Trabajo de Programación N°4

AUTOR

Génesis Jesús Laguna Caldera

UNIVERSIDAD EIA
ENVIGADO
2024

UNIVERSIDAD EIA
Ser, Saber y Servir

1. INTRODUCCIÓN

El siguiente trabajo corresponde a la asignatura de programación con el profesor Andrés Quintero Zea, en donde se nos planteó como reto desarrollar un proyecto de analítica de datos, usando técnicas de aprendizaje supervisado respecto a una problemática orientada a el área de interés de cada estudiante, en este caso, ingeniería mecatrónica.

2. DESCRIPCIÓN DEL EJERCICIO

El estudiante debe ser capaz de desarrollar un proyecto de analítica de datos siguiendo los siguientes parámetros:

- Definir un problema predictivo.
- Obtener un dataset.
- Hacer análisis exploratorio de los datos.
- Realizar el procesado y limpieza de datos.
- Encontrar los mejores hiperparámetros para dos algoritmos predictivos.
- Realizar las curvas de aprendizaje.
- Realizar una evaluación diagnóstica.
- Comparar el desempeño de los dos modelos escogidos

3. DESCRICCIÓN DEL DATASET

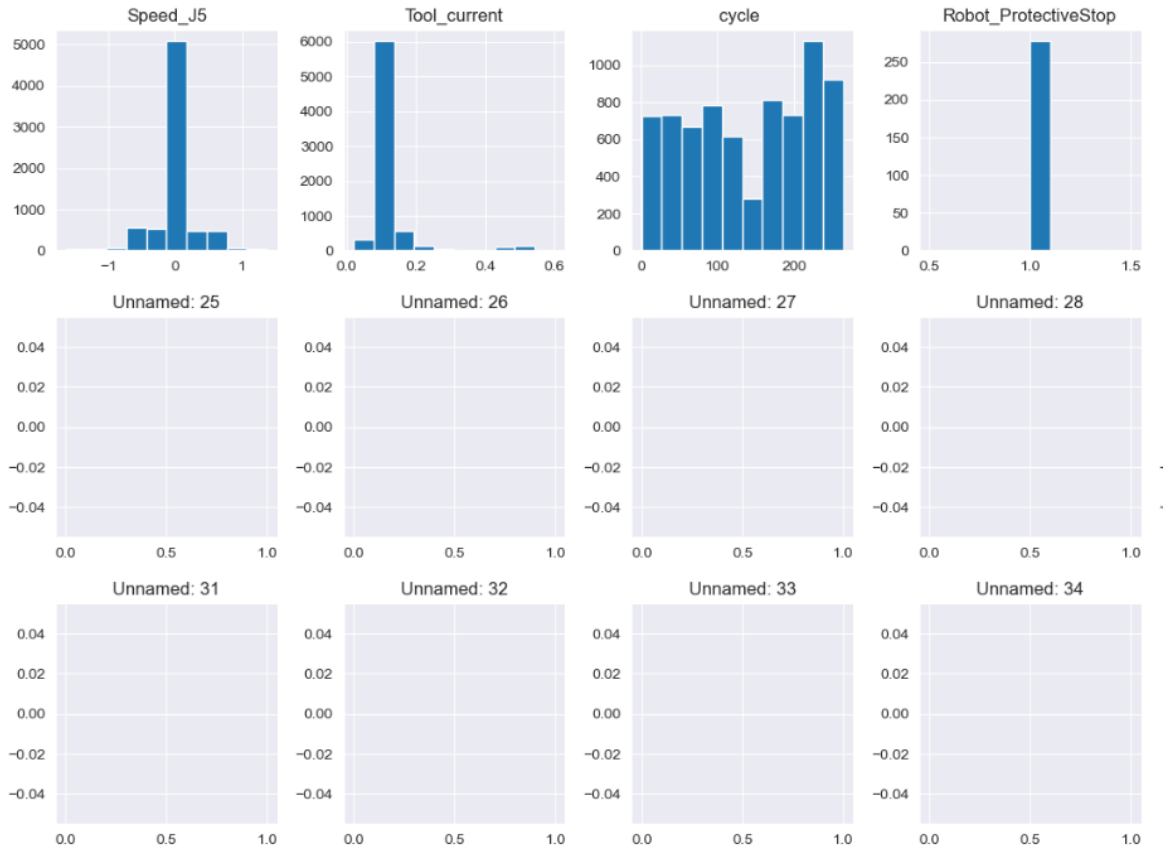
Para este ejercicio se utilizó un conjunto de datos de UR3 CobotOps del repositorio *UCI Machine Learning Repository* sobre el aprendizaje automático de sistemas robóticos y de automatización, en aplicaciones de robótica industrial para la detección de fallas y optimización operativa. Este dataset compara las siguientes variables:

- Corrientes eléctricas.
- Temperaturas.
- Velocidades en las juntas.
- Corriente de agarre.
- Recuento de ciclos de operación.
- Paradas de protección.
- Pérdidas de agarre.

El archivo consta de 7409 instancias y 20 características a comparar, entre los cuales se encuentran valores enteros, reales y categóricos.

4. PROCEDIMIENTO

Se realizó una exploración de datos en donde se determinaron 1 característica con datos tipo *object*, 2 características con datos tipo *int* y 31 características con datos tipo *float*. Se identificaron no muy a fondo cómo trabajaba el dataset con procedimientos como análisis de gráficos y valores estadísticos (moda, media, mediana, sesgo y curtosis) de las variables. De lo anterior se concluyeron varias problemáticas del dataset, como la aparición de datos erróneos y presencia de valores atípicos o nulos.



Luego de la exploración, se continuó con el procesado de datos, donde se identificaron datos faltantes y atípicos. Se realizó una gráfica para cada característica por separado. De ese procedimiento, se descartaron características con datos vacíos como "*Unnamed: 24…Unnamed 34*". Continuando con la limpieza de datos, se eliminaron los datos duplicados y se creó un nuevo dataframe sin los valores atípicos. Se plantearon dos modelos de aprendizaje autónomo en base al nuevo dataframe.

```
#Identificamos duplicados
new_data.duplicated().sum()
```

```
53
```

```
#Eliminamos duplicados
new_data_drop = new_data.drop_duplicates()
new_data_drop.shape
```

```
(7356, 19)
```

```
new_data_drop.duplicated().sum()
```

```
0
```

4.1 ERROR:

Al momento de generar el código para los modelos de aprendizajes atónomos dentro del ejercicio, se presentó una falla en el código el cual hablaba de valores nulos.

```
ValueError: Input y contains NaN.
```

Se intentó proseguir con el código con métodos como el "*data.dropna(inplace=True)*" y el "*data.isnull().any(axis=1)*" a pesar de que previamente ya se habían eliminado los valores nulos, como lo refleja el código:

```
#Eliminamos variables e instancias nulas
df_nulos = df.dropna(subset=['Current_J0', 'Temperature_T0', 'C
print(df_nulos.isnull().sum())
```

```
Num                0
Timestamp          0
Current_J0         0
Temperature_T0     0
Current_J1         0
Temperature_J1     0
Current_J2         0
Temperature_J2     0
```

UNIVERSIDAD EIA

Ser, Saber y Servir

```
df[['Current_J0', 'Temperature_T0', 'Current_J1', 'Temperature_J1', 'Current_J2', 'Temperature_J2', 'Current_J3', 'Temperature_J3', 'Current_J4', 'Temper
```

Temperature_J1	Current_J2	Temperature_J2	Current_J3	Temperature_J3	Current_J4	Temperature_J4	...	Unnamed: 25	Unnamed: 26	Unnamed: 27	Unnamed: 28	Unnamed: 29
7355.000000	7355.000000	7355.000000	7355.000000	7355.000000	7355.000000	7355.000000	...	0.0	0.0	0.0	0.0	0.0
37.659636	-1.199381	38.064064	-0.605312	40.936999	-0.022968	42.605167	...	NaN	NaN	NaN	NaN	NaN
3.247315	0.609984	3.311948	0.514937	3.182399	0.630789	3.677670	...	NaN	NaN	NaN	NaN	NaN
29.312500	-4.171966	29.375000	-3.333102	32.125000	-4.738406	32.250000	...	NaN	NaN	NaN	NaN	NaN
35.375000	-1.552803	35.750000	-0.830933	38.937500	-0.125809	40.375000	...	NaN	NaN	NaN	NaN	NaN
39.687500	-1.077137	40.187500	-0.571190	43.062500	-0.012325	45.062500	...	NaN	NaN	NaN	NaN	NaN
40.125000	-0.838721	40.437500	-0.388398	43.125000	0.086098	45.187500	...	NaN	NaN	NaN	NaN	NaN
40.500000	2.464940	40.937500	2.270268	43.437500	4.089389	45.375000	...	NaN	NaN	NaN	NaN	NaN

```
# Creamos un nuevo DataFrame eliminando los valores nulos
new_variables = ['Current_J0', 'Temperature_T0', 'Current_J1', 'Temperature_J1', 'Current_J2', 'Temperature_J2', 'Current_J3', 'Temperature_J3', 'Current
new_data = df[new_variables]
new_data.head()
```

J2	Current_J3	Temperature_J3	Current_J4	Temperature_J4	Current_J5	Temperature_J5	Speed_J0	Speed_J1	Speed_J2	Speed_J3	Speed_J4	Speed_J5	Tool_current
50	-0.998570	32.1250	-0.062540	32.2500	-0.152622	32.0000	2.955651e-01	-0.000490	0.001310	-0.132836	-0.007479	-0.152962	0.082732
75	-0.206097	32.1875	-1.062762	32.2500	-0.260764	32.0000	-7.390000e-30	-0.000304	0.002185	0.001668	-0.000767	0.000417	0.505895
75	-0.351499	32.1250	-0.668869	32.3125	0.039071	32.0625	1.369386e-01	0.007795	-2.535874	0.379867	0.000455	-0.496856	0.079420
75	-1.209115	32.1250	-0.819755	32.2500	0.153903	32.0000	-9.030032e-02	-0.004911	-0.009096	-0.384196	0.018411	0.425559	0.083325
75	-2.356471	32.1875	-0.966427	32.3125	0.178998	32.0000	1.268088e-01	0.005567	0.001138	-0.353284	0.014994	0.180989	0.086379

En vista de que el código parecía funcionar con normalidad, a simple vista no presentaba alguna incongruencia, se decidió explorar manualmente el dataset original para poder determinar específicamente las filas con valores nulos que impedían el correcto funcionamiento del programa, en donde se eliminaron la siguiente lista de filas:

808	807	2022-10-26T08:30:53.785Z	0,071988709	30,5625	-1,819340944	32,5	-0,923094749	32,8125	-0,523037076
809	808	2022-10-26T08:30:54.791Z	0,012650969	30,5625	-2,548998594	32,5	-1,426130056	32,875	-0,385982335
810	809	2022-10-26T08:30:55.806Z	-0,195183918	30,5625	-2,71168828	32,5	-1,85068965	32,875	-0,208415523
811	810	2022-10-26T08:30:56.806Z	5,308233738	30,5625	-1,990114212	32,5	-0,93177402	32,8125	-1,055757165
812	811	2022-10-26T08:30:57.820Z	0,130929425	30,5625	-1,455661774	32,5625	-0,841269135	32,875	-1,009340763
813	812	2022-10-26T08:30:58.822Z	-4,488698483	30,5625	-1,568230391	32,5625	-1,276592016	32,875	0,149745002
814	813	2022-10-26T08:30:59.825Z	0,084576957	30,5625	-1,939658642	32,5625	-1,657230616	32,8125	-0,378848046
815	814	2022-10-26T08:31:00.829Z							
816	815	2022-10-26T08:31:01.837Z	-0,073029056	30,5625	-1,578484774	32,5625	-0,786987126	32,875	-0,657846451
817	816	2022-10-26T08:31:02.850Z	-0,073779047	30,625	-1,636137486	32,5625	-0,76189065	32,875	-0,673082888
818	817	2022-10-26T08:31:03.857Z	-0,590647042	30,5625	-4,078726768	32,5625	-2,498441696	32,8125	-0,543398619
819	818	2022-10-26T08:31:04.861Z	-0,34976685	30,5625	-3,248551369	32,5625	-1,143917441	32,8125	-0,501746774
820	819	2022-10-26T08:31:05.865Z	0,196712196	30,625	-2,520131111	32,5625	-1,640829325	32,875	-1,523337007
821	820	2022-10-26T08:31:06.866Z	0,111074872	30,625	-2,311326981	32,5	-0,784653068	32,875	-0,292731583
822	821	2022-10-26T08:31:07.870Z	-0,350803256	30,625	-2,735381126	32,5625	-0,625871003	32,875	-0,314958572
823	822	2022-10-26T08:31:08.874Z	-0,208270669	30,5625	-2,774007797	32,5625	-0,938729048	32,875	-0,498708963
824	823	2022-10-26T08:31:09.875Z	-0,107494637	30,625	-2,513792515	32,5	-0,989869952	32,875	-0,422404498
825	824	2022-10-26T08:31:10.881Z	0,045985527	30,5625	-1,127392054	32,5625	-0,962841392	32,875	-0,811466694
826	825	2022-10-26T08:31:11.884Z	0,166319147	30,625	-2,483165979	32,5625	-1,303452492	32,875	-0,777454615
827	826	2022-10-26T08:31:12.891Z	0,559508562	30,625	-2,805593014	32,5625	-1,910330057	32,875	-0,483397603
828	827	2022-10-26T08:31:13.895Z	0,500189662	30,625	-3,135270596	32,5625	-1,698276281	32,875	-0,879472911
829	828	2022-10-26T08:31:14.901Z	0,526107073	30,625	-2,79365921	32,5625	-1,710398436	32,875	-2,056708097
830	829	2022-10-26T08:31:15.904Z	0,283674151	30,5625	-1,953180671	32,5625	-0,683012486	32,9375	-0,319044232
831	830	2022-10-26T08:31:16.905Z	-0,143554077	30,625	-2,271070957	32,625	-0,856633544	32,9375	-0,511373281
832	831	2022-10-26T08:31:17.916Z	-0,093877479	30,625	-2,221180439	32,625	-0,909261465	32,9375	-0,530019224
833	832	2022-10-26T08:31:18.918Z	-0,023404196	30,625	-2,606589794	32,625	-1,325214624	32,9375	-0,313906431
834	833	2022-10-26T08:31:19.922Z	-0,176613003	30,625	-2,738044262	32,625	-1,813218355	32,9375	-0,217785418
835	834	2022-10-26T08:31:20.930Z	5,423906326	30,625	-1,966696501	32,5625	-0,717392027	32,9375	-1,126256943

Filas a Eliminar
814, 948, 977, 1047, 1316, 2090, 2238, 2317, 2393, 2669, 3392, 3441, 3490, 3984, 4106, 4289, 4459, 4677, 4699, 4805, 4822, 4921, 4993, 5223, 5344, 5465, 5587, 5648, 5710, 5778, 5847, 5905, 5969, 6037, 6104, 6292, 6348, 6413, 6470, 6528, 6591, 6645, 6697, 6763, 6819, 6937, 6993, 7040, 7091, 7137, 7188, 7250, 7306, 7366

```
# Eliminamos las filas con valores nulos
df = newdf.drop(index=[814, 948, 977, 1047, 1316, 2090, 2238, 2317, 2393, 2669, 3392, 3441, 3490, 3984, 4106, 4289, 4
```

Sin embargo, el error persistía, lo que dificultó seguir con el planteamiento de los modelos, ya que el dataset contaba con muchas incongruencias.