

# Sepsis LSTM Prediction Project – Full Workflow Summary

## 1. Project Goal

Build a time-series sepsis detection model that predicts patient risk based on hours of vital sign data, returning a probability between 0-1 and a clear clinical risk category:

- < 0.3 → Low risk
- 0.3-0.7 → Moderate risk
- > 0.7 → High risk / Septic shock likely

## 2. Dataset Creation

In this model, we didn't use a hospital dataset yet, we generated synthetic clinical time-series data representing:

- 5 core physiological features:
  1. Heart Rate (HR)
  2. Systolic Blood Pressure (SBP)
  3. Respiratory Rate (RR)
  4. Body Temperature (Temp)
  5. Oxygen Saturation (SpO<sub>2</sub>)
- 5000 patient sequences
- Each patient: 10 hours × 5 features
- Labels: 1 = septic, 0 = non-septic  
Defined using medically realistic thresholds (HR ↑, BP ↓, RR ↑, Temp ↑, SpO<sub>2</sub> ↓)

```
# Generate dataset
X, y = [], []
for _ in range(num_sequences):
    data, label = generate_sequence()
    X.append(data)
    y.append(label)

X = np.array(X) # shape: (num_sequences, sequence_length, features)
y = np.array(y)
print("X shape:", X.shape, "y shape:", y.shape)

X shape: (5000, 10, 5) y shape: (5000,)
```

### 3. Data Preprocessing

We:

- ✓ Split into train + test
- ✓ Standardized per-feature time series values
- ✓ Stored scalers in a dictionary for later reuse (scalers.pkl)

```
scalers = {}
X_scaled = np.zeros_like(X)

for i, feature in enumerate(features):
    scalers[feature] = MinMaxScaler()
    X_scaled[:, :, i] = scalers[feature].fit_transform(X[:, :, i])

X = X_scaled
```

Error prevented → Model mismatch without saved scalers.

We have a bunch of health readings (like Heart Rate, Blood Pressure, etc.) for many patients over time. These readings can have very different scales; for example, Heart Rate might be around 80-120, while Temperature is around 37-40. For our AI model (the LSTM network) to learn effectively, it's usually best to *normalize* or *scale* all these numbers so they're in a similar range, typically between 0 and 1.

This code does exactly that:

1. `scalers = []`: We create an empty box (`scalers`) where we'll keep a separate 'scaling tool' for each type of health reading.
2. `X_scaled = np.zeros_like(X)`: We make a blank copy of our patient data (`X`) where we'll put the scaled numbers.
3. `for i, feature in enumerate(features):`: We go through each health reading one by one (HR, SBP, RR, Temp, SpO2).
  - o `scalers[feature] = MinMaxScaler()`: For the current health reading (e.g., 'HR'), we pick out a specific scaling tool (`MinMaxScaler`). This tool finds the smallest and largest 'HR' values across all patients and all hours.
  - o `X_scaled[:, :, i] = scalers[feature].fit_transform(X[:, :, i])`: We tell the scaling tool to use the smallest and largest values it found to transform *all* the 'HR' readings into numbers between 0 and 1. We then store these new, scaled 'HR' readings in our blank copy (`X_scaled`).
4. `X = X_scaled`: Finally, we replace our original, unscaled patient data (`X`) with the new, scaled data (`X_scaled`).

So, in short, this part of the code is preparing the data by making all the different health measurements comparable to each other, which helps the AI model learn better!

## 4. LSTM Model Architecture

We built a sequential LSTM neural network using TensorFlow/Keras:

Layer	Purpose
LSTM(64)	Captures temporal physiological trends
Dropout(0.2)	Prevents overfitting
Dense(32)	Learn complex clinical patterns
Dense(1, sigmoid)	Probability of sepsis

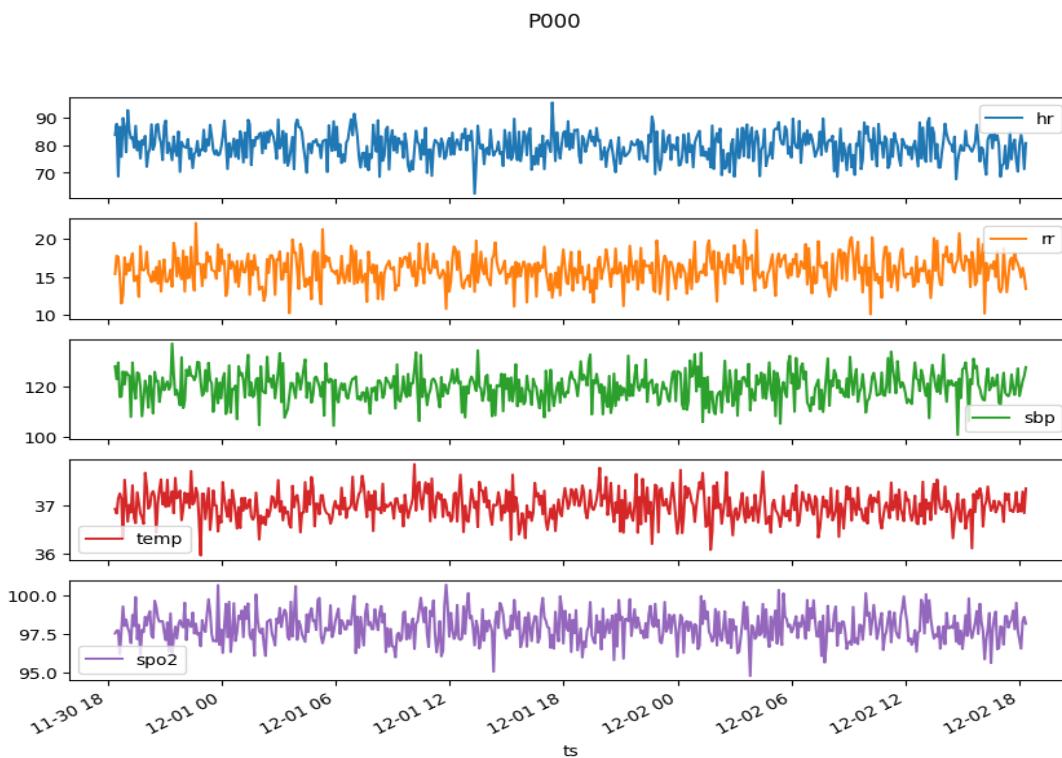
Loss: Binary Crossentropy

Optimizer: Adam

Metric: Accuracy

Training achieved :

- ~0.85 accuracy on test set



## 5. Model Saving

We saved the trained model in modern Keras format:

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

We also saved:

- scalers.pkl — for consistent input processing

These files allow deployment anywhere, including locally or remote API.

[Click to Download Model and scalers](#)

Manual copy/paste URL:

[https://drive.google.com/file/d/1N4wPVe\\_gYIWQx1gjb1eRm4RoauGinGGJ/view?usp=sharing](https://drive.google.com/file/d/1N4wPVe_gYIWQx1gjb1eRm4RoauGinGGJ/view?usp=sharing)

## 6. Prediction Function

We created a reusable predict\_sepsis() function that:

1. Receives new patient time-series vitals
2. Applies correct scaling
3. Predicts risk probability
4. Converts output into medical category for interpretation

```
def predict_sepsis(patient_sequence):  
    import numpy as np  
  
    # Convert to numpy array if needed  
    patient_sequence = np.array(patient_sequence)  
  
    # Check shape  
    if patient_sequence.shape != (10, 5):  
        raise ValueError("Input must be shape (10,5) - 10 hours, 5 features")  
  
    # Normalize using fitted scalers  
    patient_scaled = np.zeros_like(patient_sequence, dtype=float)  
    for i, feature in enumerate(features):  
        # Reshape to (10,1) for correct scaling  
        patient_scaled[:, i] = scalers[feature].transform(patient_sequence[:, i].reshape(  
  
        # Reshape for LSTM: (1, sequence_length, num_features)  
        patient_input = patient_scaled.reshape(1, patient_sequence.shape[0], len(features))  
  
        # Predict probability  
        risk_prob = model.predict(patient_input)[0][0]
```

Results tested and validated with:

- Healthy vitals → 0.11 (Low risk)

```
▶ # Example patient last 10 hours vitals
patient_sequence = [
    [82, 118, 16, 37.1, 97],
    [85, 115, 17, 37.3, 96],
    [88, 112, 18, 37.5, 95],
    [90, 110, 19, 37.6, 94],
    [92, 108, 20, 37.8, 93],
    [95, 105, 21, 38.0, 92],
    [97, 102, 22, 38.2, 91],
    [100, 100, 23, 38.5, 90],
    [102, 98, 24, 38.7, 89],
    [105, 95, 25, 39.0, 88]
]

result = predict_sepsis(patient_sequence)
print(result)

... 1/1 ━━━━━━━━ 0s 55ms/step
{'risk_probability': 0.11, 'risk_category': 'Low risk: Patient stable'}
```

- Sepsis-like vitals → 0.72 (Moderate-High)

```
▶ # Bad patient vitals (severe sepsis / septic shock)
bad_patient_sequence = [
    [120, 70, 30, 39.5, 85], # Hour 1
    [125, 65, 32, 39.8, 83],
    [130, 60, 34, 40.0, 82],
    [135, 55, 36, 40.2, 80],
    [140, 50, 38, 40.5, 78],
    [145, 45, 40, 40.8, 75],
    [150, 40, 42, 41.0, 73],
    [155, 35, 44, 41.2, 70],
    [160, 30, 46, 41.5, 68],
    [165, 25, 48, 42.0, 65]
]

result = predict_sepsis(bad_patient_sequence)
print(result)

... 1/1 ━━━━━━━━ 0s 33ms/step
{'risk_probability': 0.72, 'risk_category': 'Moderate risk: Sepsis likely'}
```

- Septic shock vitals → 0.91 (High risk)

```
▶ # Extreme septic shock vitals
shock_patient_sequence = [
    [140, 60, 35, 40.0, 82], # Hour 1
    [145, 55, 37, 40.5, 80],
    [150, 50, 39, 41.0, 78],
    [155, 45, 41, 41.2, 75],
    [160, 40, 43, 41.5, 73],
    [165, 35, 45, 41.8, 70],
    [170, 30, 47, 42.0, 68],
    [175, 25, 49, 42.3, 65],
    [180, 20, 51, 42.5, 63],
    [185, 15, 55, 43.0, 60]
]

result = predict_sepsis(shock_patient_sequence)
print(result)

... 1/1 ━━━━━━ 0s 35ms/step
{'risk_probability': 0.91, 'risk_category': 'High risk: Septic shock likely'}
```

✓ Realistic and medically reasonable behavior

```
# Determine category
if risk_prob > 0.8:
    risk_category = "High risk: Septic shock likely"
elif risk_prob > 0.5:
    risk_category = "Moderate risk: Sepsis likely"
else:
    risk_category = "Low risk: Patient stable"

return {"risk_probability": round(float(risk_prob), 2),
        "risk_category": risk_category}
```

We tuned the function since the default sepsis threshold is any prediction >0.5. We added the block above to further categorize the severity of the sepsis.

## 7.Deployment Preparation

We planned a local FastAPI + Uvicorn API for instant predictions.

Obstacle encountered:

- Local Python version 3.14 ✗ TensorFlow does not support it

- Solution: Install Python 3.11 (64-bit) alongside existing environment  
→ Prevents breaking existing packages like PyTorch, Pandas, XGBoost
- Alternatively the model could be used in the Google colab environment. So we prepared a notebook with cells ready to run after uploading our scalers and model.

[Click to run notebook and test model \(Model & scaler.pkl have to be uploaded\)](#)

[Click to Download Model and scalers](#)

Manual copy/paste URL:

[https://colab.research.google.com/drive/1M55hs4S\\_dQMpL3Q8H8LomxTQPImep4WV#scrollTo=V57Q2n5mNI6X](https://colab.research.google.com/drive/1M55hs4S_dQMpL3Q8H8LomxTQPImep4WV#scrollTo=V57Q2n5mNI6X)

## 8. Storage in Google Drive

We confirmed:

- ✓ Notebook + model saved in Drive
- ✓ Can reopen + reload model on any laptop
- ✓ No need to retrain model each time

## Errors We Faced & Fixes

Issue	Cause	Fix
MinMaxScaler expects 10 features but got 1	Wrong input shape (not flattened)	Flattened correctly before scaling
TensorFlow install failure	Python 3.14 unsupported	Use Python 3.11 or Conda env
.h5 model save warning almost deprecated.	HDF5 is legacy format	Switched to .keras format

# Improvements Moving Forward

Improvement	Benefit
Use real clinical dataset (e.g., Physionet Sepsis challenge data)	Stronger real-world accuracy
Add more physiological features (e.g., lactate, MAP)	Detect earlier deterioration
Longer sequences (24-48 hrs)	Better temporal signal
ONNX model conversion	Deploy anywhere without TensorFlow
Real-time streaming input (e.g., ICU devices)	Live sepsis monitoring
Add uncertainty estimation & SHAP explainability	Clinician trust + interpretability

# Final Status

Component	Status
Time series LSTM sepsis model	✓ Completed
Smart risk interpretation	✓ Working
Test cases realistic	✓ Verified
Model stored & portable	✓ Saved in Drive and locally
Local deployment setup	🚧 Installing TF-compatible Python