamaño para las características
VPP_S = size(M_PP,1);

%Creamos el vector para las características
VCS_Sense = zeros(VPP_S - ventEstats, 1);

%Creamos el vector para la etiqueta
VCS_L = zeros(VPP_S - ventEstats, 1);

%Con un ciclo for generamos las 14 características estadísticas
for n = 1 : (VPP_S - ventEstats)

    %Promedio
    VCS_Sense(n, 1) = mean(M_PP(n : n + ventEstats,1));
    VCS_Sense(n, 2) = mean(M_PP(n : n + ventEstats,2));
    VCS_Sense(n, 3) = mean(M_PP(n : n + ventEstats,3));
    VCS_Sense(n, 4) = mean(M_PP(n : n + ventEstats,4));
    VCS_Sense(n, 5) = mean(M_PP(n : n + ventEstats,5));
    VCS_Sense(n, 6) = mean(M_PP(n : n + ventEstats,6));

    %Media Armonica
    VCS_Sense(n, 7) = harmmean(M_PP(n : n + ventEstats,1));
    VCS_Sense(n, 8) = harmmean(M_PP(n : n + ventEstats,2));
    VCS_Sense(n, 9) = harmmean(M_PP(n : n + ventEstats,3));
    VCS_Sense(n, 10) = harmmean(M_PP(n : n + ventEstats,4));
    VCS_Sense(n, 11) = harmmean(M_PP(n : n + ventEstats,5));
    VCS_Sense(n, 12) = harmmean(M_PP(n : n + ventEstats,6));

    %Curtosis
    VCS_Sense(n, 13) = kurtosis(M_PP(n : n + ventEstats,1));
    VCS_Sense(n, 14) = kurtosis(M_PP(n : n + ventEstats,2));
    VCS_Sense(n, 15) = kurtosis(M_PP(n : n + ventEstats,3));
    VCS_Sense(n, 16) = kurtosis(M_PP(n : n + ventEstats,4));
    VCS_Sense(n, 17) = kurtosis(M_PP(n : n + ventEstats,5));
    VCS_Sense(n, 18) = kurtosis(M_PP(n : n + ventEstats,6));

    % Mediana
    VCS_Sense(n, 19) = median(M_PP(n : n + ventEstats,1));
    VCS_Sense(n, 20) = median(M_PP(n : n + ventEstats,2));
    VCS_Sense(n, 21) = median(M_PP(n : n + ventEstats,3));
    VCS_Sense(n, 22) = median(M_PP(n : n + ventEstats,4));
    VCS_Sense(n, 23) = median(M_PP(n : n + ventEstats,5));
    VCS_Sense(n, 24) = median(M_PP(n : n + ventEstats,6));

    % Moda
    VCS_Sense(n, 25) = mode(M_PP(n : n + ventEstats,1));
    VCS_Sense(n, 26) = mode(M_PP(n : n + ventEstats,2));
    VCS_Sense(n, 27) = mode(M_PP(n : n + ventEstats,3));
    VCS_Sense(n, 28) = mode(M_PP(n : n + ventEstats,4));
    VCS_Sense(n, 29) = mode(M_PP(n : n + ventEstats,5));
    VCS_Sense(n, 30) = mode(M_PP(n : n + ventEstats,6));

    % Minimo
    VCS_Sense(n, 31) = min(M_PP(n : n + ventEstats,1));
    VCS_Sense(n, 32) = min(M_PP(n : n + ventEstats,2));
    VCS_Sense(n, 33) = min(M_PP(n : n + ventEstats,3));
```

```

VCS_Sense(n, 34) = min(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 35) = min(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 36) = min(M_PP(n : n + ventEstats,6));

% Maximo
VCS_Sense(n, 37) = max(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 38) = max(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 39) = max(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 40) = max(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 41) = max(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 42) = max(M_PP(n : n + ventEstats,6));

% Rango
VCS_Sense(n, 43) = range(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 44) = range(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 45) = range(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 46) = range(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 47) = range(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 48) = range(M_PP(n : n + ventEstats,6));

% Valor eficiente o Valor RMS
VCS_Sense(n, 49) = rms(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 50) = rms(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 51) = rms(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 52) = rms(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 53) = rms(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 54) = rms(M_PP(n : n + ventEstats,6));

% Asimetria
VCS_Sense(n, 55) = skewness(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 56) = skewness(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 57) = skewness(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 58) = skewness(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 59) = skewness(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 60) = skewness(M_PP(n : n + ventEstats,6));

% Desviacion Estandar
VCS_Sense(n, 61) = std(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 62) = std(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 63) = std(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 64) = std(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 65) = std(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 66) = std(M_PP(n : n + ventEstats,6));

% Suma
VCS_Sense(n, 67) = sum(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 68) = sum(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 69) = sum(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 70) = sum(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 71) = sum(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 72) = sum(M_PP(n : n + ventEstats,6));

% Media recortada
VCS_Sense(n, 73) = trimmean(M_PP(n : n + ventEstats,1),1);
VCS_Sense(n, 74) = trimmean(M_PP(n : n + ventEstats,2),1);
VCS_Sense(n, 75) = trimmean(M_PP(n : n + ventEstats,3),1);
VCS_Sense(n, 76) = trimmean(M_PP(n : n + ventEstats,4),1);
VCS_Sense(n, 77) = trimmean(M_PP(n : n + ventEstats,5),1);
VCS_Sense(n, 78) = trimmean(M_PP(n : n + ventEstats,6),1);

% Varianza
VCS_Sense(n, 79) = var(M_PP(n : n + ventEstats,1));
VCS_Sense(n, 80) = var(M_PP(n : n + ventEstats,2));
VCS_Sense(n, 81) = var(M_PP(n : n + ventEstats,3));
VCS_Sense(n, 82) = var(M_PP(n : n + ventEstats,4));
VCS_Sense(n, 83) = var(M_PP(n : n + ventEstats,5));
VCS_Sense(n, 84) = var(M_PP(n : n + ventEstats,6));

%Agregamos la etiqueta a nuestra matriz
VCS_Sense(n, 85) = M_PP(n,7);
end

surf(VCS_Sense);

%Limpiamos nuestro Workspace
clear("v", "fhz", "ventEstats", "VPP_S", "n");

```



Este codigo hace exactamente lo mismo que el preprocesamiento pero ahora guarda todos los datos en un solo vector y va agregando columnas, al final se hacen 84 columnas de datos + 1 columna de la etiqueta, son 14 características por 6 ejes.

Al final se le agrega la etiqueta pero ahora debemos regresar los datos de la etiqueta de numerico a carácter de nuevo usando Excel y su herramienta “Buscar y reemplazar”:

	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	
1	STD_Az	STD_Gx	STD_Gy	STD_Gz	SUM_Ax	SUM_Ay	SUM_Az	SUM_Gx	SUM_Gy	SUM_Gz	MRECORTAD	MRECORTAD	MRECORTAD	MRECORTAD	MRECORTAD	MRECORTAD	VAR_Ax	VAR_Ay	VAR_Az	VAR_Gx	VAR_Gy	VAR_Gz	L	
2	0.45078605	0.21579032	0.17654029	0.0747176	70.9170068	94.9436963	975.837504	-23.5878426	-24.9178941	-17.7676277	0.7163334	0.95902724	9.85694448	-0.23826104	-0.2516959	-0.17947099	0.3403039	0.50985647	0.20320806	0.04656546	0.03116648	0.00558272	BR	
3	0.45674527	0.21544617	0.17983666	0.07331827	72.8563614	93.178025	976.434578	-23.7657259	-25.5259483	-17.8851499	0.73592284	0.94119217	9.86297554	-0.24005784	-0.25783786	-0.18065808	0.35697464	0.52632689	0.20861624	0.04641705	0.03234123	0.00537557	BR	
4	0.46301451	0.21505741	0.1831151	0.07195949	74.8349585	91.3488203	977.067895	-23.9562828	-26.1317914	-17.9950624	0.75590867	0.92271536	9.86937267	-0.24198266	-0.26395749	-0.18176831	0.3743533	0.54307644	0.21438243	0.04624969	0.03353114	0.00517817	BR	
5	0.46957965	0.21462509	0.1862592	0.07067691	76.8529797	89.4560824	977.737454	-24.1683592	-26.7376872	-18.0950846	0.77629089	0.90359679	9.8761359	-0.24412484	-0.27007765	-0.18277863	0.39243824	0.56005856	0.22050505	0.04606393	0.03469249	0.00499523	BR	
6	0.47646493	0.21414677	0.18927562	0.06948077	78.9146677	87.4942035	978.449499	-24.401955	-27.3496358	-18.1852166	0.79711786	0.88377983	9.88332827	-0.24648439	-0.27619834	-0.18368906	0.41119094	0.57733512	0.22701883	0.04585884	0.03582526	0.00482758	BR	
7	0.48360632	0.21361928	0.19217036	0.06838129	81.0254162	85.4589307	979.205011	-24.6570703	-27.9496372	-18.2654582	0.81843855	0.86322152	9.89059571	-0.24906132	-0.28231957	-0.18449958	0.43048486	0.59490659	0.23387508	0.0456332	0.03692945	0.004676	BR	
8	0.49097889	0.21305511	0.19467269	0.06740891	83.1850436	83.350264	980.00399	-24.93503	-28.5402812	-18.3350916	0.84025297	0.84192186	9.8990302	-0.25196899	-0.28831123	-0.18520295	0.45030274	0.61271571	0.24106027	0.04539248	0.03789746	0.00454396	BR	
9	0.49848144	0.21245259	0.19681389	0.0665713	85.399427	81.1673745	980.843766	-25.2358342	-29.1231608	-18.3941166	0.86262047	0.81987247	9.90751278	-0.25490742	-0.29417334	-0.18579916	0.47055074	0.6306901	0.24948374	0.0451361	0.03873571	0.00443174	BR	
10	0.50610536	0.2118093	0.19862151	0.06587524	87.666277	78.9132532	981.72835	-25.5594828	-29.690683	-18.4425332	0.88551795	0.79710357	9.91644798	-0.25817659	-0.29990589	-0.18628821	0.49104175	0.64867554	0.25614263	0.04486318	0.03945051	0.00433955	BR	
11	0.51382104	0.21113476	0.19996593	0.06531271	89.9855938	76.5879	982.657744	-25.9052634	-30.2304927	-18.4817152	0.90894539	0.77361515	9.9253853	-0.26166933	-0.30535851	-0.18668399	0.51173972	0.66660152	0.26401207	0.04457789	0.03998637	0.00426575	BR	
12	0.52167981	0.21042659	0.20090052	0.06488469	92.3467893	74.1940924	983.639764	-26.2731758	-30.7425899	-18.5116624	0.93279585	0.74943528	9.93575519	-0.26538562	-0.31053121	-0.18698649	0.5323601	0.68435145	0.27214983	0.04427935	0.04036102	0.00412100	BR	
13	0.52967432	0.20968165	0.20147379	0.06459093	94.740106	71.7315574	984.677743	-26.6632203	-31.2269746	-18.5323749	0.95697077	0.72456119	9.94623983	-0.26932546	-0.31542399	-0.18719571	0.55265352	0.70189921	0.28055489	0.0439664	0.04059169	0.00417199	BR	
14	0.53776709	0.20889056	0.2017539	0.06437228	97.165544	69.2002952	985.771681	-27.0721842	-31.6808746	-18.5480791	0.98147014	0.69899288	9.9572897	-0.27345641	-0.32000884	-0.18735433	0.57258764	0.71916919	0.28919345	0.04363527	0.04070464	0.00414379	BR	
15	0.54589668	0.20804811	0.20178303	0.06422642	99.6177183	66.6015269	986.921871	-27.5000675	-32.1042899	-18.5587749	1.00623958	0.6727427	9.96890779	-0.27777846	-0.32428576	-0.18746237	0.59200079	0.73613015	0.29800319	0.04328402	0.04071639	0.00412503	BR	
16	0.55394867	0.20714843	0.20160085	0.06415062	102.093387	63.9379601	988.122062	-27.9468703	-32.4972205	-18.5644624	1.03124633	0.64583798	9.98103093	-0.28229162	-0.32825475	-0.18751982	0.61071399	0.752730872	0.30685912	0.04291047	0.04064629	0.0041153	BR	
17	0.56188995	0.20615774	0.20127691	0.06411117	104.592549	61.2095948	989.372253	-28.4028217	-32.8645538	-18.5675068	1.05649039	0.61827874	9.99365912	-0.28689719	-0.33196519	-0.18755057	0.62869833	0.76882928	0.31572031	0.04250102	0.0405124	0.00411024	BR	
18	0.56958188	0.20506982	0.20083648	0.06410612	107.111256	58.4196235	990.661747	-28.8679216	-33.2062899	-18.5679082	1.08193188	0.59009721	10.0066843	-0.29159517	-0.33541707	-0.18755463	0.64574884	0.78437483	0.32442352	0.04205363	0.04033529	0.0041096	BR	
19	0.57683475	0.20387804	0.20030378	0.06413343	109.648044	55.5695431	991.975999	-29.3421703	-33.5224288	-18.5656665	1.107556	0.56130852	10.0199596	-0.29638556	-0.33861039	-0.18753199	0.6617463	0.79924622	0.33273832	0.04156625	0.04012161	0.0041131	BR	
20	0.58364113	0.20255199	0.19970455	0.06418848	112.202913	52.6593536	993.315008	-29.8142814	-33.8163857	-18.5610152	1.13336275	0.53191266	10.0334849	-0.30115436	-0.34157965	-0.187485	0.67666442	0.81336482	0.34063697	0.04102731	0.03988191	0.00412016	BR	
21	0.58985837	0.20109075	0.19905685	0.06426955	114.774772	49.6907095	994.667269	-30.284255	-34.0881607	-18.553954	1.15934113	0.50192636	10.0477441	-0.30590157	-0.34432486	-0.18741368	0.69049291	0.8266282	0.3479329	0.04043749	0.03962363	0.00413058	BR	
22	0.59543703	0.19949307	0.19837818	0.06437487	117.36158	46.6674079	996.025429	-30.7520911	-34.3377538	-18.5444832	1.1854705	0.47138796	10.0608629	-0.31062718	-0.346846	-0.18731801	0.70327911	0.83888465	0.35454526	0.03979749	0.0393539	0.00414412	BR	
23	0.60038836	0.19774574	0.1976838	0.06449972	119.963336	43.5894487	997.38949	-31.2125744	-34.5656816	-18.5330568	1.21175087	0.44029746	10.0746413	-0.31527853	-0.3491483	-0.18720259	0.71500201	0.85005426	0.36046618	0.03910338	0.03907889	0.00416021	BR	
24	0.60468631	0.19585175	0.19698973	0.06464324	122.575759	40.4623895	998.751691	-31.6657049	-34.7719441	-18.5196749	1.23813897	0.40871101	10.0884009	-0.31985561	-0.35123176	-0.18706742	0.7256921	0.85995161	0.36564553	0.03835791	0.03880495	0.00417875	BR	
25	0.60833924	0.19381375	0.19631148	0.06480459	125.193142	37.2918151	1000.10802	-32.1114827	-34.9564314	-18.5043374	1.26457719	0.376685	10.1021012	-0.32435841	-0.35309638	-0.1869125	0.73536041	0.86840316	0.37007663	0.03756377	0.0385382	0.00419964	BR	
26	0.61136521	0.19163043	0.1956582	0.06497257	127.815486	34.0777256	1001.45846	-32.5497244	-35.122826	-18.4882485	1.29106551	0.34421945	10.1157421	-0.3287851	-0.35477602	-0.18674999	0.74400031	0.87533661	0.37376742	0.03672222	0.03828213	0.00422144	BR	
27	0.61378982	0.18934044	0.19504048	0.06514705	130.435759	30.8248785	1002.79775	-32.9804299	-35.2707982	-18.4714082	1.31753292	0.31136241	10.1292702	-0.33313566	-0.35627069	-0.18657988	0.75158886	0.88064381	0.37673794	0.03583602	0.03804079	0.00424414	BR	
28	0.61566051	0.18683655	0.19446849	0.06532786	133.044547	27.5405177	1004.12713	-33.4035994	-35.4004579	-18.4538165	1.34388431	0.27818705	10.1426982	-0.3374101	-0.35758038	-0.18640219	0.75804922	0.88424709	0.37903787	0.0349079	0.03781799	0.00426773	BR	
29	0.61699235	0.18421661	0.19391919	0.0655089	135.64185	24.2246433	1005.44659	-33.8214049	-35.5200413	-18.4360638	1.3701197	0.24469337	10.1560262	-0.34163035	-0.3587883	-0.18622287	0.7633972	0.8869123	0.38067956	0.03395376	0.03760465	0.00429142	BR	
30	0.61783465	0.18144281	0.19339679	0.06569014	138.208021	20.8899099	1006.76196	-34.2338466	-35.6295482	-18.4181499	1.39604062	0.211700919	10.1693128	-0.34579643	-0.35989443	-0.18604192	0.7675555	0.886147	0.38171965	0.03292149	0.03740232	0.00431519	BR	
31	0.61822934	0.17851322	0.19290537	0.06587186	140.723699	17.5471885	1008.07938	-34.6409244	-35.7289788	-18.4000749	1.4214515	0.17274433	10.18262	-0.34990833	-0.36089878	-0.18585934	0.77053232	0.88446557	0.38220751	0.03186697	0.03721248	0.00433906	BR	
32	0.61817543	0.17538574	0.19240766	0.06607806	143.188882	14.196479	1009.39883	-35.0474855	-35.8264538	-18.3793124	1.44635234	0.14339878	10.1959478	-0.35401501	-0.36188337	-0.18564962	0.772739746	0.88103314	0.38214086	0.03076016	0.03702071	0.00436631	BR	
33	0.61769301	0.1720501	0.1919043	0.06630896	145.588562	10.8528503	1010.73321	-35.4535299	-35.9219732	-18.3558624	1.47059154	0.10962475	10.2094263	-0.35811646	-0.36284821	-0.18541275	0.77327764	0.87594359	0.38154466	0.02960124	0.03682726	0.00439688	BR	
34	0.61677468	0.16849446	0.19139596	0.06656357	147.911293	7.5431396	1012.09306	-35.8590577	-36.0155371	-18.3297249	1.49405346	0.07619333	10.2231623	-0.3622127	-0.3637933	-0.18514874	0.77332176	0.86928277	0.380411	0.02839038	0.03663241	0.00443071	BR	
35	0.61539654	0.16465945	0.19086827	0.0669244	150.157073	4.26734704	1013.47841	-36.2676382	-36.1093774	-18.2942929	1.51673811	0.04310452	10.2371556	-0.36633978	-0.36474119	-0.18477266	0.77261895	0.86110684	0.3787129	0.02711274	0.0364307	0.00447888	BR	
36	0.613533																							

Primero importamos el archivo de Excel anterior para tenerlo a la mano en el Workspace:

Ya con los cambios hechos guardamos el archivo en formato CSV y lo importamos al workspace para trabajar con el:

```
PFinal = readtable('C:\Users\solda\OneDrive\Escritorio\SmartWatch\datosProcesados_nano33.csv');
```

Una vez instalada la abrimos y generamos una nueva sesión desde una variable del Workspace:

**Data set**

Data Set Variable: PFinal (72214x85 table)

Response: L (cell, 8 unique)

**Predictors**

	Name	Type	Range
<input checked="" type="checkbox"/>	MEAN_Ax	double	-0.926182 .. 1.22539
<input checked="" type="checkbox"/>	MEAN_Ay	double	-0.987279 .. 0.980414
<input checked="" type="checkbox"/>	MEAN_Az	double	-0.702977 .. 0.94376
<input checked="" type="checkbox"/>	MEAN_Gx	double	-182.577 .. 60.4562
<input checked="" type="checkbox"/>	MEAN_Gy	double	-84.0784 .. 91.9075
<input checked="" type="checkbox"/>	MEAN_Gz	double	-120.404 .. 107.955

Add All Remove All

[How to prepare data](#) Refresh

**Validation**

Validation Scheme: Cross-Validation

Protects against overfitting. For data not set aside for testing, the app partitions the data into folds and estimates the accuracy on each fold.

Cross-validation folds: 5

[Read about validation](#)

**Test**

☐ Set aside a test data set

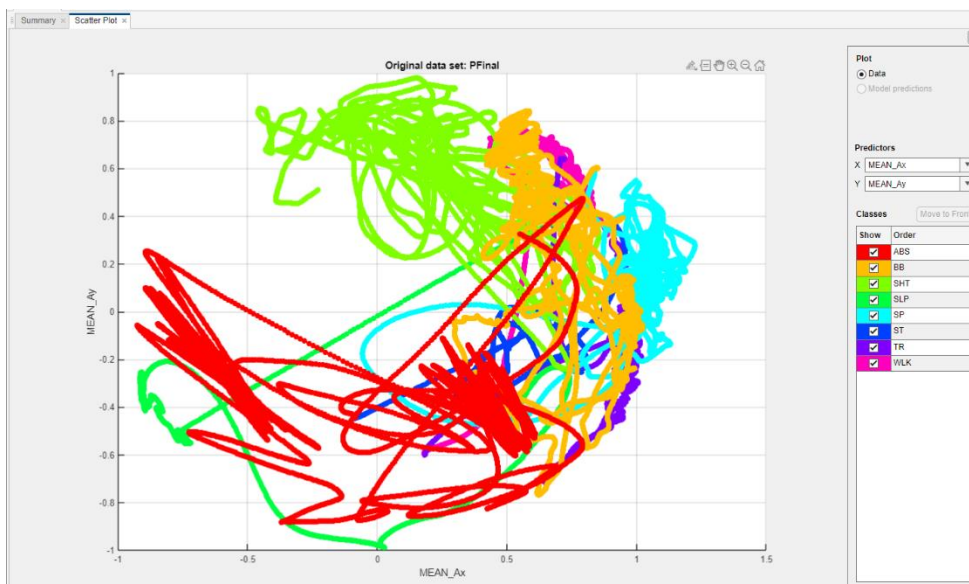
Percent set aside: 10

Use a test set to evaluate model performance after tuning and training models. To import a separate test set instead of partitioning the current data set, use the Test Data button after starting an app session.

[Read about test data](#)

Start Session Cancel

Como se observa se carga la variable que contiene el archivo y todos los datos por columna, damos clic en Start Session y podemos observar la grafica según la etiqueta:

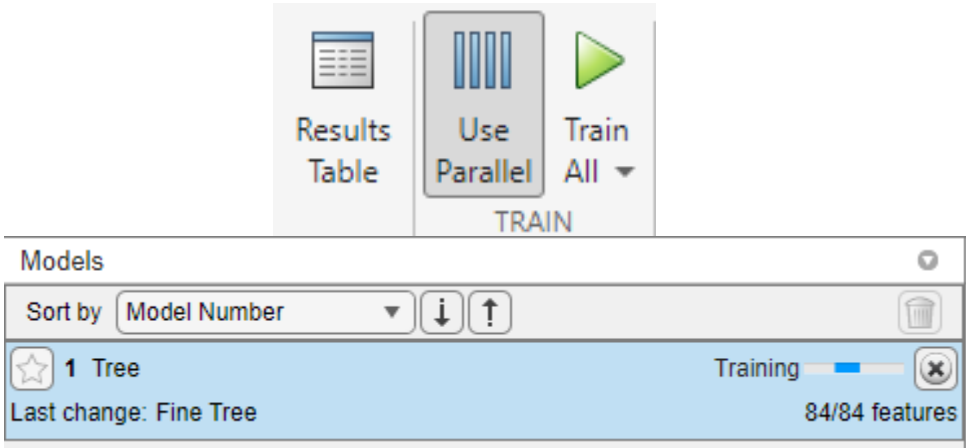




Una vez hecho eso damos clic en entrenar todos y vemos que se empieza a entrenar nuestro modelo de tipo Fine Tree (Arbol de decisión fino):



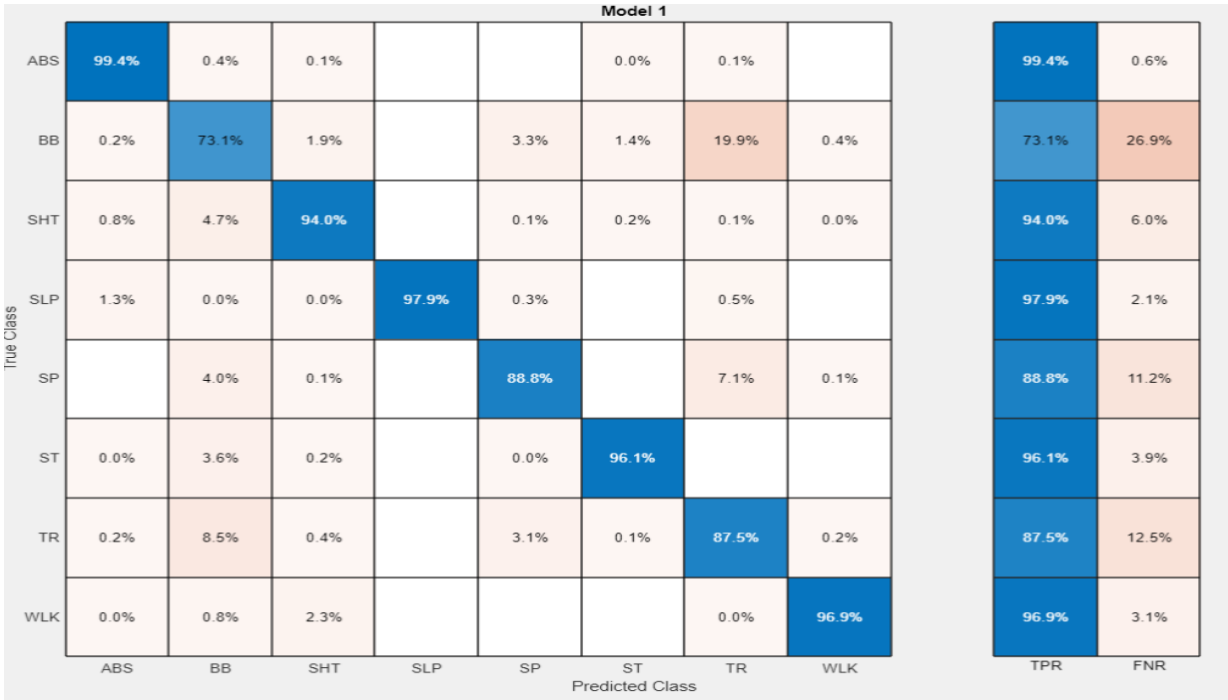
Ahora lo entrenamos:



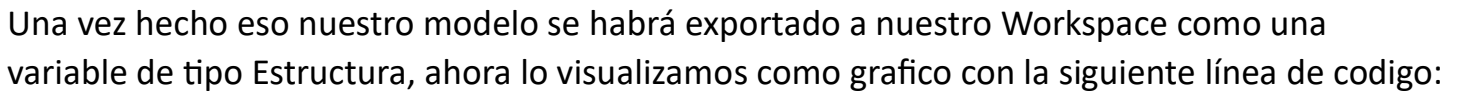
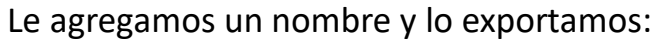
Una vez termine el proceso podemos ver la precisión que tiene nuestro modelo:



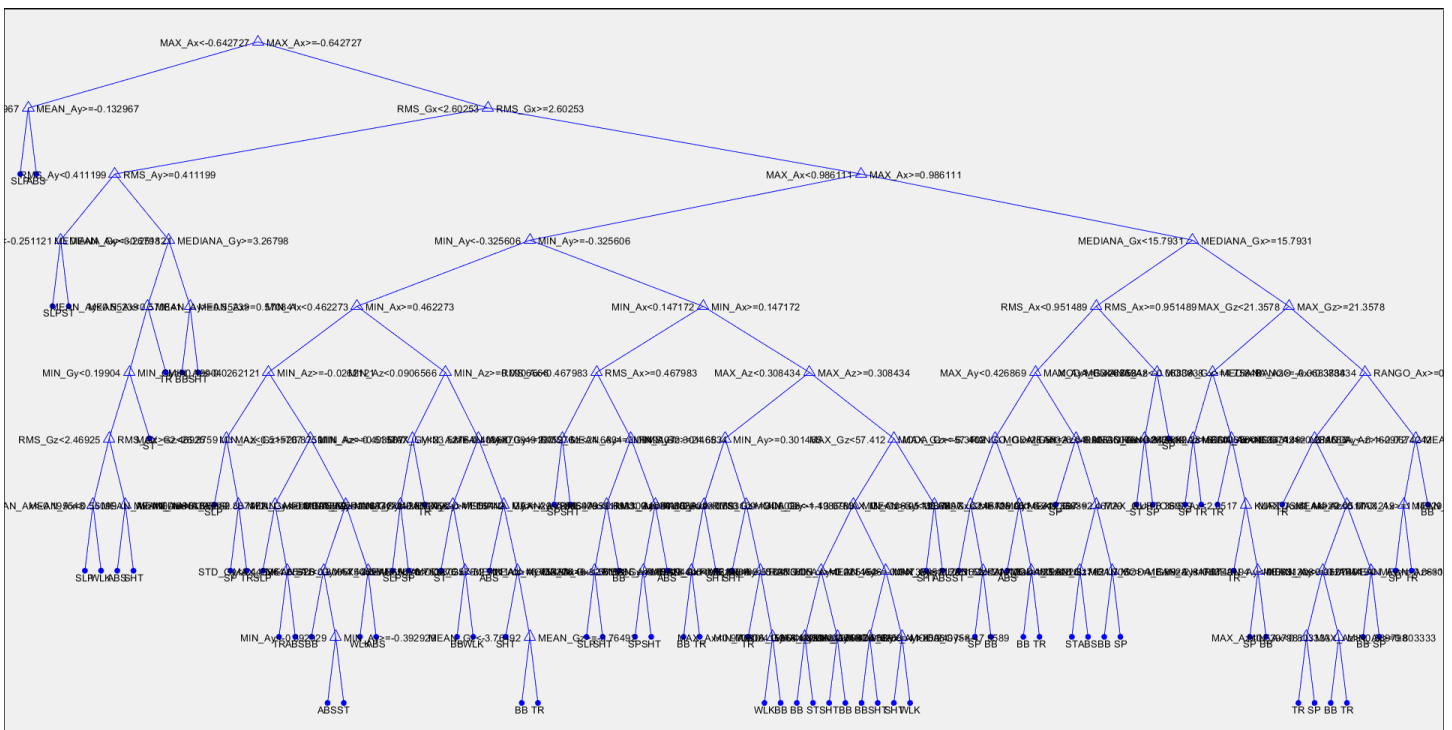
Al igual la matriz de confusión que contiene la validación por etiqueta:



Damos clic en la sección Export Model:



A continuacion se muestra el árbol de decisión:

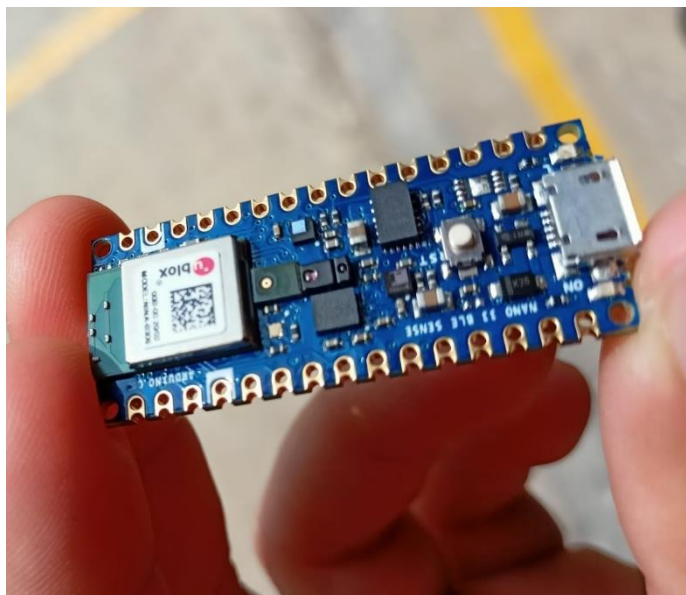


Como se observa el modelo toma decisiones en base a ciertas características estadísticas y los valores que encuentra a lo largo de la adquisición de los datos.

Ya implementado eso ahora toca el turno del Arduino.

## Arduino nano33 BLE Sense

El Arduino nano33 BLE Sense es una tarjeta de desarrollo muy interesante, contiene sensores en la misma placa lo cual facilita el uso de sensores, ahorra gastos y nos permite tener todo lo que se necesite en un circuito compacto, además incorpora un microprocesador capaz de soportar inteligencia artificial:



El nano33 contiene un IMU (Unidad de Medicion Inercial) que hace uso de un acelerómetro, un giroscopio y un magnetómetro, en este caso nos centraremos en los primeros dos.

Para adquirir los datos se configura el firmware del archivo el cual enviara los datos en un formato establecido mediante el monitor serial a una hoja de calculo de Excel la cual usara una macro para poder adquirir los datos:

### Sketch de Arduino

```
#include <Arduino_LSM9DS1.h>

void setup() {
  // open serial connection
  Serial.begin(9600);

  if(!IMU.begin()){
    Serial.println("Fallo al iniciar el IMU");
    while(1);
  }

  // define 2 rows: first named "Counter", second named "millis"
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,Ax,Ay,Az,Gx,Gy,Gz");
}
```

```

void loop() {

  // simple print out of number and millis
  // output "DATA,TIME,4711,

  float Ax, Ay, Az, Gx, Gy, Gz;

  if(IMU.gyroscopeAvailable() && IMU.accelerationAvailable()){
    IMU.readAcceleration(Ax, Ay, Az);
    IMU.readGyroscope(Gx, Gy, Gz);

    Serial.print("DATA," + String(Ax) + String(Ay) + String(Az) + String(Gx) + String(Gy) + String(Gz) + ",TIME,");

    Serial.print(Ax);
    Serial.print(",");
    Serial.print(Ay);
    Serial.print(",");
    Serial.print(Az);
    Serial.print(",");
    Serial.print(Gx);
    Serial.print(",");
    Serial.print(Gy);
    Serial.print(",");
    Serial.print(Gz);
    Serial.println();
  }
}

```

Y la macro viene incluida en la hoja de Excel, esta macro nos permite seleccionar el puerto COM al cual esta conectada nuestra tarjeta, los bautios a los que trabaja asi como limpiar las celdas, etc:

	B	C	D	E	F	G	H	I	J	K	L	M	N
1	TIME	AX	AY	AZ	GX	GY	GZ						
2	04:35:43 p. m.	0.49		-0.68	-0.59	-5.07	4.03	2.32					
3	04:35:43 p. m.	0.48		-0.69	-0.58	-5.43	3.23	2.62					
4	04:35:43 p. m.	0.48		-0.69	-0.58	-6.29	2.81	3.23					
5	04:35:43 p. m.	0.47		-0.69	-0.58	-6.96	3.05	3.85					
6	04:35:43 p. m.	0.47		-0.69	-0.58	-7.2	3.36	3.6					
7	04:35:43 p. m.	0.48		-0.69	-0.58	-7.26	3.48	3.36					
8	04:35:43 p. m.	0.47		-0.69	-0.58	-7.63	3.54	3.3					
9	04:35:43 p. m.	0.48		-0.69	-0.58	-7.45	3.85	3.42					
10	04:35:43 p. m.	0.49		-0.7	-0.59	-6.96	3.3						
11	04:35:43 p. m.	0.48		-0.7	-0.59	-7.32	1.95						
12	04:35:43 p. m.	0.47		-0.7	-0.58	-7.45	0.37						
13	04:35:43 p. m.	0.47		-0.7	-0.57	-7.93	-0.18						
14	04:35:43 p. m.	0.46		-0.69	-0.57	-8.24	-0.06						
15	04:35:43 p. m.	0.47		-0.69	-0.57	-8.3	-0.06						
16	04:35:44 p. m.	0.48		-0.7	-0.57	-7.51	-0.92						
17	04:35:44 p. m.	0.47		-0.7	-0.56	-7.32	-2.38						
18	04:35:44 p. m.	0.46		-0.68	-0.55	-7.45	-2.87						
19	04:35:44 p. m.	0.45		-0.69	-0.54	-7.02	-2.38						
20	04:35:44 p. m.	0.47		-0.69	-0.54	-6.9	-1.4						
21	04:35:44 p. m.	0.47		-0.69	-0.55	-6.71	-0.43						
22	04:35:44 p. m.	0.48		-0.68	-0.55	-5.62	-0.18						
23	04:35:44 p. m.	0.48		-0.69	-0.55	-4.33	-0.18						
24	04:35:44 p. m.	0.48		-0.68	-0.54	-4.39	-0.12						
25	04:35:44 p. m.	0.48		-0.68	-0.53	-4.64	0						
26	04:35:44 p. m.	0.48		-0.68	-0.53	-5.25	1.22						
27	04:35:44 p. m.	0.49		-0.68	-0.53	-5.68	2.38						
28	04:35:44 p. m.	0.49		-0.67	-0.53	-5.98	2.2	16.36					
29	04:35:44 p. m.	0.48		-0.66	-0.53	-5.31	1.34	16.97					
30	04:35:44 p. m.	0.46		-0.66	-0.52	-4.33	0.24	17.58					
31	04:35:44 p. m.	0.45		-0.65	-0.51	-3.42	-0.67	18.37					
32	04:35:44 p. m.	0.46		-0.66	-0.5	-2.62	-0.37	19.29					
33	04:35:44 p. m.	0.46		-0.66	-0.5	-2.01	-0.12	20.39					
34	04:35:44 p. m.	0.45		-0.65	-0.5	-1.28	0.67	22.09					
35	04:35:44 p. m.	0.44		-0.64	-0.51	-0.37	3.3	23.93					
36	04:35:44 p. m.	0.45		-0.64	-0.53	2.38	6.9	25.39					
37	04:35:44 p. m.	0.48		-0.65	-0.56	6.04	11.29	26.49					

**Control**  
☐ Download Data  
☐ Clear Stored Data  
☐ User1  
☐ User2













**Settings**  
Port:   
Baud:   
  
☒ Reset on Connect  
☐ Use 1st Worksheet at the time of "Connect"  
☐ Use active Worksheet at the time of "Connect"

**Controller Messages:**  

PLX-DAQ Status

Aquí se muestra una hoja de calculo ya usada donde se exponen los datos obtenidos, asi como la hora, al igual que la ventana de la macro.

De aquí en adelante el procedimiento es el mismo, se generan archivos de datos por separado según cada actividad a realizar y al final se juntan todos en un solo archivo CSV:

	abdominales.csv	07/06/2023 05:56 p. m.	Archivo de valores...	605 KB
	acostado.csv	07/06/2023 05:55 p. m.	Archivo de valores...	367 KB
	Actividad_nano33BLE.csv	07/06/2023 06:02 p. m.	Archivo de valores...	2,699 KB
	botebb.csv	07/06/2023 04:02 p. m.	Archivo de valores...	524 KB
	caminando.csv	07/06/2023 04:01 p. m.	Archivo de valores...	474 KB
	datosProcesados_nano33.csv	11/06/2023 02:23 p. m.	Archivo de valores...	71,865 KB
	datosProcesados_nano33.xlsx	07/06/2023 06:52 p. m.	Hoja de cálculo d...	67,468 KB
	nano33_Preprocesado.csv	07/06/2023 06:02 p. m.	Archivo de valores...	2,597 KB
	salto_payaso.csv	07/06/2023 05:46 p. m.	Archivo de valores...	836 KB
	sentado.csv	07/06/2023 05:56 p. m.	Archivo de valores...	758 KB
	tirando.csv	07/06/2023 05:57 p. m.	Archivo de valores...	750 KB
	trote.csv	07/06/2023 04:01 p. m.	Archivo de valores...	376 KB

Ya que juntamos todos los datos, los preprocesamos y terminamos todo el proceso nos queda un archivo de 72,215 filas:

		BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG		
72177	0.1030664	31.7241034	17.7143542	18.8105548	-38.4593939	45.8491919	13.2353535	2223.0601	-802.343838	-341.697879	-0.38847873	0.46312315	0.13369044	22.4551525	-8.10448322	-3.45149373	0.01007142	0.01016988	0.01062268	1006.41874	313.798346	353.836972	SHT	
72178	0.10903571	30.8617859	17.7305385	18.375561	-38.2907071	45.7234343	13.5542424	2348.67152	-756.543333	-401.197879	-0.38677482	0.46185287	0.13691154	23.7239547	-7.64185185	-4.05250383	0.01055669	0.00974398	0.01188879	952.449826	314.371995	337.66124	SHT	
72179	0.11551652	29.9236716	17.7387909	17.9093273	-38.0705051	45.6118182	13.9284849	2473.59051	-710.456869	-460.52101	-0.38455056	0.46072544	0.14069177	24.9857627	-7.17633201	-4.65172738	0.01117225	0.00936462	0.01334407	955.426123	314.664704	320.744005	SHT	
72180	0.12233709	28.9124736	17.7394109	17.4113882	-37.7937374	45.5166867	14.3556566	2596.81535	-664.062929	-519.491414	-0.38175492	0.45976635	0.14500663	26.2304581	-6.70770636	-5.24738802	0.01192149	0.00904116	0.01496636	935.93113	314.686699	303.156441	SHT	
72181	0.12942899	27.8330606	17.7321105	16.8797165	-37.4570707	45.4362626	14.8387879	2717.27444	-617.263232	-578.123232	-0.37935425	0.45896215	0.14988675	27.4472166	-6.23498215	-5.83962861	0.01279944	0.00876611	0.01675187	774.675262	314.427744	284.924883	SHT	
72182	0.13659055	26.6972802	17.7161516	16.5121276	-37.0689899	45.3675758	15.3707071	2833.40061	-569.947576	-636.428182	-0.37443424	0.45825834	0.15525967	28.6202081	-5.75704622	-6.42856749	0.01377089	0.00853137	0.01865709	712.744777	313.862028	266.085508	SHT	
72183	0.1433646	25.5248491	17.6898816	15.7099672	-36.6342424	45.3101011	15.9423232	2943.33331	-521.584943	-694.041414	-0.37004285	0.45767779	0.16103357	29.7306193	-5.26852872	-7.01051934	0.01481152	0.00833498	0.02063417	651.513722	312.93191	246.803037	SHT	
72184	0.15037325	24.3438558	17.6505499	15.076256	-36.1512121	45.2613131	16.5517172	3045.0795	-471.379697	-970.478283	-0.36516376	0.45718498	0.16718906	30.7583787	-4.76141108	-7.58058872	0.01590909	0.00816864	0.02265147	592.623314	311.541911	227.293495	SHT	
72185	0.15715098	23.1884603	17.5962364	14.410655	-35.6071717	45.2228283	17.1907071	3136.83212	-419.807677	-805.590808	-0.3596684	0.45679625	0.17364351	31.6851729	-4.24048158	-8.13728089	0.01707312	0.00803795	0.02469643	537.704689	309.627534	207.666796	SHT	
72186	0.16312417	22.0921962	17.5276418	13.7328361	-35.022332	45.1964447	17.8332323	3217.40616	-368.250505	-857.733535	-0.35376084	0.45652995	0.18013366	32.4990521	-3.71970207	-8.66397511	0.01824074	0.00794885	0.02664785	488.065132	307.218228	188.590787	SHT	
72187	0.1688846	21.0882415	17.445215	13.0592482	-34.4078788	45.1737374	18.4818182	3286.07657	-316.538485	-905.914444	-0.34755433	0.45630038	0.18668503	33.1926926	-3.19735843	-9.15065095	0.01937312	0.00787234	0.02852201	444.71393	304.335526	170.549363	SHT	
72188	0.17411071	20.2179366	17.3493545	12.4178848	-33.7558586	45.1545455	19.1355556	3341.9001	-264.285152	-948.691515	-0.34096827	0.45610652	0.19328844	33.7565667	-2.66954699	-9.58274258	0.02047351	0.00780799	0.03031454	408.764961	301.0001	154.203863	SHT	
72189	0.17880077	19.5216526	17.2422	11.8293383	-33.0753535	45.1337374	19.7735354	3384.2195	-212.34495	-985.468586	-0.33409448	0.45589634	0.19973268	34.1840353	-2.14489848	-9.95422814	0.0215201	0.00773847	0.03196972	381.094919	297.293459	139.933245	SHT	
72190	0.18287608	19.0293449	17.1261175	11.3242177	-32.3850505	45.1166667	20.3748485	3412.94657	-161.797374	-1015.24515	-0.32712172	0.45572391	0.20506055	34.4742077	-1.63431691	-10.2550015	0.02248066	0.00768164	0.03344366	362.115966	293.303901	128.237905	SHT	
72191	0.18617805	18.7498031	17.0055625	10.9356321	-31.7265657	45.0952525	20.9326263	3428.83778	-114.177677	-1037.09263	-0.32047036	0.4555076	0.21144067	34.634725	-1.15330987	-10.4756831	0.02329794	0.00761055	0.03473678	353.555117	289.189156	119.588049	SHT	
72192	0.18945859	18.6931466	16.8842914	10.69262	-31.1282828	45.0810101	21.4592929	3432.02384	-70.029495	-1050.32667	-0.3144271	0.45536374	0.21676054	34.6669075	-0.70736864	-10.6093603	0.0239519	0.00756344	0.03589456	349.433729	285.079294	114.332122	SHT	
72193	0.19253486	18.8610366	16.7599508	10.6514263	-30.5666667	45.0714141	22.0246465	3422.53606	-27.6021212	-1052.54717	-0.30875421	0.45526681	0.22247118	34.5710713	-0.27880931	-10.6317896	0.02448995	0.00753187	0.03706967	355.7387	280.89595	113.452882	SHT	
72194	0.19556455	19.2547829	16.6239987	10.851159	-30.0451515	45.0919192	22.6289899	3399.81717	16.250101	-1041.5897	-0.30348638	0.45547393	0.22857566	34.3415876	0.16414243	-10.5211081	0.02492437	0.0075988	0.03824549	370.746663	276.357331	117.747651	SHT	
72195	0.19844218	19.8580359	16.4740815	11.2192433	-29.540303	45.2072727	23.2570707	3363.7298	62.0419192	-1020.51333	-0.2938369	0.4563912	0.23491991	33.9770687	0.62668605	-10.3082155	0.02528578	0.00757048	0.0393793	394.34159	271.395361	125.87142	SHT	
72196	0.20098987	20.5447466	16.3107721	11.6607953	-29.100101	45.1787878	23.8687879	3320.71899	109.392929	-993.735455	-0.29394041	0.45676645	0.24019887	33.5426161	1.10497908	-10.0377319	0.02555531	0.00803657	0.04039657	422.086614	266.041287	135.974146	SHT	
72197	0.20318786	21.188025	16.1307861	12.0992476	-28.7072727	45.699495	24.4522222	3278.51101	158.984444	-965.457576	-0.28997245	0.46161106	0.24699214	33.1162728	1.60590348	-9.75209673	0.02576251	0.00950367	0.04128531	448.352403	260.202259	146.391793	SHT	
72198	0.20500998	21.7477051	15.5278963	12.492548	-28.3364647	46.01	24.9870707	3240.22849	211.808182	-938.550022	-0.28622692	0.46474748	0.25294605	32.7295807	2.13947658	-9.48035057	0.0259312	0.01043725	0.04202902	472.962678	253.69788	156.063631	SHT	
72199	0.2065271	22.2252276	15.7061898	12.8357583	-27.9940404	46.3327273	25.480303	3206.36697	265.916162	-913.765556	-0.28278609	0.46600725	0.2573768	32.3875452	2.68602184	-9.22955511	0.02606358	0.01138286	0.04265344	493.960744	246.684938	164.756691	SHT	
72200	0.20782988	22.6322382	15.4729264	13.130332	-27.6843434	46.6526263	25.9505051	3176.57444	319.049394	-891.437677	-0.27963983	0.47123865	0.26212631	32.0866106	3.22227115	-9.00442098	0.02616376	0.01229437	0.0431933	512.224715	239.410832	172.405619	SHT	
72201	0.20899337	22.9872208	15.2305952	13.3750437	-27.6807367	46.9564647	26.4190909	3149.88313	370.518687	-872.086465	-0.27663708	0.47480772	0.2668595	31.740731	3.742613	-8.80895419	0.02634408	0.01313589	0.04367623	528.412321	231.97013	176.891794	SHT	
72202	0.21005012	23.3104691	14.9783191	13.5769571	-27.0771717	47.2383589	26.8967677	3124.94647	420.440606	-855.523737	-0.27350679	0.47717543	0.27166452	31.5651158	4.24687481	-8.64165391	0.02631385	0.01389411	0.04412105	534.37797	234.350043	184.333705	SHT	
72203	0.21099515	23.6145082	14.717599	13.7479804	-26.7549495	47.4974748	27.3838384	3100.91737	468.362626	-841.039899	-0.27050272	0.47977247	0.27660443	31.3223977	4.73933562	-8.49535252	0.0263727	0.01456746	0.04452054	557.644598	216.607721	189.00966	SHT	
72204	0.21182774	23.9097288	14.4504254	13.8954399	-26.2429495	47.7400505	27.7494945	3077.06202	513.875859	-828.193884	-0.2669646	0.48227232	0.28155615	31.0814346	5.19066524	-8.36564024	0.0264178	0.01517822	0.04487099	571.653611	208.814795	193.083251	SHT	
72205	0.21252295	24.2021523	14.175286	14.024823	-26.0966667	47.9760606	28.3641414	3052.80343	557.331717	-816.64798	-0.26010611	0.48698067	0.28650648	30.8363994	5.29616331	-8.24896949	0.02644786	0.01574837	0.045166	585.744175	200.938734	196.695659	SHT	
72206	0.21308375	24.4958601	13.890998	14.1481285	-25.7505051	48.2119192	28.8477778	3027.93152	599.129798	-805.95202	-0.26010611	0.48698067	0.2913917	30.5851668	6.05181614	-8.1409295	0.02645989	0.01574881	0.04540468	604.500743	192.959827	199.991343	SHT	
72207	0.21351792	24.7892465	13.614012	14.2496239	-25.3957576	48.4543434	29.3247475	3002.57919	639.074748	-795.86222	-0.2565228	0.48943781	0.29620957	30.3290828	6.45530048	-8.03901235	0.02644988	0.01684787	0.0455899	614.506743	184.998115	203.051782	SHT	
72208	0.2138369	25.0793287	13.3113135	14.3494035	-25.039697	48.7076768	29.7934343	2977.0																
72209	0.21404924	25.36387	13.023887	14.414407	-24.683232	48.9734444	30.1297701	2771.881919	77.881919	-77.31001	-0.2496174	0.49691764	0.30560405	29.8098	7.20082747	-7.85152536	0.02635746	0.01798308	0.04678005	629.532828	189.535208	205.55208	SHT	
72210	0.21415919	25.639061	12.7345014	14.5245261	-24.321612	49.2510011	30.7670678	2926.39	24.1171	-769.01801	-0.24567289	0.49748495	0.31017853	29.559495	7.54789001	-7.767894	0.0267469	0.01856502	0.04584616	657.360687	162.167526	190.98189	SHT	
72211	0.21417313	25.9059441	12.471506	14.5977367	-23.960303	49.5332323	30.915619	2870.25354	780.25354	-61.6402	-0.2402326	0.50035368	0.31466687	29.387797	7.78935174	-7.69353174	0.02616863	0.0191426	0.04587013	671.117941	154.93157	213.084077	SHT	
72212	0.21409132	26.1663683	12.1587804	14.6605704	-23.5989899	49.8213113	30.888488	2766.90566	81															

## Sketch de Arduino

```
/*
 * Este programa pretende simular el proceso de decision de un modelo de Machine Learning.
 * Para ello ocupamos 3 fases que son muy importantes en la generacion de un modelo de ML:
 * - Adquisicion de los datos provenientes del IMU del Arduino nano33 BLE Sense
 * - Preprocesamiento de los datos (Suavizar la señal y obtener las características estadísticas)
 * - Implementación del árbol de decisión en una estructura if-else
 * - Manejo del LED RGB según la actividad que se está realizando
 *
 * Las características obtenidas en el programa se obtuvieron de acuerdo a la decisión que toma el
 * árbol de decisión, en el modelo entrenado en Matlab se asignan 14 características estadísticas, pero
 * no todas son tomadas en cuenta.
 *
 * El tipo de árbol de decisión que implementamos es de tipo Medium Tree, debido a que el tipo Fine Tree
 * es demasiado extenso y no encuentro como convertirlo a código. El árbol Coarse Tree tenía un 72.9% de
 * resultados positivos entonces no es conveniente implementarlo. El Medium Tree tiene una precisión de
 * 82.6% por lo cual no es del todo exacto pero es lo suficientemente preciso.
 */

//Biblioteca para usar el IMU del Arduino nano33 BLE Sense
#include <Arduino_LSM9DS1.h>

// Vectores de almacenamiento temporal de la señal
float rAx[10];
float rAy[10];
float rAz[10];

float rGx[10];
float rGy[10];
float rGz[10];

// Contadores
int contador = 0;
int n = 0;

// Variable de información de la suma de las señales
float p_AxSum = 0;
float p_AySum = 0;
float p_AzSum = 0;

float p_GxSum = 0;
float p_GySum = 0;
float p_GzSum = 0;

// Variables finales de la señal preprocesada
float f_Ax = 0;
float f_Ay = 0;
float f_Az = 0;

float f_Gx = 0;
float f_Gy = 0;
float f_Gz = 0;

//LED RGB
const int LED_BUILTIN_RED = LEDR;
const int LED_BUILTIN_GREEN = LEDG;
const int LED_BUILTIN_BLUE = LEDB;

void setup() {
  // Inicialización del IMU
  Serial.begin(9600);
  while (!Serial)
    ;

  if (!IMU.begin()) {
    Serial.println("Fallo al iniciar el IMU");
    while (1)
      ;
  }

  //Configuramos el LED RGB como salida
  pinMode(LED_BUILTIN_RED, OUTPUT);
  pinMode(LED_BUILTIN_GREEN, OUTPUT);
  pinMode(LED_BUILTIN_BLUE, OUTPUT);
}
```



```

}

void loop() {
    // Lectura de los valores del IMU
    float ax, ay, az, gx, gy, gz;
    if (IMU.accelerationAvailable() && IMU.gyroscopeAvailable()) {
        IMU.readAcceleration(ax, ay, az);
        IMU.readGyroscope(gx, gy, gz);

        // Almacenamiento de los datos en los vectores
        if (contador < 10) {
            rAx[contador] = ax;
            rAy[contador] = ay;
            rAz[contador] = az;
            rGx[contador] = gx;
            rGy[contador] = gy;
            rGz[contador] = gz;
            contador++;
        } else {
            contador = 0;
        }

        // Reinicio de las sumas
        p_AxSum = 0;
        p_AySum = 0;
        p_AzSum = 0;
        p_GxSum = 0;
        p_GySum = 0;
        p_GzSum = 0;

        // Suma de los valores en los vectores
        for (n = 0; n < 10; n++) {
            p_AxSum += rAx[n];
            p_AySum += rAy[n];
            p_AzSum += rAz[n];
            p_GxSum += rGx[n];
            p_GySum += rGy[n];
            p_GzSum += rGz[n];
        }

        // Cálculo de promedios
        f_Ax = p_AxSum / 10;
        f_Ay = p_AySum / 10;
        f_Az = p_AzSum / 10;
        f_Gx = p_GxSum / 10;
        f_Gy = p_GySum / 10;
        f_Gz = p_GzSum / 10;

        // Obtencion de las características estadísticas
        float max_Ax = obtenerMaximo(rAx, 10);
        float min_Ax = obtenerMinimo(rAx, 10);
        float max_Az = obtenerMaximo(rAz, 10);
        float min_Az = obtenerMinimo(rAz, 10);
        float min_Ay = obtenerMinimo(rAy, 10);
        float median_Gx = obtenerMediana(rGx, 10);
        float median_Gy = obtenerMediana(rGy, 10);
        float rms_Ax = obtenerRMS(rAx, 10);
        float rms_Ay = obtenerRMS(rAy, 10);
        float rms_Gx = obtenerRMS(rGx, 10);

        // Determinar la actividad usando el metodo conjunto
        String actividad = obtenerActividad(max_Ax, min_Ax, max_Az, min_Az, min_Ay, median_Gy, median_Gx, rms_Ax, rms_Ay, rms_Gx);

        // Control del LED según la actividad que se esta realizando
        controlarLED(actividad);
    }
    delay(10);
}

// Función para obtener el máximo
float obtenerMaximo(float arr[], int size) {
    float maxVal = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > maxVal) {
            maxVal = arr[i];
        }
    }
}

```

```

    }
}
return maxVal;
}

// Función para obtener el mínimo
float obtenerMinimo(float arr[], int size) {
    float minVal = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < minVal) {
            minVal = arr[i];
        }
    }
    return minVal;
}

// Función para obtener la mediana
float obtenerMediana(float arr[], int size) {
    float tempArr[size];
    for (int i = 0; i < size; i++) {
        tempArr[i] = arr[i];
    }
    // Ordenar el arreglo en orden ascendente
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (tempArr[j] > tempArr[j + 1]) {
                float temp = tempArr[j];
                tempArr[j] = tempArr[j + 1];
                tempArr[j + 1] = temp;
            }
        }
    }
    // Obtener la mediana
    if (size % 2 == 0) {
        return (tempArr[size / 2 - 1] + tempArr[size / 2]) / 2;
    } else {
        return tempArr[size / 2];
    }
}

// Función para obtener el valor RMS
float obtenerRMS(float arr[], int size) {
    float sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i] * arr[i];
    }
    float meanSquare = sum / size;
    return sqrt(meanSquare);
}

// Función para obtener la actividad según el árbol de decisión
String obtenerActividad(float max_Ax, float min_Ax, float max_Az, float min_Az, float min_Ay, float median_Gx, float
median_Gy, float rms_Ax, float rms_Ay, float rms_Gx) {
    if (max_Ax < -0.642727) {
        if (f_Ay < -0.132967) {
            return "acostado";
        } else {
            return "abdominales";
        }
    } else {
        if (rms_Gx < 2.60253) {
            if (rms_Ay < 0.411199) {
                if (f_Ax < -0.251121) {
                    return "acostado";
                } else {
                    return "sentado";
                }
            } else {
                if (median_Gy < 3.26798) {
                    if (f_Ay < 0.55239) {
                        return "abdominales";
                    } else {
                        return "trotando";
                    }
                } else {
                    if (f_Ax < 0.570841) {

```

```

        return "bote_balon";
    } else {
        return "Tiro_canasta";
    }
}
}
} else {
    if (max_Ax < 0.986111) {
        if (min_Ay < -0.325606) {
            if (min_Ax < 0.462273) {
                if (min_Az < -0.0262121) {
                    return "acostado";
                } else {
                    return "Trote";
                }
            } else {
                if (min_Az < 0.0906566) {
                    return "Trote";
                } else {
                    return "bote_balon";
                }
            }
        } else {
            if (min_Ax < 0.147172) {
                return "Tiro_canasta";
            } else {
                if (max_Az < 0.308434) {
                    return "caminando";
                } else {
                    return "bote_balon";
                }
            }
        }
    }
} else {
    if (median_Gx < 15.7931) {
        if (rms_Ax < 0.951489) {
            return "bote_balon";
        } else {
            return "salto_payaso";
        }
    } else {
        return "trote";
    }
}
}
}
}
}

```

```

// Función para controlar el LED según la actividad
void controlarLED(String actividad) {

```

```

    if (actividad == "trote") {
        // Encender el LED en morado
        analogWrite(LED_BUILTIN_RED, 255);
        analogWrite(LED_BUILTIN_GREEN, 0);
        analogWrite(LED_BUILTIN_BLUE, 255);
        Serial.println("trote");
    } else if (actividad == "caminando") {
        // Encender el LED en rosa
        analogWrite(LED_BUILTIN_RED, 255);
        analogWrite(LED_BUILTIN_GREEN, 192);
        analogWrite(LED_BUILTIN_BLUE, 203);
        Serial.println("caminando");
    } else if (actividad == "bote_balon") {
        // Encender el LED en naranja
        analogWrite(LED_BUILTIN_RED, 255);
        analogWrite(LED_BUILTIN_GREEN, 165);
        analogWrite(LED_BUILTIN_BLUE, 0);
        Serial.println("botando");
    } else if (actividad == "salto_payaso") {
        // Encender el LED en azul claro
        analogWrite(LED_BUILTIN_RED, 135);
        analogWrite(LED_BUILTIN_GREEN, 206);
        analogWrite(LED_BUILTIN_BLUE, 250);
        Serial.println("saltando de payaso");
    } else if (actividad == "acostado") {

```

```

// Encender el LED en verde
analogWrite(LED_BUILTIN_RED, 0);
analogWrite(LED_BUILTIN_GREEN, 255);
analogWrite(LED_BUILTIN_BLUE, 0);
Serial.println("acostado");
} else if (actividad == "sentado") {
// Encender el LED en azul marino
analogWrite(LED_BUILTIN_RED, 0);
analogWrite(LED_BUILTIN_GREEN, 0);
analogWrite(LED_BUILTIN_BLUE, 128);
Serial.println("sentado");
} else if (actividad == "abdominales") {
// Encender el LED en rojo
analogWrite(LED_BUILTIN_RED, 255);
analogWrite(LED_BUILTIN_GREEN, 0);
analogWrite(LED_BUILTIN_BLUE, 0);
Serial.println("abdominales");
} else if (actividad == "Tiro_canasta") {
// Encender el LED en café
analogWrite(LED_BUILTIN_RED, 165);
analogWrite(LED_BUILTIN_GREEN, 42);
analogWrite(LED_BUILTIN_BLUE, 42);
Serial.println("tiro a canasta");
} else {
// Apagar el LED
analogWrite(LED_BUILTIN_RED, 0);
analogWrite(LED_BUILTIN_GREEN, 0);
analogWrite(LED_BUILTIN_BLUE, 0);
Serial.println("NR");
}
}

```

El código se encuentra comentado para su correcta interpretación, a grandes rasgos:

- Obtiene los datos del IMU.
- Preprocesa los datos usando un ventaneo.
- Obtiene las características estadísticas más significativas.
- Toma la decisión en base al árbol de decisión transformado a una estructura if-else
- De acuerdo a la actividad se enciende el LED RGB que se incorpora en el nano33 en un color diferente según la actividad que se realice.

Cabe destacar que para lograr convertir el árbol de decisión a una estructura if-else nos basamos en un tipo de árbol medio, ya que el árbol fino era demasiado grande y por falta de tiempo no pudimos hacerlo.

## Prueba de concepto

