DEEP LEARNING

Sistema de Predicción de futuras fallas en servidores de data centers basada en la telemetría

Profesor: Raul Ramos Pollan

Estudiante: Andrés Felipe Osorio Henker

CC: 98636911

1. Contexto de aplicación:

La telemetría es una de las funcionalidades disponibles en los equipos de cómputo y en entornos de redes. La información suministrada por dichos equipos es recolectada y almacenada en sistemas de gestión o NMS (Network Management System) para tener herramientas de visualización de desempeño y de detección o generación de alarmas de posibles fallas en los equipos. Los datos también sirven para tener históricos con los que es posible calcular estadísticas de desempeño, disponibilidad y ANS (Acuerdos de Niveles de Servicio). Estas herramientas son por lo general reactivas y pocas veces la información suministrada por los equipos se usa para hacer análisis predictivo. Muchos de estos sistemas almacenan grandes cantidades de información de telemetría que en conjunto con las nuevas tecnologías de machine learning y deep learning se puede aprovechar para desarrollar herramientas que sirvan para mejorar la detección temprana de posibles fallas.

2. Objetivo:

Predecir la fecha en la que un determinado servidor pueda fallar, teniendo como insumo la información periódica (series de tiempo) de telemetría que se tiene a disposición de los servidores de la nube Azure y de las fallas registradas en fechas dadas.

3. Dataset:

El Dataset fue tomado de:

https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance

De dicho Dataset se tomaron dos archivos:

- PdM_telemetry.csv: Este es el archivo principal que contiene la información de telemetría en el tiempo de los servidores. Tiene un peso de 80.1MB.

- PdM_failures.csv: Este archivo tiene las fechas de las fallas de los servidores. Tiene un peso de 24.3.kB.

Los datos tienen la siguiente forma:

```
telemetry_data = pd.read_csv('/content/drive/My Drive/dataset/PdM_telemetry.csv')
print(telemetry_data.shape)
telemetry_data.head(3)
```

```
(876100, 6)
```

	datetime	machineID	volt	rotate	pressure	vibration
0	2015-01-01 06:00:00	1	176.217853	418.504078	113.077935	45.087686
1	2015-01-01 07:00:00	1	162.879223	402.747490	95.460525	43.413973
2	2015-01-01 08:00:00	1	170.989902	527.349825	75.237905	34.178847

Forma de los datos de telemetría

```
error_data = pd.read_csv('/content/drive/My Drive/dataset/PdM_failures.csv')
print(error_data.shape)
error_data.head(3)
```

```
(761, 3)
```

	datetime	machineID	failure
0	2015-01-05 06:00:00	1	comp4
1	2015-03-06 06:00:00	1	comp1
2	2015-04-20 06:00:00	1	comp2

Forma de los datos de fallas

Los datos son entregados por machineID que es el identificador del servidor, se trabajó con los datos de un único servidor, en este caso el 99 que es el que más fallas tiene por lo tanto más posibilidades de observación de la red, la idea es no mezclar datos entre servidores porque no se sabe de ante mano si las características son similares, como el modelo o antigüedad. Los datos de las fallas se deben unificar en el tiempo con los de la telemetría para tener un único dataset el cual luce de la siguiente forma:

	datetime	volt	rotate	pressure	vibration	failure
0	2015-01-01 06:00:00	168.596133	384.747105	110.921131	41.944692	0
1	2015-01-01 07:00:00	153.667693	441.288719	128.011446	39.271527	0
2	2015-01-01 08:00:00	178.319255	512.612661	128.526147	42.975412	0
3	2015-01-01 09:00:00	162.163821	493.356626	114.435754	40.044016	0
4	2015-01-01 10:00:00	180.201336	550.512296	110.771891	43.054530	0

Forma de los datos unificados

Contamos los campos sin falla=0, con falla=1. Al realizar el conteo se ve un desbalance donde hay muchos datos sin falla, lo que se espera en el escenario real, pero es un aspecto presente para seleccionar la arquitectura de la red neuronal.

```
datos["failure"].value_counts()
```

```
failure
0    8742
1      19
Name: count, dtype: int64
```

Conteo de no fallas y fallas

Luego de indexar los datos por el atributo de tiempo “datetime” se graficó las series de tiempo para observar relaciones entre las fallas y anomalías en la telemetría.

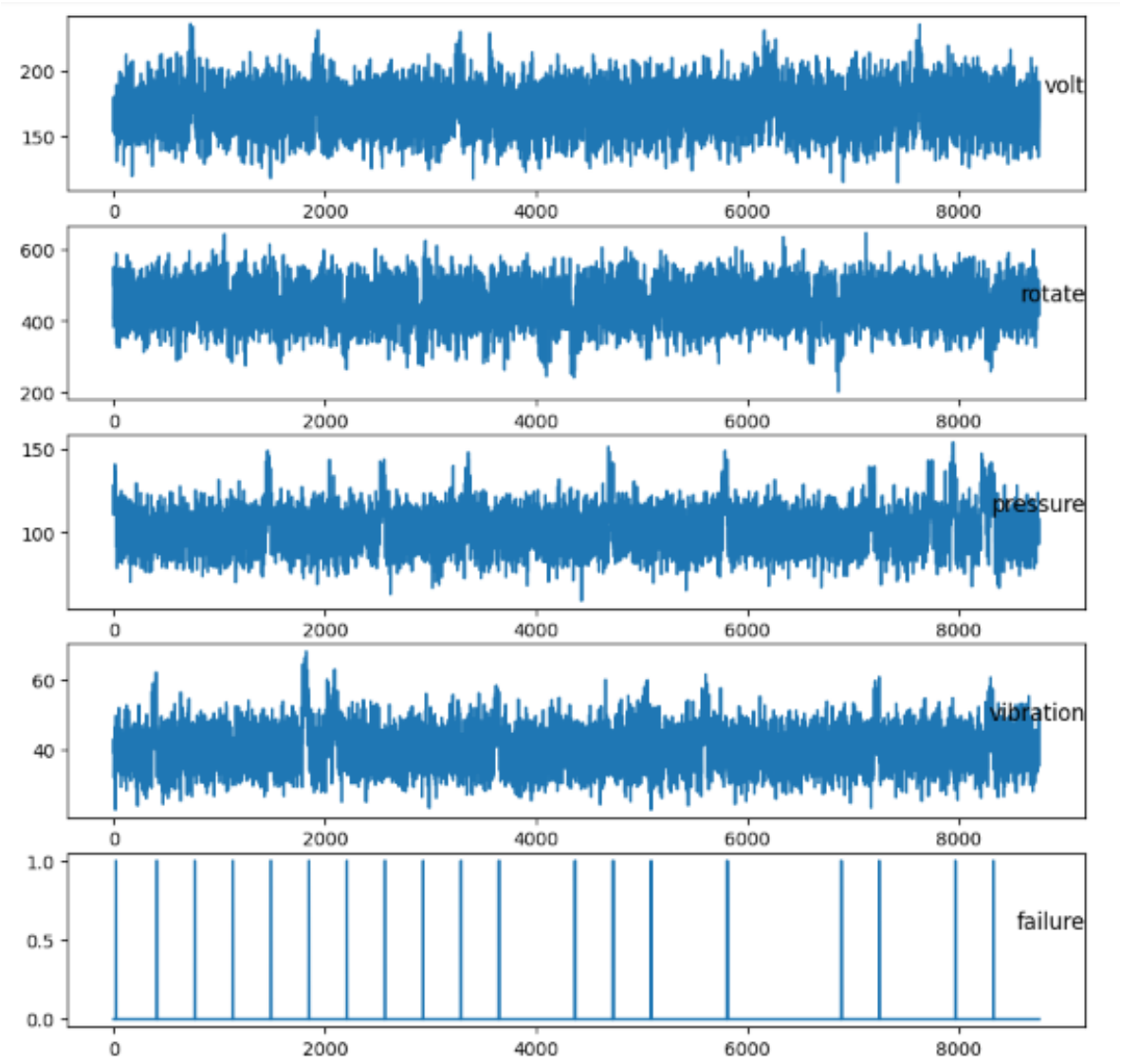

Figura 1: Gráficas de las series de tiempo.

4. Métricas de desempeño:

Las métricas para medir la precisión de la predicción de una falla se dividen en métricas de precisión estadísticas y de soporte de decisiones.

Métricas de precisión estadística: Evaluar la precisión de una técnica de filtrado comparando las calificaciones pronosticadas directamente con la calificación real del usuario. Se van a calcular las siguientes métricas:

El error cuadrático medio (RMSE): Se utilizará esta métrica para evaluar la precisión del modelo de las redes neuronales recurrentes. Un RMSE más bajo indica un mejor desempeño del modelo, pero siempre debe interpretarse en el contexto de la escala de los datos.

Observación visual: Comparar gráficamente la serie de tiempo a predecir comparando los datos de entrenamiento y pruebas para evidenciar posibles coincidencias. En nuestro caso al

ser una serie de tiempo binaria se puede observar fácilmente los puntos donde deben existir los aciertos de la red neuronal.

5. Referencias y resultados previos:

Realizando una búsqueda en revistas científicas se ve que los modelos de Redes Neuronales Recurrentes y Long Short Term Memory son comúnmente usados para realizar la predicción del mantenimiento de equipos informáticos o industriales. Esto nos muestra que estas técnicas nos pueden servir para la solución del problema planteado en el punto 2.

También se realizó un análisis de los datos y se implementó un entrenamiento usando una red neuronal profunda para realizar predicción de las fechas de falla de un solo servidor. Esto quedó plasmado en el notebook:

https://github.com/aosorih/telemetry_servers/01_exploracion_de_datos.ipynb

6. DESARROLLO

Tras explorar los datos y, según la documentación revisada, se eligen los modelos de redes neuronales recurrentes, los cuales son los más adecuados para desarrollar el problema. Los motivos por los que se elige este tipo de modelo de red neuronal son porque los datos están dados periódicamente en tiempos fijos, series de tiempo, y los predictores están muy desbalanceados, lo que puede ser un gran problema al usar una capa predictora al final de nuestra red neuronal.

6.1. Preparación de los datos: Para el desarrollo del sistema de predicción de fallas la primera parte es la preparación de los datos que consiste en los siguientes pasos. Primero convertir el dataframe de pandas a un array de numpy. Segundo convertir los datos en una forma adecuada para series de tiempo tipo [samples, look_back, features], donde el atributo look_back son los tiempos de observación hacia atrás, esto se hace para todas las características incluyendo la serie que queremos predecir. De forma separada generamos un array solo con los datos de la serie predictora. Para este paso se usó la función create_datasetMV:

```
def create_datasetMV(dataset, look_back=2):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back):
        a = dataset[i:(i+look_back)]
        dataX.append(a)
        dataY.append(dataset[i + look_back][-1])
    return np.array(dataX), np.array(dataY)

X_seq, y_seq = create_datasetMV(MVSeries, look_back=2)
X_seq.shape, y_seq.shape
```

```
((8759, 2, 5), (8759,))
```

Luego se escalan y se separan los datos de entrenamiento y pruebas. En este caso 67% para entrenamiento.

6.2. Modelos de deep learning: Se define una función que nos entrega un modelo de red neuronal, la función entrega uno de los 3 tipos comunes de redes neuronales usados en series de tiempo: SimpleRNN, LSTM y GRU. Se evaluaron los datos con estos 3 tipos de redes neuronales variando los tiempos de observación hacia atrás de 1 a 5. Para cada combinación se obtiene la métrica de desempeño RMSE para determinar cuál combinación es la mejor.

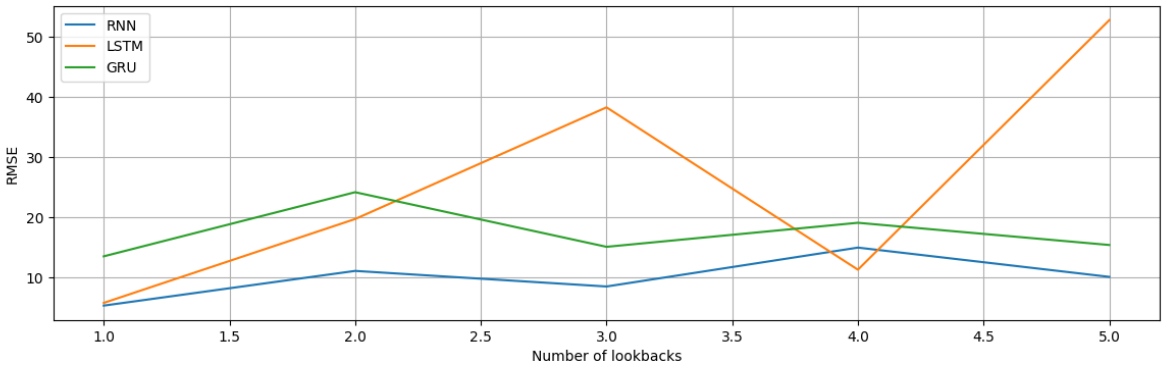

Figura 2: Gráficas de desempeño RMSE vs Modelos

De la Figura 2 vemos que las mejores combinaciones son los modelos RNN y LSTM con un valor de look_back=1.

Luego se corre de nuevo la red con los dos modelos que dieron mejor desempeño y se grafican los datos de entrenamiento y prueba para la serie predictora que es la que tiene las fallas de los servidores en el tiempo.

Corremos el modelo LSTN con los siguientes resultados gráficos:

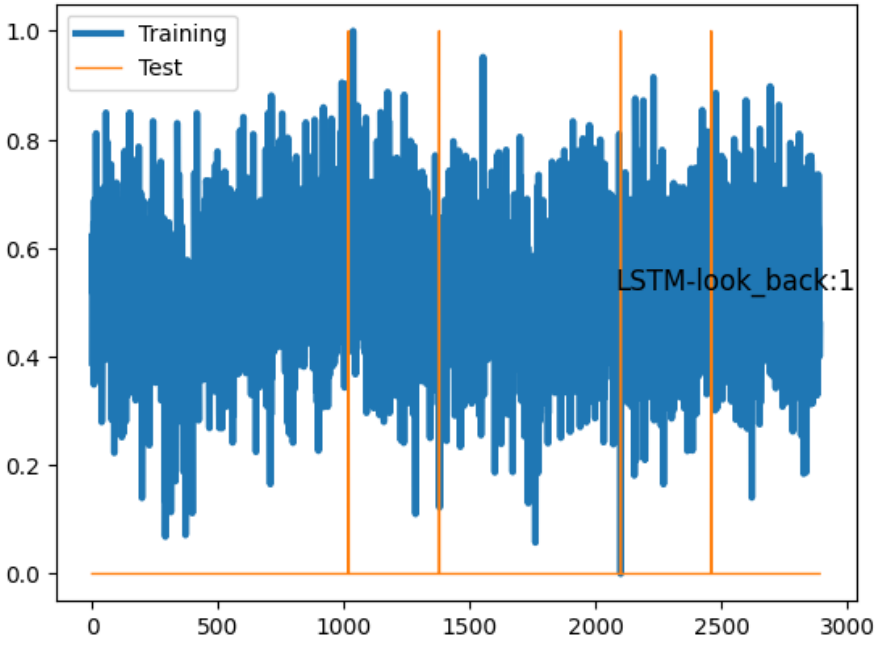

Figura 3: Comparación gráfica datos de entrenamiento y pruebas modelo LSTM.

De la Figura 3 vemos que de las 4 fallas que tenemos en pruebas hay coincidencias con picos que se dan en entrenamiento, sin embargo, se ven muchos otros picos en entrenamiento que no coinciden con las fallas reales de pruebas. Este modelo no dio una precisión adecuada.

Corremos el modelo RNN con los siguientes resultados gráficos:

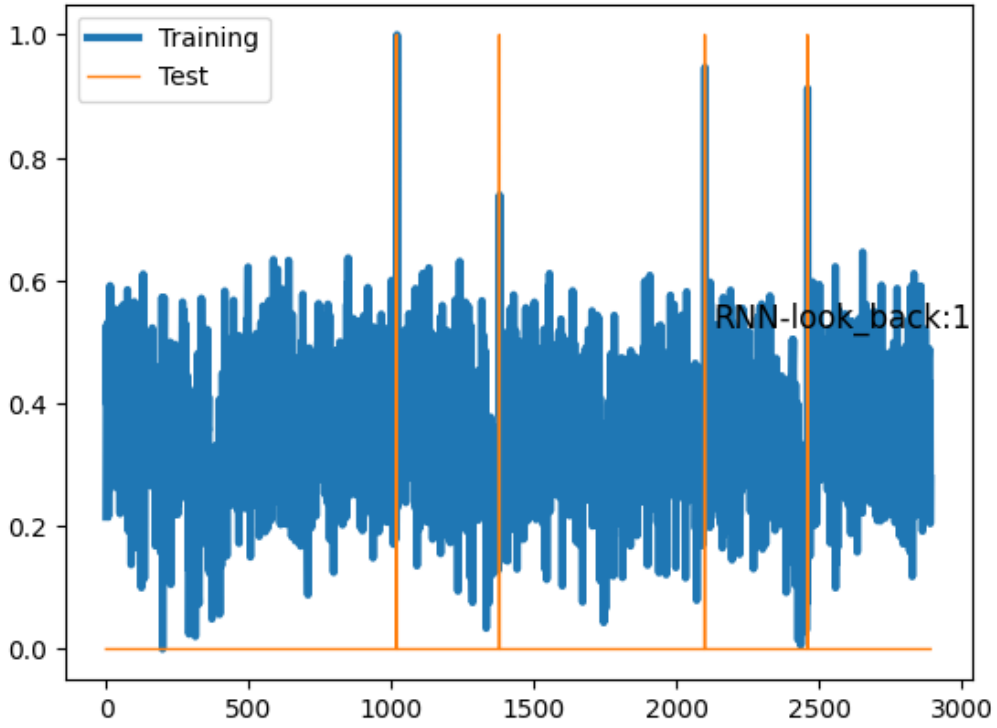

Figura 4: Comparación gráfica datos de entrenamiento y pruebas modelo RNN.

De la Figura 4 vemos que de las 4 fallas en pruebas hay coincidencias precisas con picos dados en entrenamiento, el modelo puedo identificar el tiempo en el que se presentaron. Este modelo dio una precisión muy acertada.

Referencias:

A. Cachada et al., "Maintenance 4.0: Intelligent and Predictive Maintenance System Architecture," 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, Italy, 2018, pp. 139-146, doi: 10.1109/ETFA.2018.8502489.

M. Vijayakumar, P. Shreeraj Nair, S. B. G Tilak Babu, K. Mahender, T. S Venkateswaran and N. L, "Intelligent Systems For Predictive Maintenance In Industrial IoT," 2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gautam Buddha Nagar, India, 2023, pp. 1650-1655, doi: 10.1109/UPCON59197.2023.10434814.

Q. Wang, S. Bu and Z. He, "Achieving Predictive and Proactive Maintenance for High-Speed Railway Power Equipment With LSTM-RNN," in IEEE Transactions on Industrial Informatics, vol. 16, no. 10, pp. 6509-6517, Oct. 2020, doi: 10.1109/TII.2020.2966033.

J. S. Rahhal and D. Abualnadi, "IOT Based Predictive Maintenance Using LSTM RNN Estimator," 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ICECCE49384.2020.9179459.