```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

np.random.seed(42)

time = np.arange(0, 1000)

temperature = 50 + 0.02*time + np.random.normal(0, 1, 1000)
vibration = 5 + 0.01*time + np.random.normal(0, 0.5, 1000)
pressure = 100 + 0.03*time + np.random.normal(0, 2, 1000)
acoustic = 60 + 0.015*time + np.random.normal(0, 1, 1000)

data = pd.DataFrame({
    'time': time,
    'temperature': temperature,
    'vibration': vibration,
    'pressure': pressure,
    'acoustic': acoustic
})

data.head()
```

1 to 5 of 5 entries Filter

index	time	temperature	vibration	pressure	acoustic
0	0	50.49671415301123	5.699677718293001	98.64964345005124	58.09219244212261
1	1	49.88173569882882	5.472316841456384	99.74096265856896	59.1546149892205
2	2	50.687688538100694	5.049815184960087	98.47516015800008	59.616394466578356
3	3	51.583029856408025	4.706531611147213	99.47407694072193	61.93268765734019
4	4	49.845846625276664	5.389111656806795	96.33277066609243	60.61655312453376

Show 25 per page

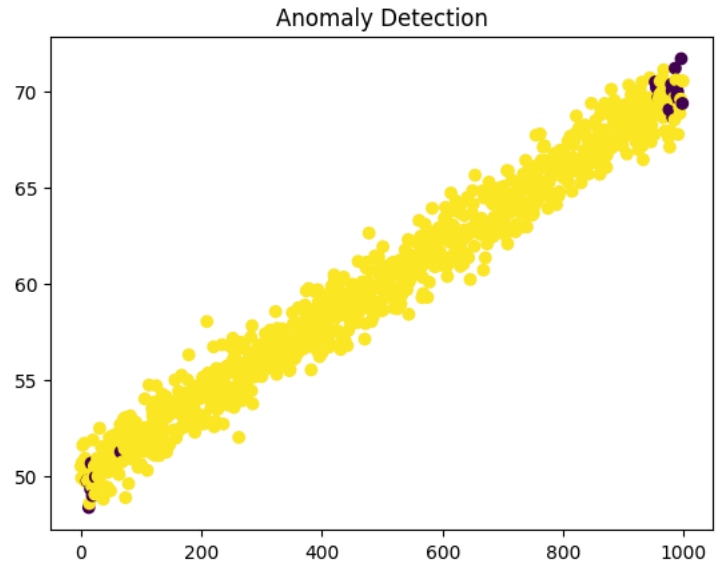
Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with data New interactive sheet

```
from sklearn.ensemble import IsolationForest

model_if = IsolationForest(contamination=0.02)
data['anomaly'] = model_if.fit_predict(data[['temperature','vibration','pressure','acoustic']])

plt.scatter(data['time'], data['temperature'], c=data['anomaly'])
plt.title("Anomaly Detection")
plt.show()
```




```
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[['temperature','vibration','pressure','acoustic']])

X = []
y = []
```

```
sequence_length = 20

for i in range(len(scaled_data)-sequence_length):
    X.append(scaled_data[i:i+sequence_length])
    y.append(scaled_data[i+sequence_length][0])

X = np.array(X)
y = np.array(y)

model = Sequential()
model.add(LSTM(50, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

model.fit(X, y, epochs=10, batch_size=32)
```

```
ɔoch 1/10
ısr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim`
 super().__init__(**kwargs)
./31 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.1642
ɔoch 2/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0062
ɔoch 3/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0028
ɔoch 4/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0021
ɔoch 5/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0019
ɔoch 6/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0020
ɔoch 7/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0018
ɔoch 8/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0018
ɔoch 9/10
./31 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0020
ɔoch 10/10
./31 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0019
ɛras.src.callbacks.history.History at 0x798449394da0>
```

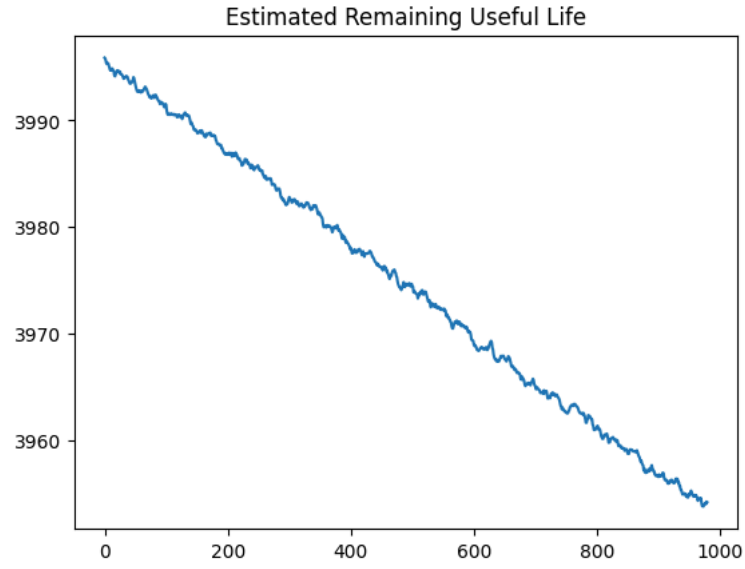
```
failure_threshold = 80

predictions = model.predict(X)

rul = (failure_threshold - predictions.flatten()) / 0.02

plt.plot(rul)
plt.title("Estimated Remaining Useful Life")
plt.show()
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```




```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y, predictions)
print("MSE:", mse)
```

```
MSE: 0.001861430985856315
```

```
pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.54.0-py3-none-any.whl.metadata (9.8 kB)
Requirement already satisfied: altair!=5.4.0,!=5.4.1,<7,>=4.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (5
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<7,>=5.5 in /usr/local/lib/python3.12/dist-packages (from streamlit) (6.2.6)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (8.3.1)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.12/dist-packages (from streamlit) (3.1
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.12/dist-packages (from streamlit) (2.0.2)
Requirement already satisfied: packaging>=20 in /usr/local/lib/python3.12/dist-packages (from streamlit) (26.0)
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<13,>=7.1.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (11.3.0)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.12/dist-packages (from streamlit) (5.29.6)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (18.1.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.12/dist-packages (from streamlit) (2.32.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (9.1.4)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.12/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in /usr/local/lib/python3.12/dist-packages (from streamlit) (6.5.1)
Requirement already satisfied: typing-extensions<5,>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (4.1
Requirement already satisfied: watchdog<7,>=2.1.5 in /usr/local/lib/python3.12/dist-packages (from streamlit) (6.0.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4.1,<7,>=4.0->strear
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4.1,<7,>=4
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4.1,<7,>=4
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.12/dist-packages (from gitpython!=3.1.19,<4,>=3.0.7
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->stre
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit) (20
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->s
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->streamlit) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->streaml
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->streaml
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.12/dist-packages (from gitdb<5,>=4.0.1->gitpython!=3
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->altair!=5.4.0,!=5.4.1
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->altair!=5.4.0
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.12/dist-packages (from jsonsche
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->altair!=
Requirement already satisfied: rpds-py>=0.25.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->altair!=5.4
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas<3,>=1
Downloading streamlit-1.54.0-py3-none-any.whl (9.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.1/9.1 MB 76.6 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 97.7 MB/s eta 0:00:00
Installing collected packages: pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.54.0
```

```
%%writefile app.py
import streamlit as st
import pandas as pd
import numpy as np

st.title("AI Predictive Maintenance - Launch Pad Systems")

time = np.arange(0, 1000)
temperature = 50 + 0.02*time + np.random.normal(0, 1, 1000)

data = pd.DataFrame({
    'time': time,
    'temperature': temperature
})

st.line_chart(data.set_index('time'))
```

```
Writing app.py
```

```
!pip install pyngrok
```

```
Collecting pyngrok
  Downloading pyngrok-7.5.0-py3-none-any.whl.metadata (8.1 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.12/dist-packages (from pyngrok) (6.0.3)
Downloading pyngrok-7.5.0-py3-none-any.whl (24 kB)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.5.0
```

```
!pip install pyngrok
```

```
Requirement already satisfied: pyngrok in /usr/local/lib/python3.12/dist-packages (7.5.0)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.12/dist-packages (from pyngrok) (6.0.3)
```

```
from pyngrok import ngrok

ngrok.set_auth_token("3AFzGmFdTHc85ZlgpSRvKxGZmjd_Fq9pufZW4TAaYcDvJkb9")
```

```
public_url = ngrok.connect(8501)
print("Public URL:", public_url)
```

```
Public URL: NgrokTunnel: "https://karlene-jarless-consuelo.ngrok-free.dev" -> "http://localhost:8501"
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

np.random.seed(42)

n_samples = 10000
time = np.arange(n_samples)

temperature = 70 + 0.01*time + np.random.normal(0, 2, n_samples)
vibration = 40 + 0.005*time + np.random.normal(0, 1.5, n_samples)
pressure = 25 + 0.008*time + np.random.normal(0, 1, n_samples)

# Simulate degradation trend
degradation = 0.0005 * time
RUL = 1000 - degradation*time
RUL[RUL < 0] = 0

data = pd.DataFrame({
    "time": time,
    "temperature": temperature,
    "vibration": vibration,
    "pressure": pressure,
    "RUL": RUL
})

data.to_csv("launchpad_sensor_data.csv", index=False)
data.head()
```

1 to 5 of 5 entries Filter

index	time	temperature	vibration	pressure	RUL
0	0	70.99342830602247	38.98225790426917	25.348286247666817	1000.0
1	1	69.73347139765764	39.54675080541952	25.291323592535665	999.9995
2	2	71.31537707620139	39.11392840843229	24.079480154194506	999.998
3	3	73.07605971281605	40.18062707041918	25.60358422176374	999.9955
4	4	69.57169325055334	41.81576779719287	23.54191732442338	999.992

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with data New interactive sheet

```
from sklearn.ensemble import IsolationForest

features = data[["temperature", "vibration", "pressure"]]

iso = IsolationForest(contamination=0.02)
data["anomaly"] = iso.fit_predict(features)

data["anomaly"] = data["anomaly"].map({1:0, -1:1})

print("Anomalies Detected:", data["anomaly"].sum())
```

```
Anomalies Detected: 200
```

```
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(features)

X = []
y = []

window = 20

for i in range(len(scaled_features)-window):
    X.append(scaled_features[i:i+window])
    y.append(data["RUL"].iloc[i+window])
```

```
X, y = np.array(X), np.array(y)

model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(window, 3)))
model.add(LSTM(50))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')

model.fit(X, y, epochs=5, batch_size=64)

model.save("lstm_rul_model.h5")
```

```
;
al/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argume
).__init__(**kwargs)
━━━━━━━━━━━━━━━━━━━━ 7s 26ms/step - loss: 74630.8281
;
━━━━━━━━━━━━━━━━━━━━ 4s 21ms/step - loss: 73121.0703
;
━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - loss: 73308.5078
;
━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 67510.9688
;
━━━━━━━━━━━━━━━━━━━━ 4s 27ms/step - loss: 67513.7109
absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is co
```

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.losses import MeanSquaredError

# Generate dummy dataset
np.random.seed(42)
n_samples = 3000
time = np.arange(n_samples)

temperature = 70 + 0.01*time + np.random.normal(0,2,n_samples)
vibration = 40 + 0.005*time + np.random.normal(0,1.5,n_samples)
pressure = 25 + 0.008*time + np.random.normal(0,1,n_samples)

RUL = 1000 - 0.05*time
RUL[RUL < 0] = 0

data = pd.DataFrame({
    "temperature": temperature,
    "vibration": vibration,
    "pressure": pressure,
    "RUL": RUL
})

# Scaling
scaler = MinMaxScaler()
features = scaler.fit_transform(data[["temperature","vibration","pressure"]])

X, y = [], []
window = 10

for i in range(len(features)-window):
    X.append(features[i:i+window])
    y.append(data["RUL"].iloc[i+window])

X, y = np.array(X), np.array(y)

# Build model
model = Sequential([
    LSTM(64, return_sequences=True, input_shape=(window,3)),
    LSTM(32),
    Dense(1)
])

# IMPORTANT FIX HERE 👇
model.compile(
    optimizer='adam',
    loss=MeanSquaredError()
)

model.fit(X, y, epochs=5, batch_size=32)

# SAVE IN NEW FORMAT (IMPORTANT)
```

```
model.save("lstm_rul_model.keras")

print("Model Saved Successfully!")
```

```
es/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Se

ep - loss: 854128.2500

ep - loss: 841175.6875

ep - loss: 832981.6875

ep - loss: 826517.5625

ep - loss: 820311.6250
```

```
!pip install pyngrok streamlit --quiet
```

```
from pyngrok import ngrok

ngrok.set_auth_token("3AFzGmFdTHc85ZlgpSRvKxGZmjd_Fq9pufZW4TAaYcDvJkb9")
```

```
%%writefile app.py
import streamlit as st
import numpy as np
from tensorflow.keras.models import load_model

st.title("AI-Based Predictive Maintenance for Launch Pad Systems")

model = load_model("lstm_rul_model.keras", compile=False)

temperature = st.slider("Temperature", 0, 200, 70)
vibration = st.slider("Vibration", 0, 100, 40)
pressure = st.slider("Pressure", 0, 100, 25)

if st.button("Predict"):

    input_data = np.array([[temperature, vibration, pressure]] * 10)
    input_data = input_data.reshape((1,10,3))

    predicted_rul = model.predict(input_data)[0][0]

    if predicted_rul < 200:
        st.error(f"⚠ Critical Warning! Estimated RUL: {predicted_rul:.2f} hours")
    else:
        st.success(f"Machine Operating Normally. Estimated RUL: {predicted_rul:.2f} hours")
```

```
Overwriting app.py
```

```
import subprocess

subprocess.Popen(["streamlit", "run", "app.py"])
```

```
<Popen: returncode: None args: ['streamlit', 'run', 'app.py']>
```

```
public_url = ngrok.connect(8501)
print("Public URL:", public_url)
```

```
Public URL: NgrokTunnel: "https://karlene-jarless-consuelo.ngrok-free.dev" -> "http://localhost:8501"
```

```
!ps aux | grep streamlit
```

```
root       75667 14.4  6.3 3647132 843044 ?      Sl   19:37   0:31 /usr/bin/python3 /usr/local/bin/streamlit run app.py
root       76645  0.0  0.0   7372  3476 ?        S    19:40   0:00 /bin/bash -c ps aux | grep streamlit
root       76647  0.0  0.0   6480  2544 ?        S    19:40   0:00 grep streamlit
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

predictions = model.predict(X)
mae = mean_absolute_error(y, predictions)
rmse = np.sqrt(mean_squared_error(y, predictions))

print("MAE:", mae)
print("RMSE:", rmse)
```

```
94/94 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step
MAE: 902.9485929253109
RMSE: 903.9793654816448
```