Sepsis Timeline Project

Tidig prediktion av sepsis med kliniska tidsserier

Kursrapport

Deep Learning - metoder och tillämpningar (HT25)

Student:	**Salar Sadek (820221)**
Datum:	18 januari 2026
Dataset:	PhysioNet/CinC Challenge 2019 (training_setA)
Delmängd:	dev5000 (5 000 patienter)
Miljö:	TensorFlow/Keras + notebooks test

Projektmapp: Google Drive Projektmapp

Innehåll

Figurer

Sammanfattning

Sepsis är ett livshotande tillstånd där tidig upptäckt kan påverka både mortalitet och resursanvändning i vården. I denna rapport studerar jag tidig prediktion av sepsis utifrån multivariata kliniska tidsserier baserade på PhysioNet/CinC Challenge 2019 [5]. Arbetet är upplagt som ett praktiskt data science-projekt (jag är data science student för närvarande) med fokus på en reproducerbar pipeline: explorativ dataanalys (EDA), hantering av saknade värden, standardisering, fönsterindelning samt modellträning och utvärdering. Jag jämför fyra sekvensmodeller på delmängden dev5000: LSTM (baseline), LSTM (förbättrad), GRU (baseline) och Transformer (baseline).

Eftersom sepsis är en sällsynt klass prioriteras PR-AUC i utvärderingen, kompletterat med ROC-AUC och tröskelbaserade mått (precision, recall och F1) samt confusion matrices. Resultaten visar att Transformer når hög validerings-PR-AUC men tappar på test, vilket tyder på ett tydligt generaliseringsgap. I denna iteration framstår LSTM baseline som den mest robusta modellen när man väger samman test PR-AUC och F1 vid val-optimerad tröskel. GRU har däremot högre recall (fångar fler positiva fall) men producerar fler falska larm. Rapporten diskuterar varför tröskelval är en central designfråga i kliniska uppgifter (missade fall kontra alarm fatigue) och varför preprocessing (imputering + mask-kanaler) ofta är avgörande för att modeller ska lära sig rimliga mönster i data med mycket missingness. Slutligen presenteras konkreta förbättringar för en nästa iteration, bland annat mer systematisk hyperparametertuning, bättre hantering av oregelbunden sampling, sannolikhetskalibrering och en mer kliniskt driven utvärdering.

I tidigare studier används ofta klassiska ML-modeller (t.ex. logistisk regression, Random Forest, XGBoost) där tidsserier först sammanfattas med handgjorda features. De är ofta mer tolkningsbara men kan tappa detaljer i tidsordningen. Det finns också mer specialiserade tidsseriemodeller (t.ex. TCN och time-aware RNN:er) som hanterar oregelbunden provtagning bättre. Våra LSTM/GRU och transformer-modell är mer “standard” och försöker istället lära mönster direkt från sekvensdata efter preprocessing.

1 Introduktion

Sepsis är ett akut och potentiellt livshotande tillstånd som kan uppstå vid infektion och leda till organsvikt om behandling fördröjs. I intensivvård och akutsjukvård samlas stora mängder patientdata in kontinuerligt: vitalparametrar, laboratorievärden och kliniska observationer över tid. Detta gör sepsis till en relevant uppgift för maskininlärning på tidsserier, där målet är att upptäcka riskmönster innan tillståndet blir tydligt för en mänsklig observatör eller innan kliniska kriterier uppfylls. I PhysioNet/CinC Challenge 2019 formuleras uppgiften som tidig varning och bedöms med en nyttobaserad princip där tidig upptäckt belönas och sena eller felaktiga larm straffas [5].

Samtidigt är kliniska tidsserier svåra jämfört med renare ML-dataset. För det första är mätningar oregelbundna och påverkas av kliniska rutiner (t.ex. vissa labb tas bara vid misstanke eller i glesa intervall). För det andra är saknade värden mycket vanliga och kan vara strukturerade; frånvaro av mätningar kan spegla kliniska beslut och riskerar att bli en oönskad genvägssignal om man inte hanterar det tydligt. För det tredje är sepsis relativt ovanligt, vilket skapar kraftig klassobalans och gör att traditionella mått som accuracy kan bli missvisande. Det är en utmatning för hur modellerna ska utvärderas på ett rättvist sätt.

Slutligen kräver uppgiften att man undviker dataläckage: om fönster från samma patient hamnar i både träning och test kan resultaten bli orealistiskt optimistiska.

1.1 Syfte

Syftet med denna rapport är att bygga en tydlig och reproducerbar pipeline och jämföra modellfamiljer för tidig sepsisprediktion på `dev5000`. Med en välfungerande pipeline kan man i framtiden använda större datamängder och genomgöra utvärderingar på ett standardiserat och systematiskt sätt. Målet är inte bara att redovisa siffror, utan också att resonera om varför resultaten ser ut som de gör: hur preprocessing påverkar modellernas inlärning, hur metrikval förändrar tolkningen, och hur tröskelval leder till olika feltyper. I en klinisk kontext är det särskilt viktigt att förstå tradeoff mellan att missa fall (false negatives) och att skapa många falska positiva (false positives) som kan leda till onödiga åtgärder och alarm fatigue.

1.2 Varför PR-AUC?

Vid klassobalans kan en modell få hög accuracy genom att nästan alltid prediktera ingen sepsis. ROC-AUC kan ibland se relativt bra ut även när precisionen för positiva fall är låg. PR-AUC (Average Precision) fokuserar på precision/recall för den positiva klassen och är därför mer informativ när prevalensen är låg och när det praktiska målet är att hitta sällsynta fall utan att drunkna i falsklarm.

2 Data och metodöversikt

Datasetet kommer från PhysioNet/CinC Challenge 2019 och består av patientvisa timserier där varje rad representerar en timme och kolumnerna är kliniska variabler. De är många variabler och för mig som saknar medicinsk bakgrund är de svåra att tolka. Varje patientfil innehåller vitalparametrar, laboratorievärden och demografiska variabler samt en label som anger om patienten befinner sig i ett tidigt sepsis-fönster. I praktiken betyder detta att modellen behöver lära sig mönster över tid (t.ex. gradvisa förändringar i puls, blodtryck eller labbvärden) snarare än att bara titta på ett enskilt mättillfälle. En viktig sak här är att datan inte är perfekt: vissa variabler mäts sällan, vissa patienter har korta serier och andra långa, och missingness är en stor del av problemet. När man tittar in i datafilerna ser man att många värden saknas och en välgenomarbetad datarengöring behövs. För att göra datan lite mer begriplig för sammanfattas centrala variabler i Tabell 1, och projektets struktur finns i Appendix A.

2.1 Viktiga variabler i patientfiler (sammanfattning)

Tabell 1: Översikt av centrala kolumner i patientfilerna (PhysioNet/CinC 2019). Tabellen är en komprimerad sammanfattning av variabler som typiskt används i modellerna: vitalparametrar, laboratorievärden och demografi, samt hur labeln definieras i relation till sepsistidpunkt.

Grupp	**Variabel**	**Beskrivning (enhet)**
Vitalparametrar	HR	Hjärtfrekvens (slag/min)
	O2Sat	Pulsoximetri (%)
	Temp	Temperatur (°C)
	SBP	Systoliskt blodtryck (mmHg)
	MAP	Medelartärtryck (mmHg)
	DBP	Diastoliskt blodtryck (mmHg)
	Resp	Andningsfrekvens (andetag/min)
	EtCO2	End-tidal CO_2 (mmHg)
Lab (exempel)	Lactate	Laktat (mg/dL)
	Creatinine	Kreatinin (mg/dL)
	WBC	Leukocyter ($10^3/\mu L$)
	Platelets	Trombocyter ($10^3/\mu L$)
	Bilirubin_total	Bilirubin total (mg/dL)
	BUN	Urea-nitrogen (mg/dL)
	pH	Arteriellt pH (N/A)
	PaCO2	Arteriellt CO_2-tryck (mmHg)
	SaO2	Arteriell syremättnad (%)
	Glucose	Serumglukos (mg/dL)
Demografi	Age	Ålder (år; 100 för ≥ 90)
	Gender	Kön (0=Kvinna, 1=Man)
	Unit1/Unit2	Administrativ ICU-enhet (MICU/SICU-indikatorer)
	HospAdmTime	Timmar mellan sjukhusintag och ICU-intag
	ICULOS	ICU-vårdtid (timmar sedan ICU-intag)
Utfall	SepsisLabel	1 om $t \geq t_{sepsis} - 6$, annars 0 (icke-sepsis alltid 0)

2.2 Dataset och split

I `dev5000` använder jag 5 000 patienter för att skapa en stabil experimentmiljö som ändå är hanterbar inom ett kursprojekt. Data delas patientvis för att undvika läckage: train 3 500, val 750, test 750. Det här är viktigt eftersom pipeline skapar många fönster (samples) per patient; om patientdata blandas mellan splitar riskerar modellen att indirekt känna igen individmönster och

därmed överskatta generalisering.

2.3 Kontroller mot dataläckage

Jag gjorde flera konkreta kontroller för att minska risken för dataläckage, eftersom detta annars kan ge orealistiskt bra resultat. För det första görs splitten på patientnivå, vilket innebär att alla fönster från en patient alltid hamnar i samma split (train/val/test). Det gör att modellen inte kan känna igen en patient genom mönster som råkar återkomma i flera fönster.

För det andra beräknas all preprocessing-statistik (medianer för imputering, samt mean/std för standardisering) enbart på träningssplitten och återanvänds sedan oförändrad på val/test. Det är en typisk källa till läckage om man råkar normalisera på hela datasetet. För det tredje är labeln konstruerad som en framtidslabel med horisont $H = 6$ (se nästa avsnitt), så att input-fönstret bara innehåller historik. Jag undviker alltså att råka inkludera framtida timsteg i samma fönster som den etikett jag vill prediktera.

2.4 Måldefinition: framtidslabel med horisont H

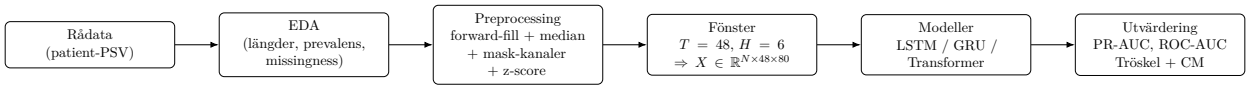
För att efterlikna tidig varning definierar jag en framtidslabel där modellen ska prediktera sepsis inom en given horisont. Jag använder $H = 6$ timmar, i linje med formuleringen i challenge-konteksten [5]. Varje input består av historik $T = 48$ timmar. En förenklad beskrivning av målet är:

$$y(t) = \mathbb{1}\{\text{sepsis inträffar i } (t, t + H]\}.$$

I praktiken skapas labeln i preprocessing-steget när fönster byggs, så att varje fönster får en binär etikett kopplad till om sepsis ligger i framtiden.

2.5 Metodfigur (pipeline)

Arbetsflödet i projektet sammanfattas i Figur 1. Målet med figuren är att visa hela kedjan från rådata till jämförbara resultat, så att läsaren förstår vad som händer mellan PSV-filer och PR-AUC.



Figur 1: Översikt av arbetsflödet från rådata till utvärderade modeller. Figuren visar hur råa patientserier blir modellinput via EDA och preprocessing, samt hur modellerna jämförs med samma mått.

2.6 Experimentinställningar

I detta projekt väljs parametrar för att balansera stabil träning, rimlig beräkningskostnad och tydlig tolkning av resultat. Fönsterlängden $T = 48$ ger modellerna tillräcklig historik för att fånga både snabba och gradvisa trender, men utan att sekvenserna blir så långa att träningen blir instabil eller onödigt tung. Horisonten $H = 6$ gör prediktionen praktiskt relevant som tidig varning, och är också nära hur uppgiften beskrivs i datasetet [5]. Jag tränar med Adam (learning rate 10^{-3}), vilket är en standardinställning som brukar fungera bra för den här typen av nätverk. Batch size 128 är en kompromiss mellan stabil gradient och minneskrav. Max 30 epoker kombineras med early stopping (patience 6, återställ bästa vikter) för att minska överanpassning. Eftersom sepsis är ovanligt används class weights så att den positiva klassen får större vikt; detta hjälper mot en trivial modell som mest gissar negativt. Jag väljer tröskel genom att maximera F1 på val, och

rapporterar därefter hur samma tröskel fungerar på test (för att minska risken att man optimerar på test).

Tabell 2: Centrala experimentval (sammanfattning).

Del	**Val**
dev5000	5000 patienter
Split (patientvis)	3500 / 750 / 750
Råfeatures	40
Representation	värden + mask $\Rightarrow$ 80 kanaler
Fönster	$T = 48$ timmar
Horisont	$H = 6$ timmar
Imputering	forward-fill per patient + median (train) fallback
Standardisering	z-score med train mean/std
Optimizer	Adam, lr = 1e-3
Batch size	128
Max epochs	30
Early stopping	monitor val_pr_auc, patience=6, restore best
Tröskelval	välj thr som maximerar F1 på val

3 Preprocessing och feature-konstruktion

Preprocessing-steget är avgörande för att modellerna ska fungera stabilt på kliniska tidsserier. De största problemen i rådata är (1) saknade värden, (2) olika skalor mellan variabler och (3) varierande längder på patientserier. I mitt projekt löser jag detta genom en pipeline som först gör imputering, sedan standardisering, och därefter bygger fönster med fast längd.

Imputering: Jag använder en praktisk strategi som fungerar bra i tidsserier: först forward-fill per patient (där det går), och om det fortfarande saknas värden används medianvärden beräknade på träningsdata. Det här är inte perfekt, men det är ett rimligt första steg och är dessutom lätt att reproducera. En viktig detalj är att median/mean/std beräknas på train split för att undvika dataläckage.

Mask-kanaler: För att modellen inte ska blanda ihop imputerade värden med verkliga observationer duplicerar jag featurekanalerna genom att lägga till en mask-indikator. Det betyder att input kan beskrivas som 80 kanaler: 40 normaliserade värden + 40 indikatorer som signalerar om värdet var observerat eller imputerat. Den här representationen hjälper modellen att hantera missingness mer robust, eftersom den kan lära sig att vissa mönster i saknad data inte nödvändigtvis betyder klinisk förändring.

Standardisering: Efter imputering standardiseras varje variabel med z-score (mean och std från train). Det gör att modellerna inte domineras av features med stora numeriska skalor, och ger ofta mer stabil inlärning.

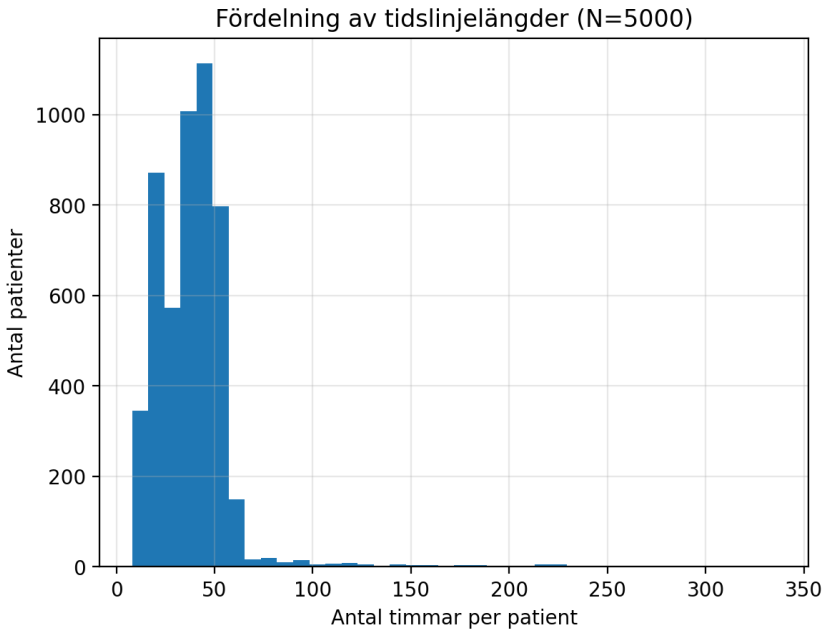
Fönster: Slutligen skapas fönster med $T = 48$ timmar, och labeln bygger på $H = 6$ timmars prediktionshorisont. Det gör att alla modeller får input med samma dimension 48×80 och att jämförelsen mellan modeller blir mer rättvis.

4 Exploratory Data Analysis (EDA)

EDA används för att förstå datans struktur och risker innan modellering: variation i sekvenslängd, prevalens och omfattning av saknade värden. Jag genomförde EDA i flera steg: (1) kartläggning av tidsserielängder per patient, (2) analys av klassfördelning och prevalens, (3) missingness-analys både aggregat (per variabel) och temporalt (per patient över tid), samt (4) visuella sanity-checks på enskilda patienter för att förstå brus, trender och mätmönster.

Ett centralt fynd är att tidsserielängderna varierar kraftigt mellan patienter, vilket motiverar fönsterindelning med fast $T = 48$. Prevalensanalysen visar tydlig klassobalans, vilket förklarar valet av PR-AUC och användningen av class weights. Missingness-figurerna visar att saknade värden är strukturerade: vissa variabler är ofta saknade och saknad data kan uppträda i block över tid. Detta är precis den typ av problem som kan lura modeller om man bara fyller NaN utan att tala om att det var NaN från början. Som en sista EDA-del tittade jag också på exempelpatienter för att få en känsla för data och se att värden ligger i rimliga intervall.

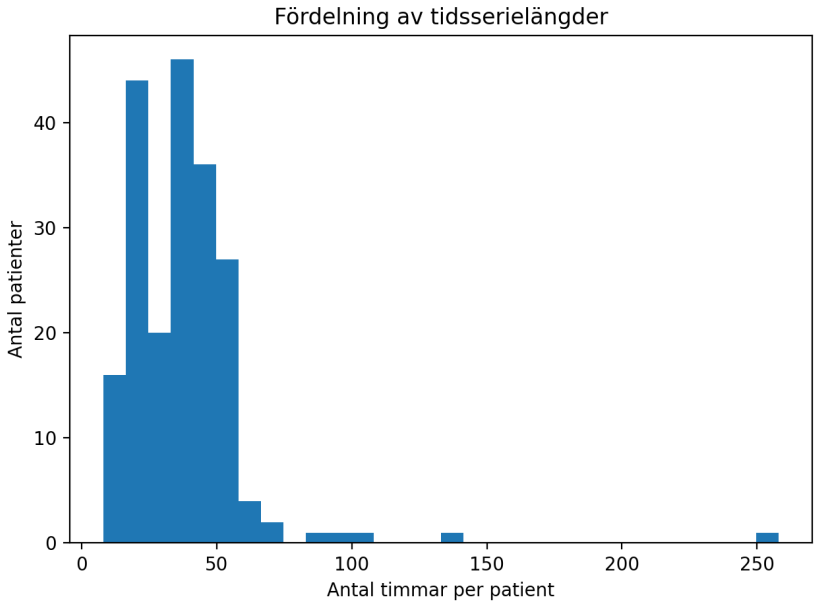
I Figur 2 visas hur längden på patientserierna varierar, och Figur 3 ger en alternativ vy av samma information. Tillsammans motiverar de att jag använder fasta fönster istället för att träna på helt olika sekvenslängder.



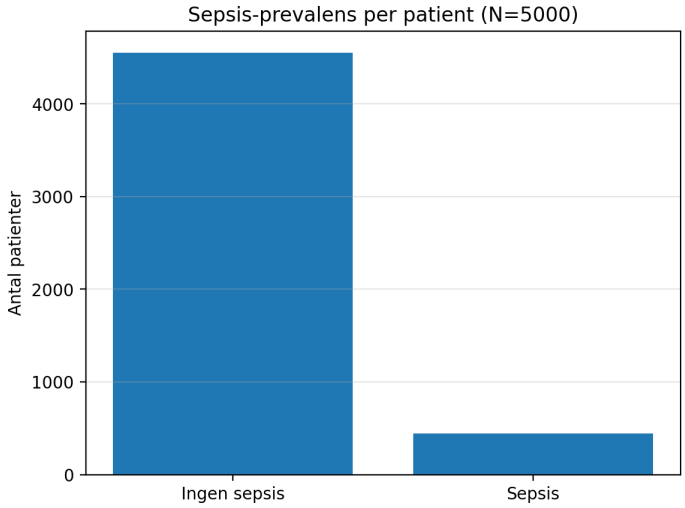
Figur 2: Fördelning av tidsserielängder (antal timmar) per patient i dev5000. Figuren visar att datan innehåller både korta och långa serier, vilket påverkar hur modellinput bör konstrueras.

Prevalensen i Figur 4 visar tydligt att sepsis är sällsynt, vilket i sin tur motiverar PR-AUC som huvudmått. Exempelpatienten i Figur 5 fungerar som sanity-check och ger en mer intuitiv känsla för variation över tid.

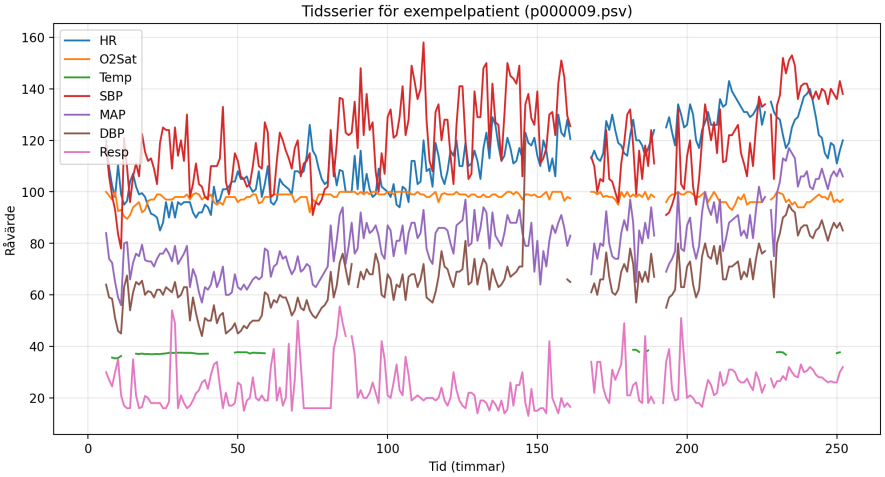
I Figur 6 ser man vilka variabler som har mest saknade värden, och Figur 7 visar att missingness ofta kommer i block, vilket betyder att saknad data inte är slumpmässig.



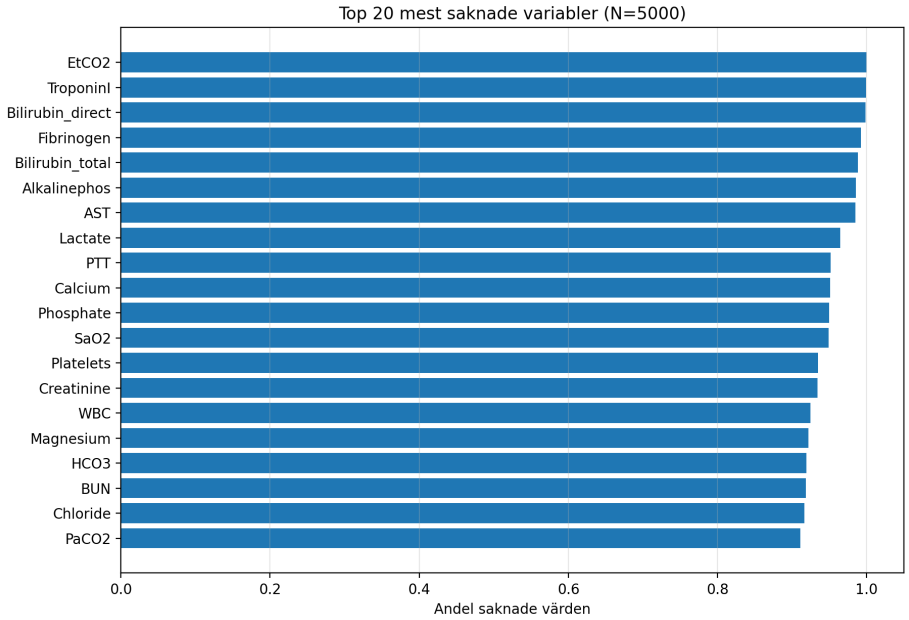
Figur 3: Histogram över tidsserielängder (alternativ visualisering). Denna figur gör det enklare att se var majoriteten av patienterna hamnar och hur svansen av mycket långa serier ser ut.



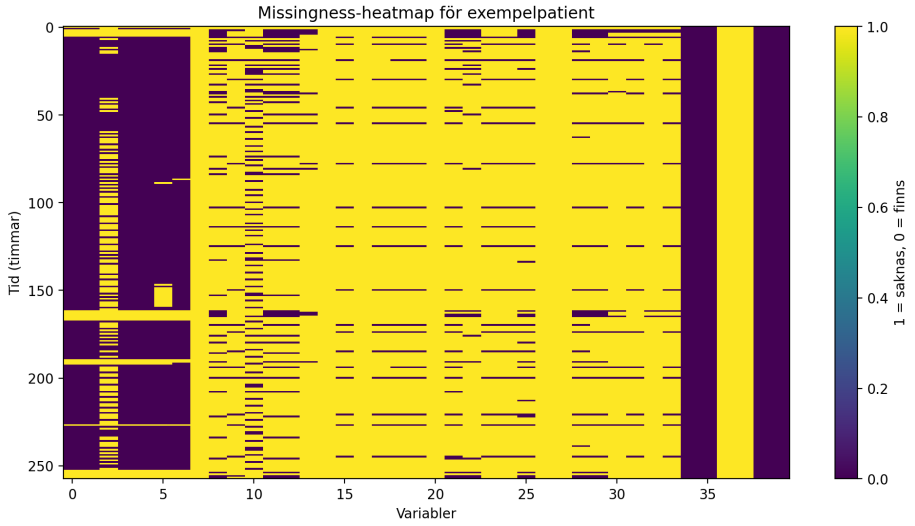
Figur 4: Sepsis-prevalens i dev5000 (andel positiva fall). Figuren illustrerar den tydliga klassobalansen och varför precision/recall-baserade mått blir viktiga.



Figur 5: Exempel på en patient-tidsserie (visualiserad från EDA). Figuren visar variation över tid och ger en intuitiv bild av att signalerna kan vara brusiga samt att vissa mätningar kan saknas eller vara glesa.



Figur 6: Topplista över variabler med mest missing data. Figuren visar att många labbvärden är ofta saknade, vilket är rimligt kliniskt men utmanande för modellering.



Figur 7: Heatmap över missingness för en exempelpatient (saknad data över tid). Figuren visar att saknad data ofta uppstår i sammanhängande block, vilket kan lura modeller om man inte använder mask-kanaler eller annan tydlig hantering.

5 Modeller

Jag jämför fyra modeller som alla är rimliga kandidater för kliniska tidsserier, men som har olika styrkor. Tanken är att få både en klassisk robust baslinje (LSTM/GRU) och en mer modern modell (Transformer) som kan fungera bra om den får rätt inställningar. Jag valde dessa modeller av tre skäl: (1) de är vanliga i kursen och i deep learning-litteratur för sekvensdata, (2) de representerar olika sätt att modellera tidsberoenden, och (3) de gör att jag kan resonera om generalisering och överanpassning i ett realistiskt kliniskt problem.

LSTM (Long Short-Term Memory) är en välkänd RNN-variant med gating-mekanismer som hjälper till att hantera långsiktiga beroenden. I kliniska data kan ett sepsisförlopp byggas upp över många timmar, och LSTM är ofta en bra startpunkt. Jag kör en baseline-LSTM som är relativt enkel: en LSTM-lager som sammanfattar sekvensen och ett litet dense-head. Sedan kör jag en förbättrad LSTM med två staplade LSTM-lager (128 och 64 units), batch normalization, och en kombination av global max/average pooling över tid. Syftet med den förbättrade modellen är att se om mer kapacitet och bättre sekvenssammanfattning kan ge högre PR-AUC utan att tappa generalisering.

GRU (Gated Recurrent Unit) är en förenklad variant av LSTM som ofta tränar snabbare och ibland presterar lika bra. GRU har färre parametrar och kan vara mer stabil i vissa inställningar. Här använder jag en GRU-baslinje med 128 units och samma typ av dense-head som LSTM baseline.

Transformer bygger på self-attention och kan i teorin fånga relationer mellan tidpunkter mer direkt än RNN. Det är attraktivt för längre beroenden, men Transformers är också tuning-känsliga och kan överanpassa om man inte regulariserar bra. Min baseline-Transformer följer en enkel encoder-struktur: projicering till $d_{model} = 64$, positionella embeddingar, 2 encoder-block (4 heads, feed-forward 128), pooling och ett dense-head.

5.1 Översiktstabell: modellstruktur (hög nivå)

I Tabell 3 sammanfattas modellerna på en hög nivå för att det ska bli lättare att följa resultatdelen modell-för-modell.

Tabell 3: Översikt av modeller och deras strukturella idé (baserat på notebooks).

Modell	Arkitektur (kort)	Kommentar
LSTM (baseline)	LSTM(128, dropout=0.2) → Dense(64, ReLU) → Dropout(0.2) → Sigmoid	Stabil baslinje med relativt få lager, bra som referens.
LSTM (förbättrad)	Masking → LSTM(128, seq) + BN → LSTM(64, seq) + BN → GMP+GAP → Dense(128) → Dense(64) → Sigmoid	Mer kapacitet + pooling över tid; kan ge bättre signal men riskerar överanpassning.
GRU (baseline)	GRU(128, dropout=0.2) → Dense(64, ReLU) → Dropout(0.2) → Sigmoid	Snabbare celltyp, ofta bra balans mellan prestanda och komplexitet.
Transformer (baseline)	Dense proj → (positional emb) → 2 encoder-block (4 heads, $d_{model} = 64$) → pooling → Dense(128) → Sigmoid	Hög valprestanda men tydlig risk för generaliseringsgap i denna iteration.

6 Resultat

Resultatdelen är uppdelad modell-för-modell för att göra det tydligt vad varje arkitektur bidrar med, hur den tränar och var den lyckas/misslyckas på val respektive test. Jag börjar med en sammanfattande tabell och går sedan igenom varje modell med tillhörande kurvor och en kort tolkning. När jag tolkar resultaten försöker jag hålla två saker i huvudet samtidigt: (1) PR-AUC och precision/recall är viktigast eftersom sepsis är sällsynt, och (2) skillnaden mellan val och test säger mycket om generalisering.

6.1 Sammanfattning: AUC och tröskelmått

Tabellerna i Tabell 4 och Tabell 5 sammanfattar modellprestanda på dev5000. Värdena är inmatade direkt från de exporterade CSV filerna som notebooken genererar och sparar ner. I LaTeX (baserat på exporterade resultat), fylls värdena direkt i tabellerna vilket gör att tabellerna alltid visas korrekta uppdaterade värden ifall man kör om analyserna i framtiden. Detta är ett strukturerat och standardiserat arbetssätt som ofta rekommenderas inom data science projekt.

Tabell 4: Sammanfattning av modellprestanda på dev5000 (AUC-mått).

Modell	Val ROC-AUC	Val PR-AUC	Test ROC-AUC	Test PR-AUC
LSTM (baseline)	0.664	0.216	0.563	0.160
LSTM (förbättrad)	0.623	0.241	0.580	0.156
GRU (baseline)	0.648	0.170	0.539	0.159
Transformer (baseline)	0.695	0.261	0.549	0.137

Tabell 5: Testprestanda vid tröskeln som maximerar F1 på val. Tabellen visar både precision/recall/F1 och felprofilen (TP/FP/FN) på test.

Modell	Thr (val)	Prec (test)	Recall (test)	F1 (test)	TP	FP	FN
LSTM (baseline)	0.45	0.151	0.509	0.233	190	1066	183
LSTM (förbättrad)	0.55	0.149	0.233	0.182	87	498	286
GRU (baseline)	0.50	0.136	0.566	0.219	211	1346	162
Transformer (baseline)	0.80	0.152	0.298	0.201	111	618	262

Tolkning av tabellerna: På val ligger Transformer högst i PR-AUC, men tappar tydligt på test (lägre test PR-AUC), vilket stödjer bilden av ett generaliseringsgap. LSTM baseline har den mest stabila helhetsbilden i denna iteration (rimlig test PR-AUC och högst test-F1 vid val-optimerad

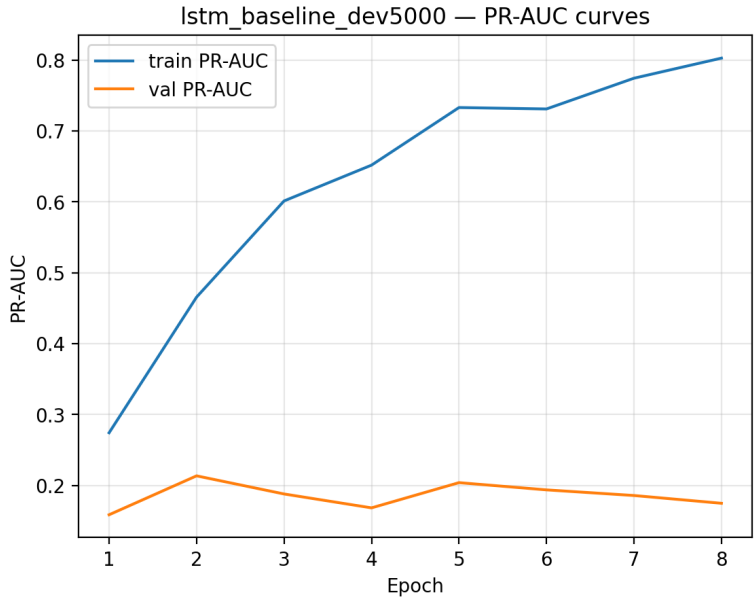
tröskel). GRU baseline ger högst recall på test, men också flest falska positiva (FP), vilket är en klassisk tradeoff i obalanserade problem.

6.2 En tydlig winner-berättelse (med reservationer)

Om jag måste välja en vinnare i denna iteration så blir det LSTM baseline. Motiveringen är att den ger stabil testnivå och en bra kombination av PR-AUC och F1 vid val-optimerad tröskel, vilket tyder på rimlig generalisering. Samtidigt måste detta tolkas praktiskt: modellen får upp recall genom att sänka tröskeln, vilket också ökar antalet falska positiva. GRU fångar fler positiva fall (hög recall) men med ännu fler falsklarm. Transformer ser stark ut på val men tappar tydligt på test, vilket gör den mindre pålitlig i nuvarande baseline-form.

6.3 LSTM baseline

I Figur 8 ser man hur PR-AUC utvecklas under träning för LSTM baseline, och det ger en känsla för om modellen stabiliseras eller börjar överanpassa. På test visas ROC/PR i Figur 9 och Figur 10, där PR-kurvan är mest relevant för den sällsynta positiva klassen. Tröskelkurvorna i Figur 11 visar att val av tröskel är en designfråga snarare än något man bara sätter till 0.5. Slutligen visar confusion matrix i Figur 12 exakt vilka feltyper modellen gör.



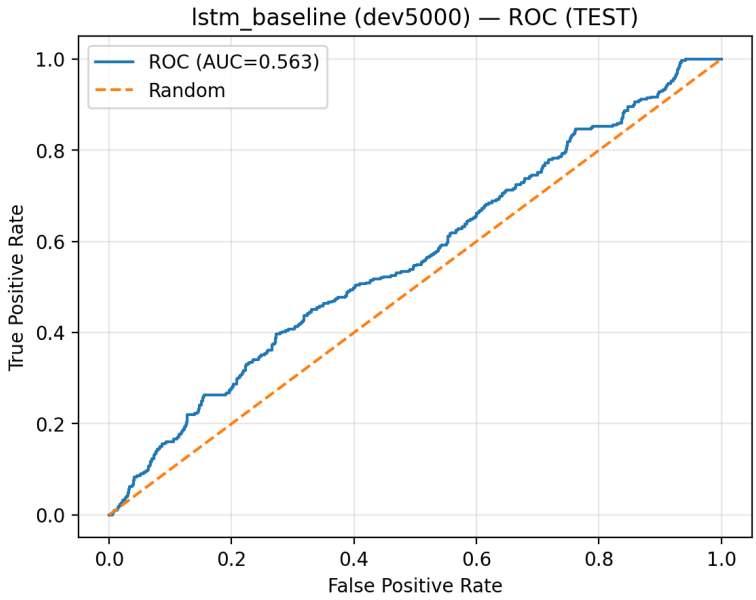
Figur 8: Learning curve för PR-AUC (LSTM baseline). Figuren visar hur PR-AUC utvecklas över epoker och ger en bild av när modellen slutar förbättras på val.

6.4 LSTM förbättrad

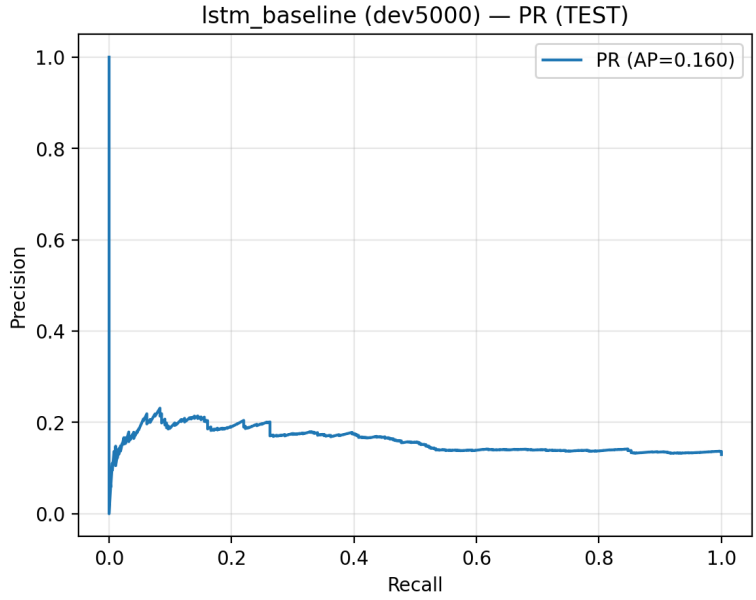
Den förbättrade LSTM-varianten illustrerar att mer kapacitet inte automatiskt betyder bättre testbeteende. I Figur 13 visas träningsutveckling i PR-AUC, och testkurvorna syns i Figur 14 och Figur 15. Confusion matrix i Figur 16 hjälper också att jämföra felprofil mot baseline.

6.5 GRU baseline

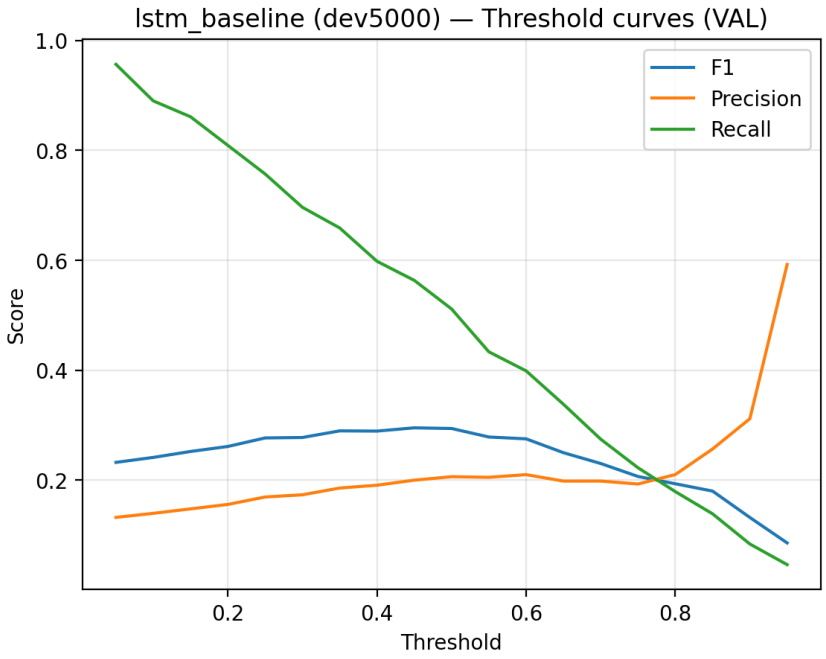
GRU baseline har hög recall och är därmed mer larmvillig. I Figur 17 syns PR-AUC-utvecklingen under träning, och testkurvorna syns i Figur 18 och Figur 19. Detta gör det möjligt att tolka



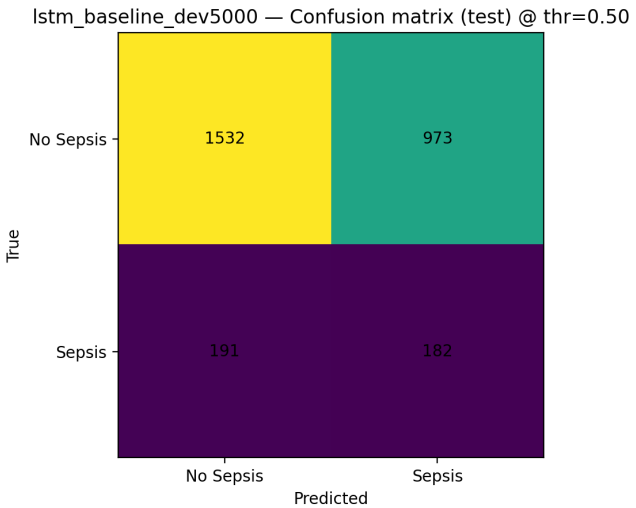
Figur 9: ROC-kurva på test (LSTM baseline). ROC-AUC visar separationsförmåga över trösklar men kan vara mer optimistisk än PR-AUC vid låg prevalens.



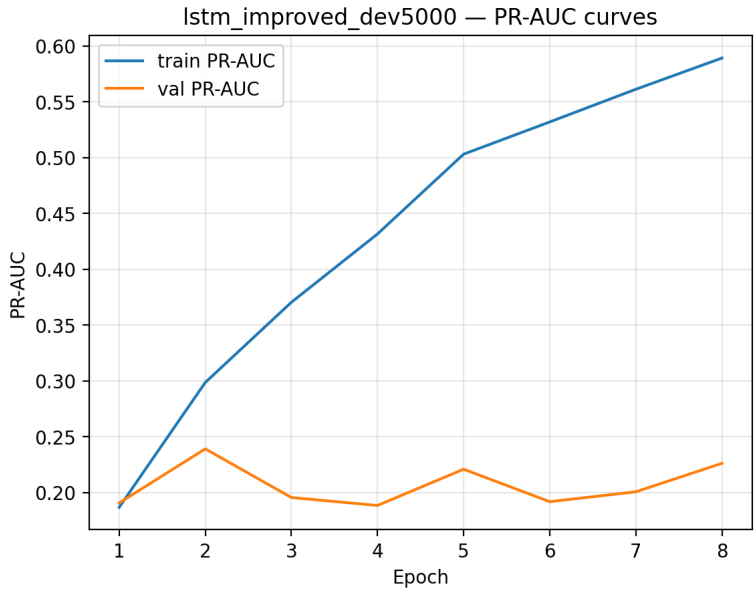
Figur 10: PR-kurva på test (LSTM baseline). PR-kurvan är central här eftersom den beskriver precision som funktion av recall för den sällsynta positiva klassen.



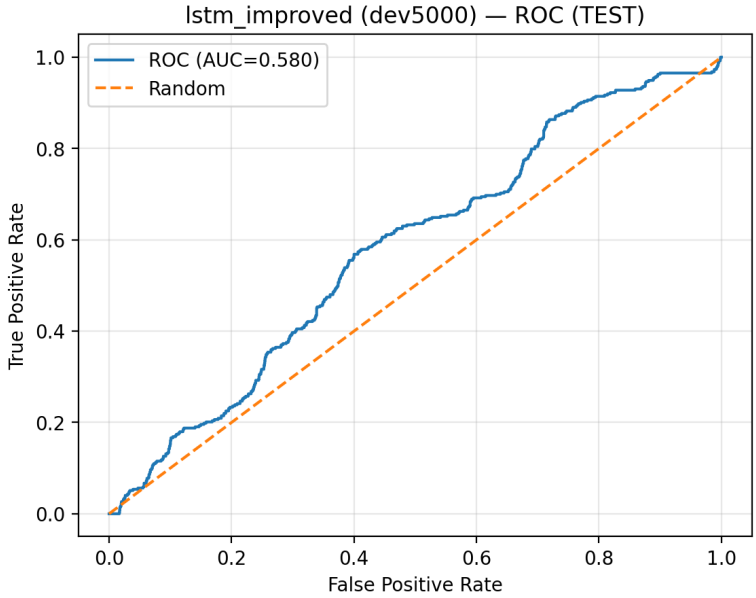
Figur 11: Tröskelkurvor (LSTM baseline). Figuren visar hur precision, recall och F1 varierar med tröskel och gör tradeoffen tydlig.



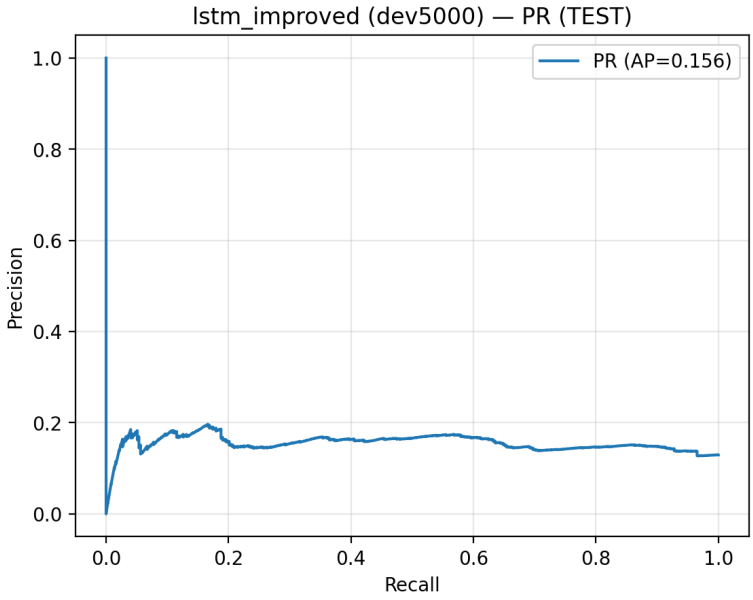
Figur 12: Confusion matrix på test (LSTM baseline, thr=0.50). Figuren visar felprofilen (TP/FP/FN/TN) och hjälper att förstå konsekvenser av tröskelval.



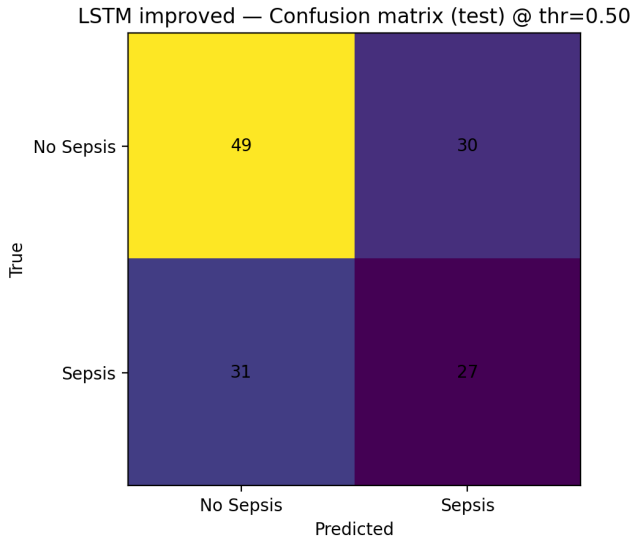
Figur 13: Learning curve för PR-AUC (LSTM förbättrad). Figuren gör det möjligt att jämföra stabilitet och toppnivå mot baseline.



Figur 14: ROC-kurva på test (LSTM förbättrad). ROC-AUC kan ligga nära baseline även när PR- och tröskelresultat skiljer sig.

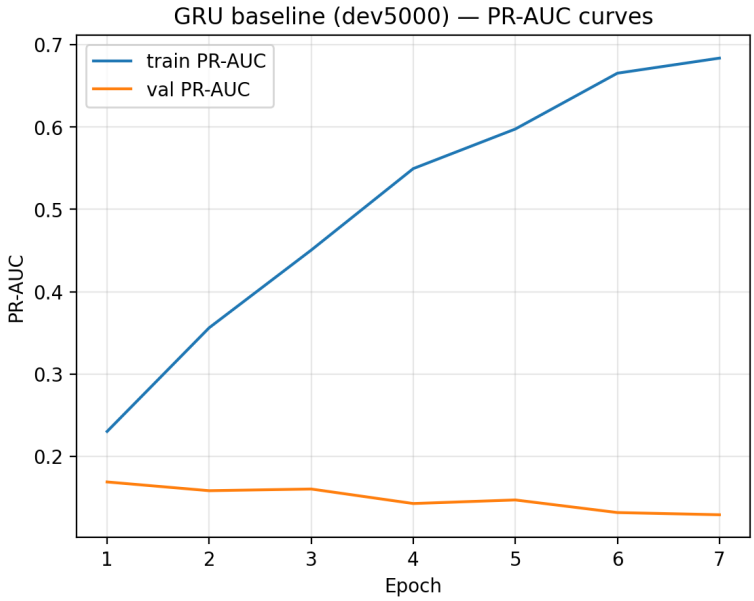


Figur 15: PR-kurva på test (LSTM förbättrad). Denna figur beskriver tradeoff mellan precision och recall och är mest relevant för sepsis som sällsynt klass.

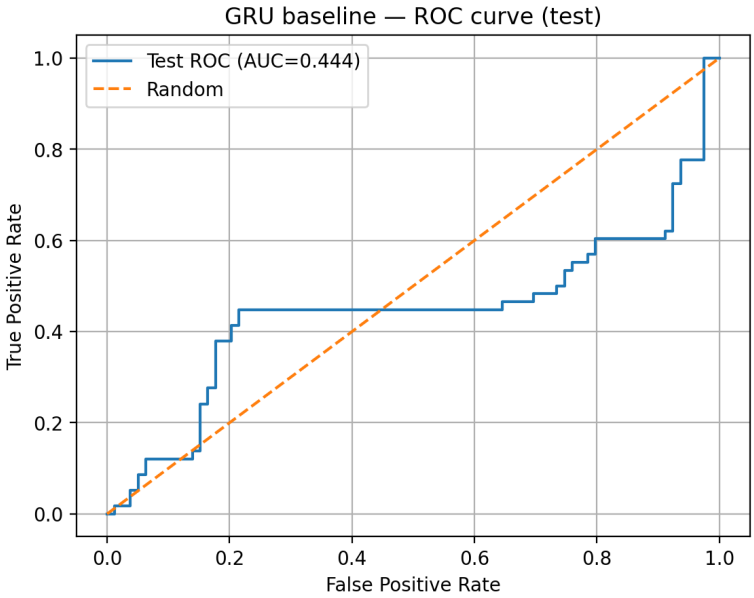


Figur 16: Confusion matrix (LSTM förbättrad). Figuren visar hur felprofilen skiljer sig från baseline och gör det enklare att resonera om falsklarm kontra missade fall.

modellen både som ranking (ROC) och praktisk precision/recall (PR).



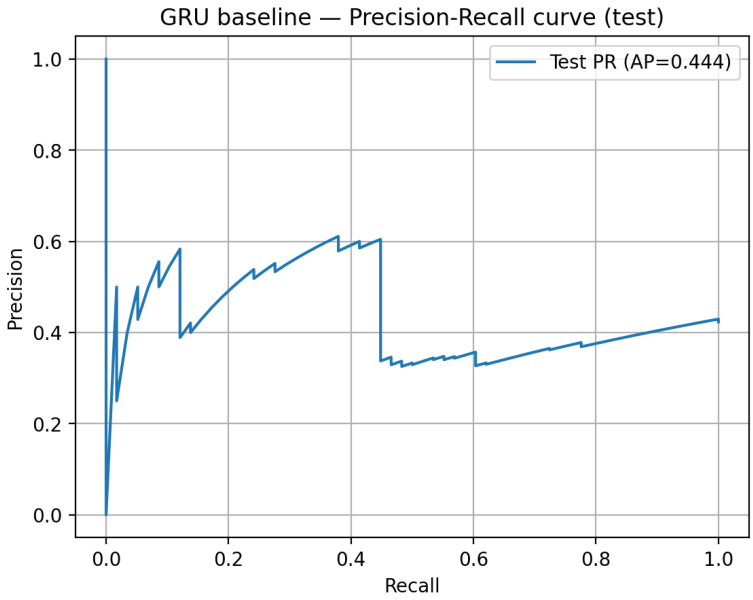
Figur 17: Learning curve för PR-AUC (GRU baseline). Figuren visar hur snabbt modellen når sin valnivå och om den stabiliseras eller börjar överanpassa.



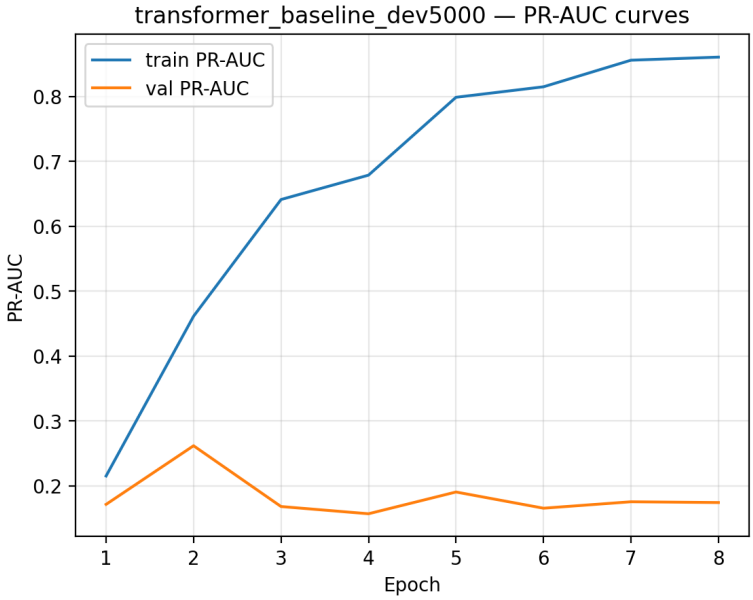
Figur 18: ROC-kurva på test (GRU baseline). ROC-AUC visar separationsförmåga men måste tolkas tillsammans med PR-kurvan i obalanserade problem.

6.6 Transformer baseline

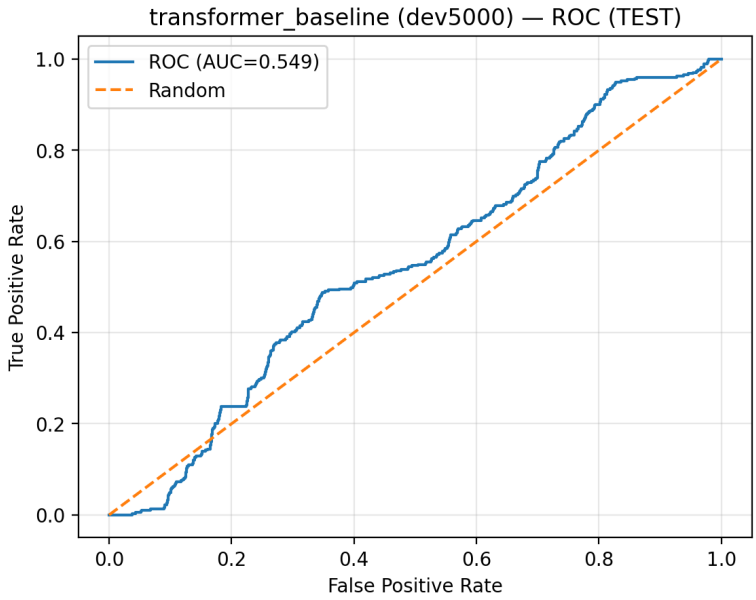
Transformer baseline visar ett tydligt generaliseringsgap. I Figur 20 syns hur val-PR-AUC utvecklas, och i Figur 22 syns att den praktiska precision/recall-relationen blir svagare på test. ROC/PR på test visas i Figur 21 och Figur 22, och confusion matrix i Figur 23 illustrerar hur vald tröskel påverkar felprofilen.



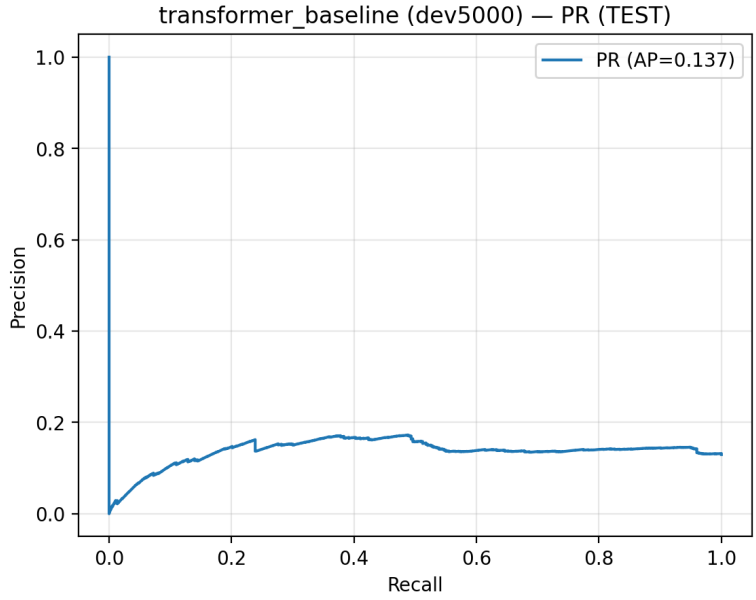
Figur 19: PR-kurva på test (GRU baseline). Figuren visar precision/recall-tradeoff och varför hög recall ofta innebär fler falsklarm när prevalensen är låg.



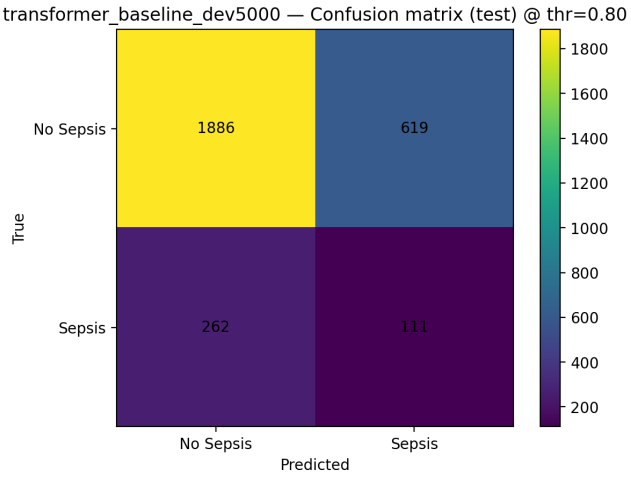
Figur 20: Learning curve för PR-AUC (Transformer baseline). Om valkurvan toppar tidigt medan träningen fortsätter förbättras är det ett klassiskt tecken på överanpassning.



Figur 21: ROC-kurva på test (Transformer baseline). ROC-AUC kan se acceptabel ut även när PR-AUC är låg, vilket gör PR-kurvan extra viktig.



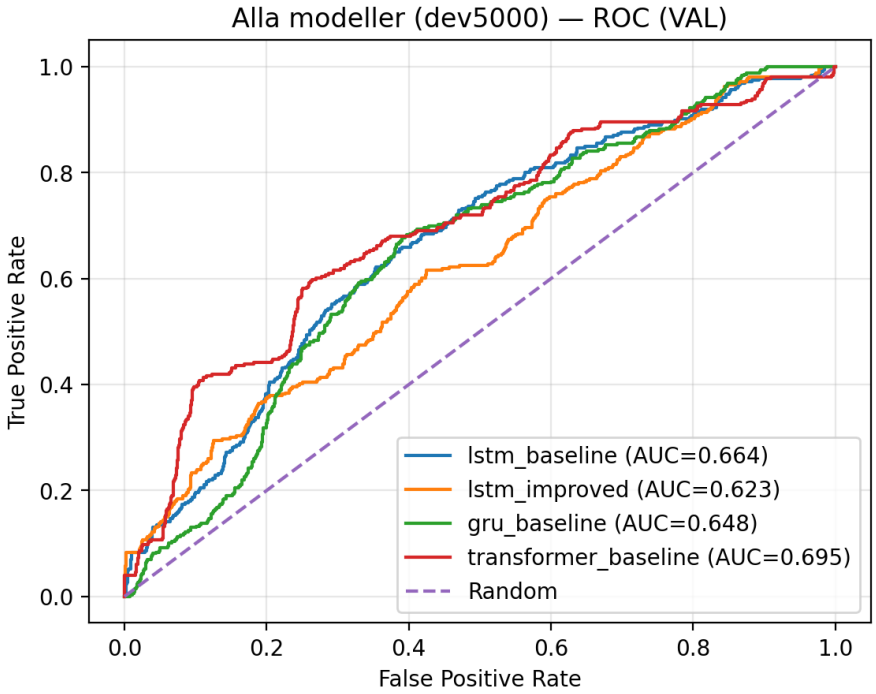
Figur 22: PR-kurva på test (Transformer baseline). Figuren visar att modellen inte håller samma precision/recall som på val, vilket är kärnan i generaliseringsgapet.



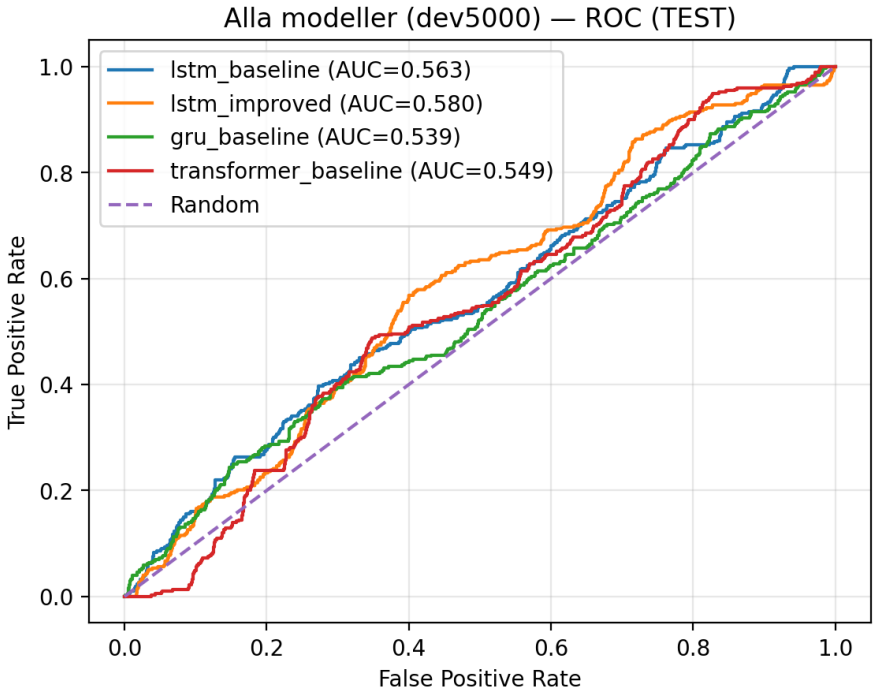
Figur 23: Confusion matrix på test (Transformer baseline, thr=0.80). Figuren visar hur en högre tröskel kan minska falsklarm men samtidigt riskerar att missa fler positiva fall.

6.7 Samlad jämförelse: ROC och PR

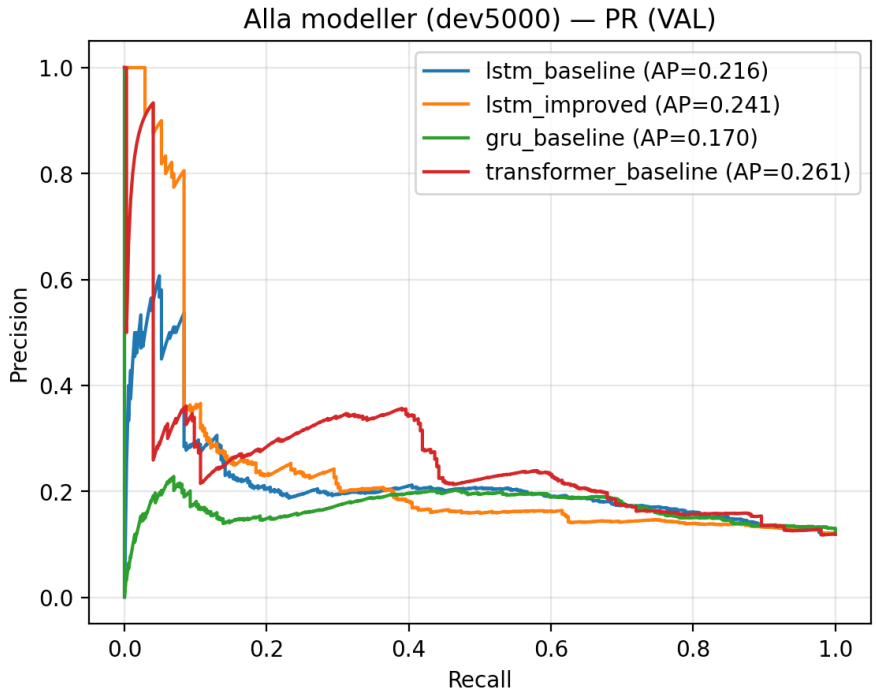
Den samlade jämförelsen i Figur 24 och Figur 25 visar hur modellerna rankar positiva och negativa fall på val/test. I obalanserade problem måste detta alltid tolkas tillsammans med PR-kurvorna i Figur 26 och Figur 27.



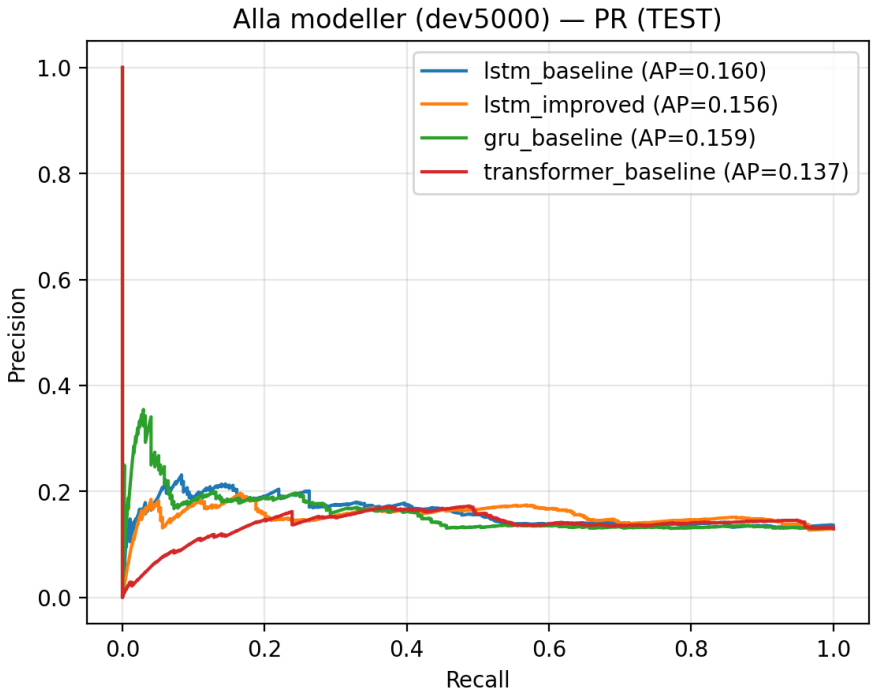
Figur 24: Sammanfattande ROC-kurvor för modellerna på val. Figuren ger en översikt av separationsförmåga, men säger inte allt om precision för den sällsynta positiva klassen.



Figur 25: Sammanfattande ROC-kurvor för modellerna på test. Figuren gör det tydligt om modellerna tappar separationsförmåga när de generaliserar till testdata.



Figur 26: Sammanfattande PR-kurvor för modellerna på val. Detta är den viktigaste jämförelsen i projektet eftersom PR är känsligt för klassobalans och visar precision/recall direkt.



Figur 27: Sammanfattande PR-kurvor för modellerna på test. Figuren visar tydligt hur generalisering påverkar praktisk precision/recall och att en modell kan se stark ut på val men tappa på test.

7 Diskussion och framtida förbättringar

Resultaten visar ganska tydligt att det här problemet är svårt, och att man måste tolka alla siffror med försiktighet. En första observation är att flera modeller kan få okej ROC-AUC, men ändå relativt låg PR-AUC. Det är typiskt för obalanserade problem: ROC-AUC mäter hur bra modellen rankar positiva över negativa i snitt, men när den positiva klassen är sällsynt kan precisionen ändå bli låg vid de trösklar man faktiskt använder. Därför är det rimligt att jag fokuserar på PR-AUC och på tröskelbaserade mått som F1, precision och recall (se Tabell 4 och Tabell 5).

När man tittar på skillnaden mellan val och test syns ett tydligt generaliseringsgap för Transformer (jämför Figur 26 med Figur 27). Det kan bero på att modellen är mer komplex och därför lättare överanpassar, att den behöver mer regularisering eller en bättre träningsstrategi (t.ex. scheduler/warmup), eller att representationen av missingness och oregelbunden sampling inte passar en enkel attention-baseline lika bra som en RNN i detta upplägg. Det betyder inte att Transformer är fel val i sig, men att baseline-versionen här troligen behöver mer tuning för att bli stabil på test.

Tröskelval är en praktisk designfråga. När jag väljer tröskel genom att maximera F1 på val får jag en tydlig och jämförbar rutin, men det är inte säkert att F1 är det man vill optimera i en klinisk process. Om man vill minimera missade fall (FN) kanske man vill maximera recall, vilket GRU baseline lutar åt. Om man vill minimera falska larm (FP) kanske man vill höja tröskeln. Tröskelkurvor (t.ex. Figur 11) och confusion matrices (t.ex. Figur 12 och Figur 23) gör detta lättare att se.

Begränsningar i denna iteration är att imputeringen är relativt enkel (forward-fill + median), vilket kan platta ut signaler när mätningar är glesa. Baseline-Transformern är dessutom känslig för inställningar och kan behöva mer regularisering och träningsstrategi för att generalisera stabilt. Utvärderingen bygger främst på AUC och tröskelmått och fångar inte fullt ut den nyttobaserade

scoring som används i den ursprungliga challenge-formuleringen. Slutligen gör klassobalansen att precision kan bli låg vid hög recall, så resultaten behöver alltid tolkas tillsammans med tröskelkurvor och confusion matrices.

Preprocessing påverkar resultatet mer än man först tror. I kliniska tidsserier är missingness så omfattande att imputering blir en del av modellens antaganden. Forward-fill + median är en pragmatisk lösning, men den kan skapa signaler som inte alltid speglar klinisk verklighet. Mask-kanaler hjälper, men det kan fortfarande vara så att modellen lär sig mönster kopplade till när mätningar tas. En nästa iteration bör därför testa mer robusta metoder, till exempel att explicit modellera tid sedan senaste mätning per variabel.

En möjlig förklaring till att LSTM baseline fungerar bäst här är att den är tillräckligt kraftfull för att fånga temporala mönster men inte så komplex att den överanpassar lika lätt som Transformer. Den har dessutom en relativt enkel head, vilket kan minska risken att modellen lär sig kombinationer som fungerar på val men inte på test.

I litteraturen finns flera andra modeller för sepsis-prediktion än våra LSTM/GRU och vår enklare transformer. Ett vanligt spår är klassiska ML-modeller (t.ex. logistisk regression, Random Forest, XGBoost) där man först gör feature engineering och sammanfattar tidsserier med statistik (medel, max, trend, "senaste värdet"). De kan fungera bra och är ofta enklare att tolka, men de tappar ibland detaljer i tidsordningen som våra sekvensmodeller kan fånga.

Det finns också mer specialiserade tidsseriemodeller (t.ex. TCN eller GRU-varianter som tar med "time gaps") som är byggda för oregelbunden provtagning och saknade värden. Jämfört med dem är våra modeller mer "standard" och vi löser mycket via preprocessing, vilket är enklare men kan göra att vi missar vissa mönster i själva sampling-tiderna.

7.1 Konkreta förbättringar för nästa iteration

- **Bättre hantering av oregelbundenhet:** Lägg till features som tid sedan senaste mätning per variabel eller testa metoder som explicit modellerar oregelbundna tidssteg.
- **Mer systematisk tuning:** En kontrollerad random/grid search för dropout, units och learning rate kan ge stabilare resultat än manuella val.
- **Loss och obalans:** Testa focal loss och jämför med class weights för att se om PR-AUC förbättras genom att modellen fokuserar mer på svåra positiva fall.
- **Kalibrering:** Testa temperature scaling på val för att få bättre sannolikhetskalibrering, vilket gör tröskelval mer stabilt.
- **Transformer-tuning:** Mer dropout, scheduler/warmup, och eventuellt färre encoder-block kan minska generaliseringsgapet.
- **Mer kliniskt driven utvärdering:** Rapportera t.ex. recall vid en fast precision-nivå eller gör en enkel cost-analys (FN vs FP).

8 Slutsats

- I denna rapport byggde jag en reproducerbar pipeline för tidig sepsisprediktion från kliniska tidsserier, där jag gick från råa patientfiler till jämförbar modellinput via imputering, standardisering och fönsterindelning. Poängen var att visa ett komplett data science-flöde och inte bara träna en modell.
- Explorativ dataanalys visade att dev5000 har stora skillnader i tidsserielängd mellan patienter, tydlig klassobalans och omfattande, strukturerad missingness. Dessa observationer påverkar direkt modellval, metrikval och hur man behöver tänka kring tolkning av resultat och osäkerhet.
- Jag prioriterade PR-AUC eftersom sepsis är en sällsynt klass och precision/recall är mer informativt än accuracy i detta fall. Resultaten visar också att ROC-AUC ibland kan se okej ut även när PR-AUC är relativt låg, vilket stärker argumentet att metrikvalet måste matcha problemets natur.
- LSTM baseline framstår som den mest robusta modellen i denna iteration eftersom den har stabil testnivå och en bra balans mellan PR-AUC och tröskelbaserad prestanda. Den generaliserar bättre än Transformer-baseline och ger en rimlig utgångspunkt för vidare förbättringar.
- GRU baseline uppnår hög recall och fångar många sepsisfall, men det sker med många falska positiva. Detta visar att modeller kan vara bra på olika sätt och att praktisk användning kräver att man definierar hur mycket falsklarm man accepterar.
- LSTM förbättrad illustrerar att mer kapacitet inte automatiskt ger bättre generalisering. Den har starka valresultat men kan tappa på test, vilket är en viktig lärdom om överanpassning och vikten av stabila utvärderingsrutiner.
- Transformer baseline visar ett tydligt generaliseringsgap och behöver mer tuning och regularisering för att bli robust. Den här modellen är därför mer ett spår för framtiden än den bästa lösningen i nuvarande iteration.
- Tröskelval är en central designfråga i många modellutvecklingar. Genom tröskelkurvor och confusion matrices blir det tydligt att man kan öka recall genom att sänka tröskeln men då ökar också antalet falska positiva. Därför måste tröskel kopplas till praktiska mål och risker och inte bara en standardinställning.
- Preprocessing och särskilt missingness-hantering, påverkar hela projektet mer än vad man först tror. Forward-fill + median är pragmatiskt men kan skapa bias, och mask-kanaler hjälper men löser inte allt. Detta är ett område där nästa iteration sannolikt kan ge tydliga förbättringar. Olika typer av "imputations"kan testas för att se vilken som ger bäst och mest rättvist resultat.
- Sammantaget visar projektet att den mest användbara modellen i en första iteration ofta är en stabil baslinje som generaliserar rimligt, snarare än den mest avancerade modellen som ser bäst ut på val. Nästa steg är att bygga vidare med bättre tidsrepresentation, mer systematisk tuning och mer kliniskt motiverad utvärdering.

Reproducerbarhet och återkörning (praktisk guide)

Denna repo är organiserad för att jag (eller någon annan) ska kunna återköra experimenten, generera samma figurer och tabeller och sedan bygga vidare med nya modeller. En viktig del är den tydliga mappstrukturen som beskrivs i Appendix A. I praktiken innebär detta att `notebooks/` används för experimentflödet, `src/` innehåller återanvändbar kod och `outputs/` samlar alla genererade artefakter (figurer, tabeller, loggar och checkpoints). Efter en ny körning kan rapporten uppdateras genom att kompilera `report/report.tex`.

För att återköra projektet på ett rent sätt rekommenderar jag att man följer notebook-ordningen i Appendix B. Man börjar med setup/sanity för att sätta paths, seed och kontroller av data/split. Sedan kör man EDA för att exportera figurer om längder, prevalens och missingness. Efter det kör man preprocessing för att skapa fönster ($T = 48$), label-horisont ($H = 6$) och normalisering/statistik som modellerna använder. Därefter tränas modellerna, där checkpoints och learning curves sparas, och sist kör man evaluation som exporterar ROC/PR-kurvor, tröskelkurvor, confusion matrices och sammanfattande tabeller (t.ex. CSV till `outputs/tables/`).

En viktig detalj för reproducerbarhet är att man sparar modellernas bästa version (checkpoint) och att man rapporterar testresultat utan att tuna på test. I mitt upplägg väljs tröskel på val (max F1), och samma tröskel används sedan på test. Det gör jämförelsen mer rättvis och gör det lättare att upprepa resultaten vid framtida körningar.

Tabell 6: Praktisk återkörning: vad kör man och var hamnar resultat? Tabellen gör det tydligt vilka steg som producerar vilka artefakter och var man hittar dem efter en ny körning.

Steg	**Vad gör du?**	**Output (var hamnar det?)**
Setup/Sanity	Kör notebook för paths, seed, datakontroller och split.	Loggar/prints; ev. cache i `data/` eller `outputs/logs/`.
EDA	Kör EDA-notebook och exportera figurer.	`outputs/figures/` (t.ex. missingness, prevalens, längder).
Preprocessing	Skapa fönster, imputera, standardisera, spara statistik.	Bearbetade arrays/caches + ev. loggar i `outputs/logs/`.
Träning (LSTM/GRU/TR)	Träna modeller med early stopping, spara bästa checkpoint.	`outputs/checkpoints/` + learning curves i `outputs/figures/`.
Utvärdering	Generera ROC/PR, tröskelkurvor, confusion matrices och jämförelsetabeller.	`outputs/figures/` + `outputs/tables/` + `outputs/logs/`.
Rapport	Kompilera LaTeX.	PDF i `report/` med figurer från `outputs/`.

AI-deklaration

Jag har använt AI-verktyg (främst ChatGPT och Gemini) som stöd i arbetet med projektet. Min huvudsakliga användning var att få hjälp med att förstå teorin bakom modellerna (särskilt LSTM/GRU och en enkel Transformer-baslinje), samt att få exempel på hur man kan strukturera kod och rapportdelar på ett mer tydligt sätt. I några fall har jag tagit hjälp av AI för att föreslå kodblock eller kodstruktur, exempelvis för att sätta upp en standardiserad träningsloop, tidig stopping, och för att generera figurer som ROC/PR-kurvor och tröskelkurvor. När sådana kodblock användes har jag försökt läsa igenom dem noggrant och anpassa dem till mitt projekt, istället för att bara kopiera rakt av.

AI-stödet var också användbart för att minska förvirring när notebook-koden blev lång och när jag behövde dubbelkolla att jag gjorde saker som att undvika dataläckage, beräkna statistik på train-split och välja rimliga metriker vid klassobalans. Samtidigt har jag varit medveten om risken att AI kan föreslå saker som inte passar exakt för datasetet eller som kan bli för generiska. Därför har jag kontrollerat resultat genom att jämföra mot output från notebooks, och jag har behållit ett studentnära arbetssätt där jag stegvis testar, loggar och tolkar resultaten innan jag drar slutsatser. Jag har sett till att varje del undersöks och att databehandlingen verkligen ligger i linje med min förståelse och förväntningar.

AI verktyg är också kraftfulla för stavkontroll och att hjälpa en att hitta relevanta formuleringar och ord som skulle passa i en teknisk rapport. Det gör att jag kan spara tid och istället fokusera på det mest relevanta som är skälva dataanalysen.

Personlig reflektion

Det här projektet tog mer tid än jag först trodde, framför allt för att kliniska tidsserier är mer stökiga än många dataset jag jobbat med tidigare. Det som var svårast var inte att träna en modell, utan att förstå datan, hantera missingness och skapa en pipeline som känns rimlig och reproducerbar. Jag märkte också att det är lätt att bli för fokuserad på en siffra (t.ex. val PR-AUC) och glömma bort att generalisering på test är det som faktiskt betyder något om man vill säga att en modell fungerar. Jag märkte också att det är viktigt att man som analytiker också har en känsla för datan och deras betydelse. Det gör att man enkelt kan bedömma hur datan ska rensas, förenklas eller användas.

Jag lärde mig mycket av att jämföra modeller som beter sig olika: GRU gav hög recall men många falska positiva, LSTM baseline blev stabilast, och Transformer såg stark ut på val men tappade på test. Det gjorde att jag fick en mer realistisk bild av varför deep learning i klinik inte bara handlar om att välja den modernaste modellen, utan om att hitta något robust, förstå felprofilen och kunna argumentera för tröskelval. En sak jag tar med mig är att bättre preprocessing och bättre utvärderingsstrategi ofta är lika viktigt som själva arkitekturen.

A Projektstruktur och mappöversikt

Tabellen nedan dokumenterar projektets mappstruktur så att det är tydligt var data, kod och resultat finns.

Tabell 7: Mappstruktur och syfte (2 kolumner).

Mapp/fil	**Innehåll och syfte**
`report/`	LaTeX-rapport (t.ex. `report.tex`) som inkluderar figurer/tabeller från `outputs`.
`notebooks/`	Jupyter-notebooks för hela arbetsflödet (setup, EDA, preprocessing, modellträning, utvärdering).
`outputs/`	Genererade artefakter: figurer, tabeller, loggar och checkpoints (t.ex. `figures/`, `tables/`, `logs/`, `checkpoints/`).
`src/`	Återanvändbar Python-kod: dataladdning, preprocessing, fönsterlogik, modellbyggare.
`data/`	Rådata och/eller bearbetad data beroende på konfiguration (patientfiler, cache).
`tests/`	Enhetstester och sanity-tester (om definierat).
`config/`	Konfigurationsfiler för paths/parametrar (om används).
`.vscode/`	Editorinställningar för VS Code (valfritt).
`pyproject.toml`	Projektmetadata och beroenden.

B Körordning för notebooks (rekommenderad för framtiden)

1. `00_setup_and_sanity.ipynb` – Paths, seed, split-kontroll, sanity-checks.
2. `01_eda.ipynb` – Längder, prevalens, missingness, export av EDA-figurer.
3. `02_preprocessing.ipynb` – Imputering, standardisering, fönster ($T = 48$) och label-horisont ($H = 6$).
4. `03_model_lstm.ipynb` – Träning av LSTM baseline och LSTM förbättrad, export av learning curves och checkpoints.
5. `04_model_gru.ipynb` – Träning av GRU baseline, export av figurer och checkpoint.
6. `05_model_transformer.ipynb` – Träning av Transformer baseline, export av figurer och checkpoint.
7. `06_evaluation.ipynb` – ROC/PR, tröskelkurvor, confusion matrices, jämförelsetabeller.

Referenser

[1] PhysioNet / Computing in Cardiology Challenge 2019. Dataset: Early Prediction of Sepsis from Clinical Data. https://physionet.org/content/challenge-2019/1.0.0/

[2] TensorFlow. An end-to-end open source machine learning platform. https://www.tensorflow.org/

[3] Keras. Deep learning API written in Python. https://keras.io/

[4] Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

[5] Reyna, M. A., Josef, C. S., Jeter, R., Shashikumar, S. P., Westover, M. B., Nemati, S., Clifford, G. D., & Sharma, A. Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019. Critical Care Medicine, 48(2):210–217, 2020.

[6] Reyna, M. A., & Clifford, G. D. Voting of predictive models for clinical outcomes: consensus of algorithms for the early prediction of sepsis from clinical data and an analysis of the PhysioNet/Computing in Cardiology Challenge 2019. arXiv:2012.11013, 2020.