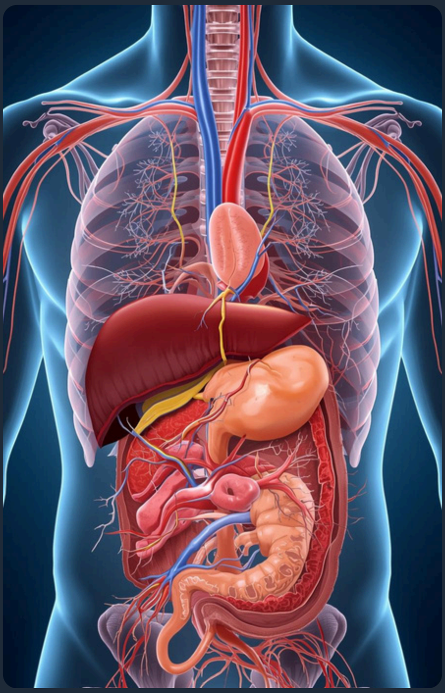
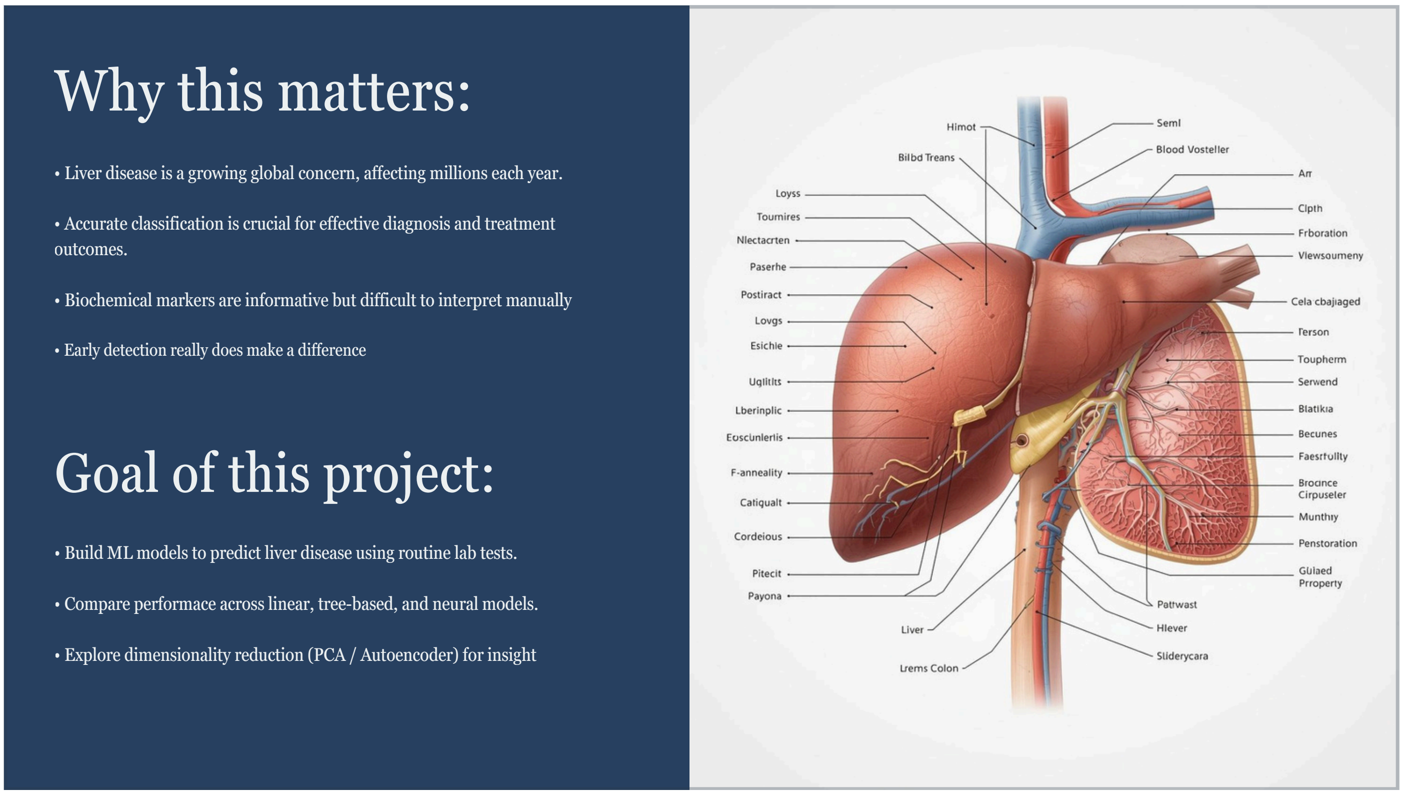
Liver Disease Classification Using ML

Jennifer Martinez
CS 4743 – Dr. Xiaomin Li



Hey everyone, my name is Jenny. Before I get into the technical side of things, I just want to say this project is actually pretty personal for me. My dad struggled with liver disease a few years ago, and honestly, we were lucky we caught it early enough to save his life.

So when we had the chance to choose a dataset for this class, this one immediately stood out to me. I figured... yeah, let's take something meaningful and see if machine learning can help flag issues earlier.



Why this matters:

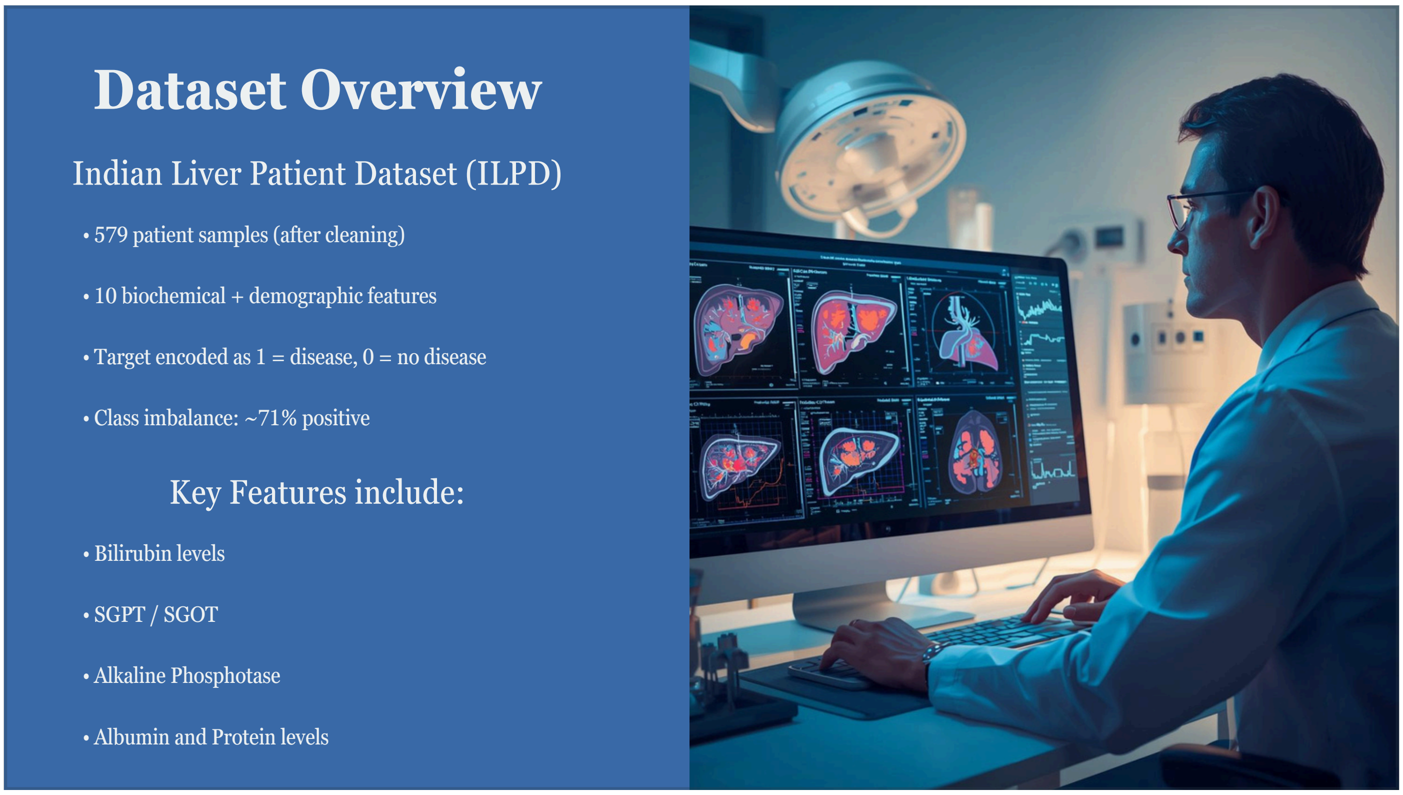
- Liver disease is a growing global concern, affecting millions each year.
- Accurate classification is crucial for effective diagnosis and treatment outcomes.
- Biochemical markers are informative but difficult to interpret manually
- Early detection really does make a difference

Goal of this project:

- Build ML models to predict liver disease using routine lab tests.
- Compare performace across linear, tree-based, and neural models.
- Explore dimensionality reduction (PCA / Autoencoder) for insight

So liver disease is one of those things that can be sneaky. People don’t always notice symptoms until it’s pretty advanced. If doctors can catch it earlier just by looking at routine blood tests, that can literally save so many lives.

But interpreting all those lab numbers together isn’t always easy or obvious. So I wanted to see whether machine learning models could help by spotting patterns that humans might miss.



Dataset Overview

Indian Liver Patient Dataset (ILPD)

- 579 patient samples (after cleaning)
- 10 biochemical + demographic features
- Target encoded as 1 = disease, 0 = no disease
- Class imbalance: ~71% positive

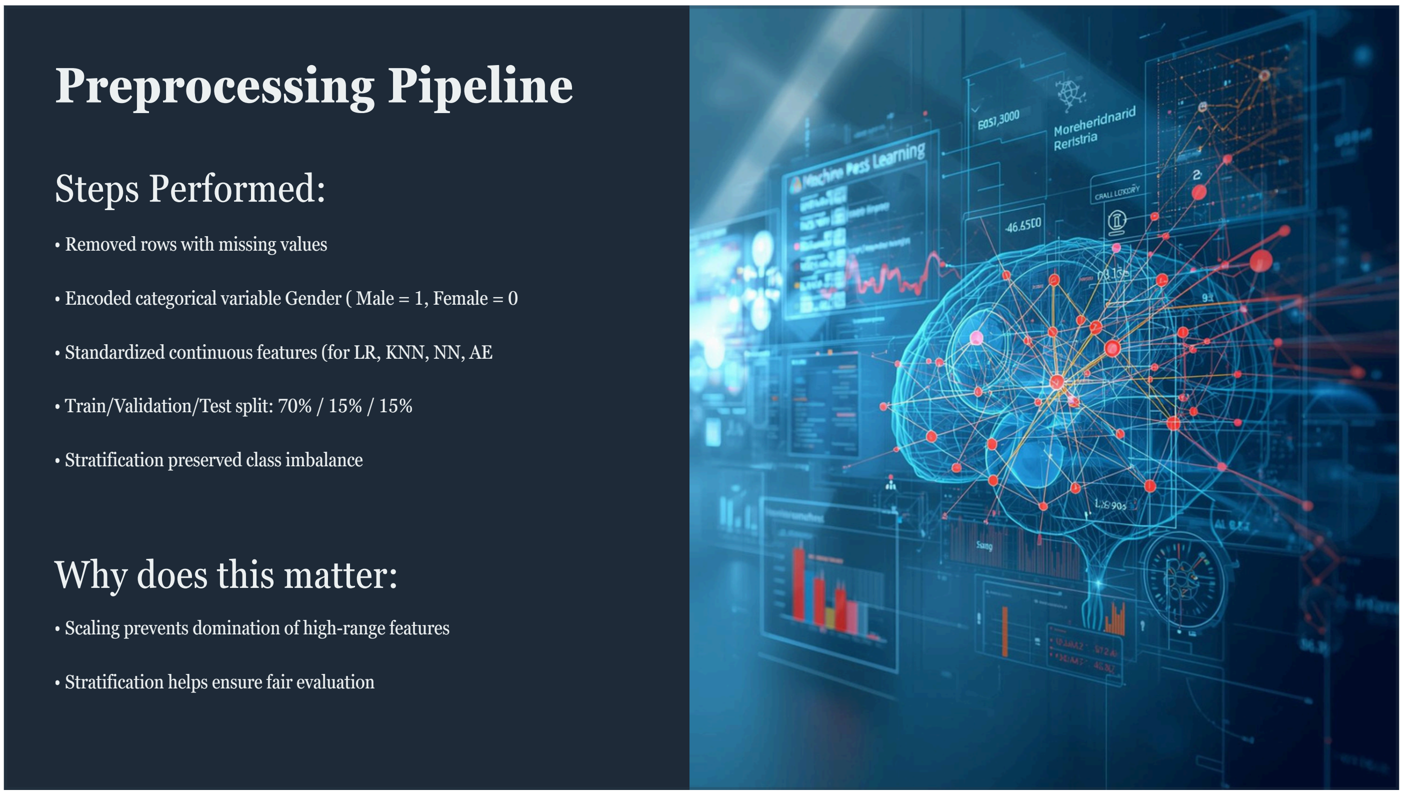
Key Features include:

- Bilirubin levels
- SGPT / SGOT
- Alkaline Phosphotase
- Albumin and Protein levels

This dataset has information from 579 patients — basically their liver-related lab values like bilirubin, enzymes, albumin, and so on.

Something important here is that around 71% of the patients actually have liver disease. So the data isn't balanced, and that affects how we evaluate our models.

I kept thinking how different people's lives are behind each of these rows — it's not just numbers.



Preprocessing Pipeline

Steps Performed:

- Removed rows with missing values
- Encoded categorical variable Gender (Male = 1, Female = 0
- Standardized continuous features (for LR, KNN, NN, AE
- Train/Validation/Test split: 70% / 15% / 15%
- Stratification preserved class imbalance

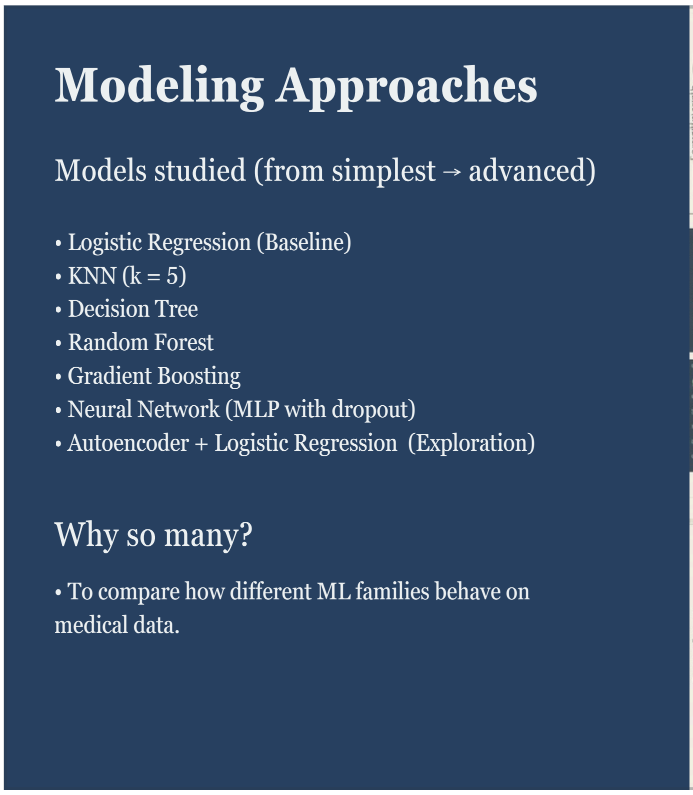
Why does this matter:

- Scaling prevents domination of high-range features
- Stratification helps ensure fair evaluation

I cleaned the dataset by removing rows with missing values. There weren't many, but it could have resulted in inaccurate findings.

Then I encoded Gender, standardized the lab results, and made sure the training, validation, and test sets kept the same imbalance so everything stayed consistent.

This whole step sounds boring, but honestly it definitely made a difference in the model results.



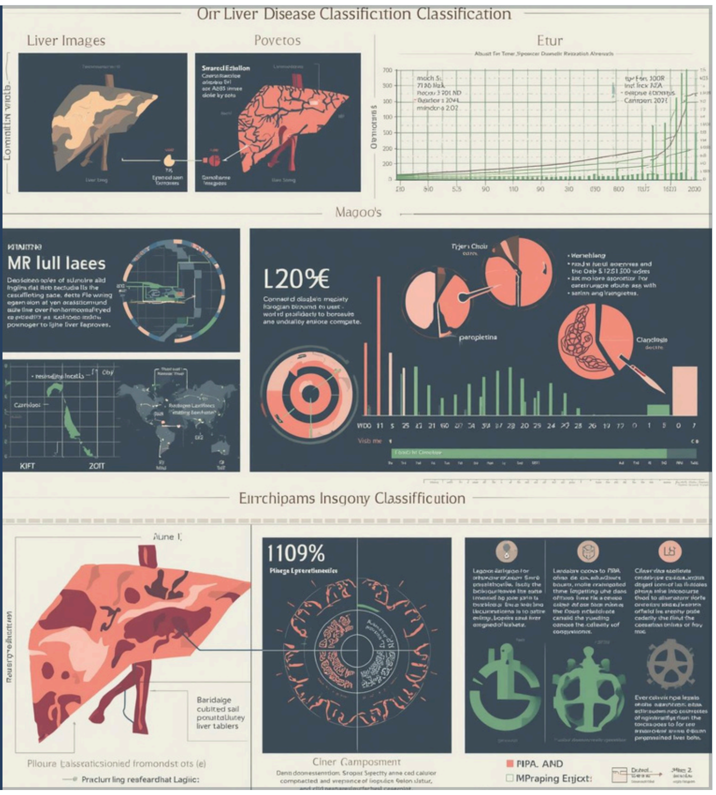
Modeling Approaches

Models studied (from simplest → advanced)

- Logistic Regression (Baseline)
- KNN (k = 5)
- Decision Tree
- Random Forest
- Gradient Boosting
- Neural Network (MLP with dropout)
- Autoencoder + Logistic Regression (Exploration)

Why so many?

- To compare how different ML families behave on medical data.

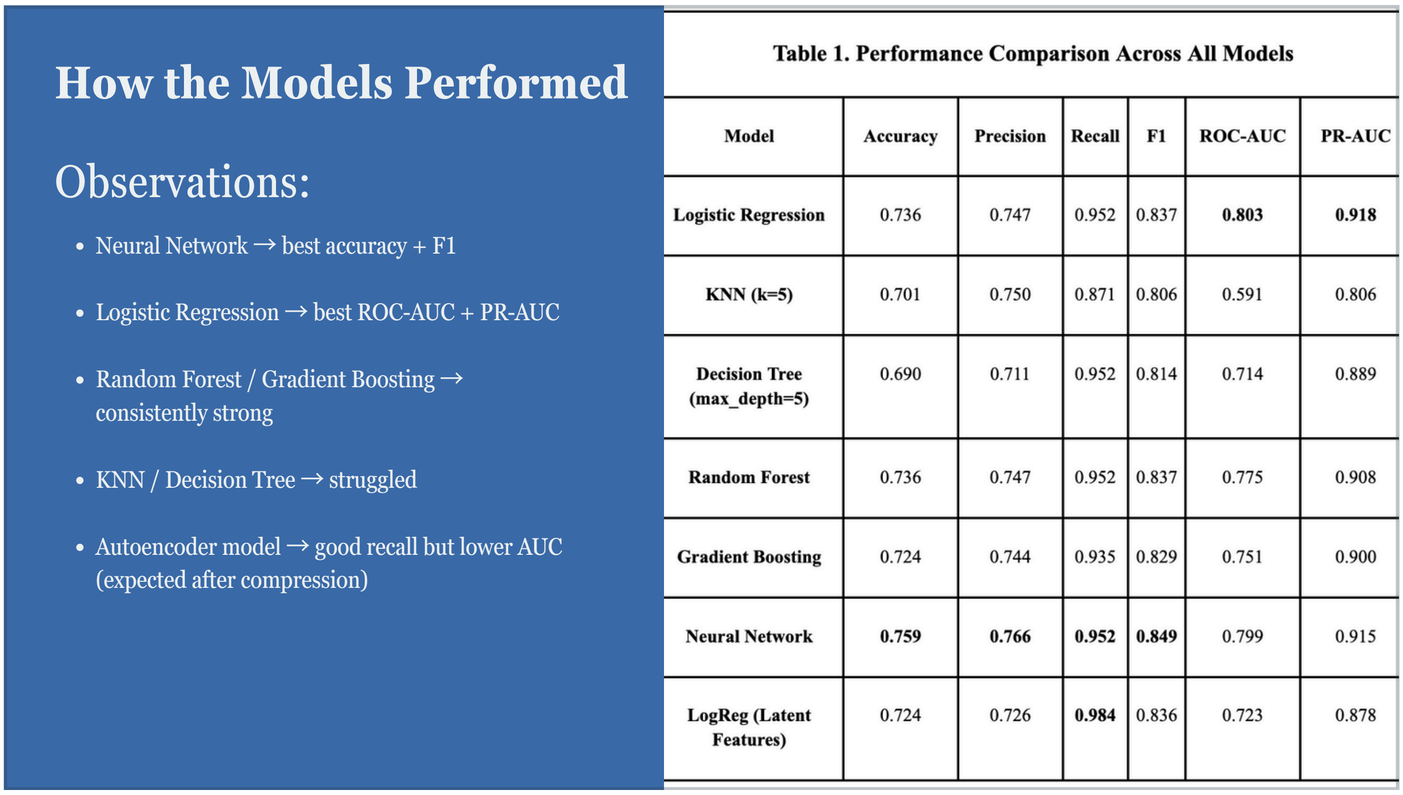


So these are all the models I tried — basically everything we learned this semester.

I started with Logistic Regression as my baseline, then moved into KNN, Decision Trees, Random Forest, Gradient Boosting, and a Neural Network.

And because I was curious, I also added a small autoencoder just to explore how the data looks in a compressed space.

I wanted this project to be more than just ‘build a model, get a score.’ Rather, I wanted to understand why each model behaves the way it does on medical data.



How the Models Performed

Observations:

- Neural Network → best accuracy + F1
- Logistic Regression → best ROC-AUC + PR-AUC
- Random Forest / Gradient Boosting → consistently strong
- KNN / Decision Tree → struggled
- Autoencoder model → good recall but lower AUC (expected after compression)

Table 1. Performance Comparison Across All Models

Model	Accuracy	Precision	Recall	F1	ROC-AUC	PR-AUC
Logistic Regression	0.736	0.747	0.952	0.837	**0.803**	**0.918**
KNN (k=5)	0.701	0.750	0.871	0.806	0.591	0.806
Decision Tree (max_depth=5)	0.690	0.711	0.952	0.814	0.714	0.889
Random Forest	0.736	0.747	0.952	0.837	0.775	0.908
Gradient Boosting	0.724	0.744	0.935	0.829	0.751	0.900
Neural Network	**0.759**	**0.766**	**0.952**	**0.849**	0.799	0.915
LogReg (Latent Features)	0.724	0.726	**0.984**	0.836	0.723	0.878

So here's how everything performed.

The Neural Network ended up having the best accuracy and F1-score, so it was really good at balancing false positives and false negatives.

Logistic Regression actually had the best ROC-AUC and PR-AUC, which surprised me — but it shows that sometimes simple models do just as well, if not better, depending on the data.

Ensemble methods like Random Forest and Gradient Boosting also did great.

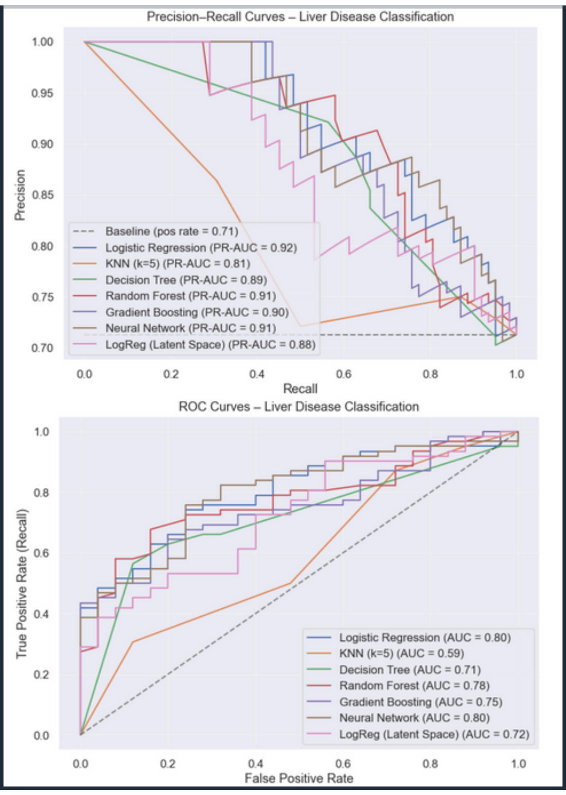
KNN and Decision Tree… yeah, they struggled a bit. Not a huge shock there because the dataset is messy and small.

And the autoencoder model — it didn't beat the others, but it still brought some insight.

ROC & PR Curves

Interpretation

- LR & NN separate classes the best
- Ensembles are close behind
- KNN performed poorly
- Latent-space model sits in the middle — matches AUC scores
- Curves help visualize strengths/weaknesses better than accuracy



These plots give a fuller picture of model performance.

You can see the Logistic Regression and Neural Network curves are the strongest — they separate the disease vs. healthy patients better than the others.

Random Forest and Gradient Boosting are close behind, which shows how stable ensemble methods are.

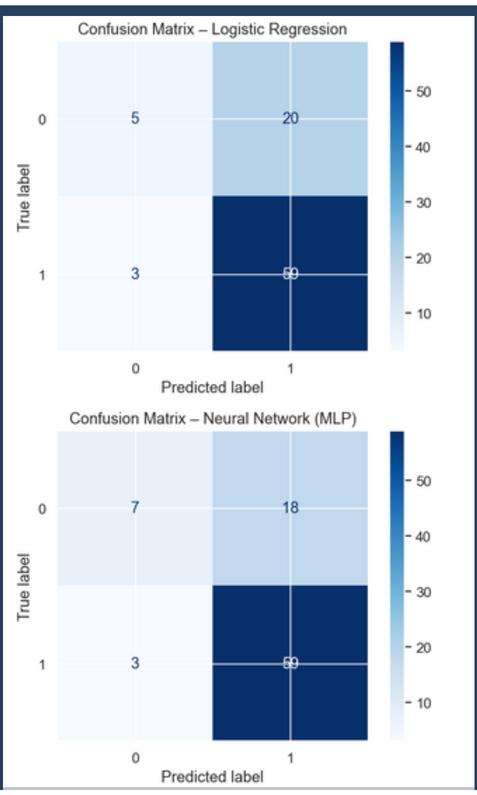
KNN stays pretty low the whole time — really showing that it struggled.

And the model based on the autoencoder's latent features sits somewhere in the middle, which fits what we expected from dimensionality reduction.

Confusion Matrices

Key points:

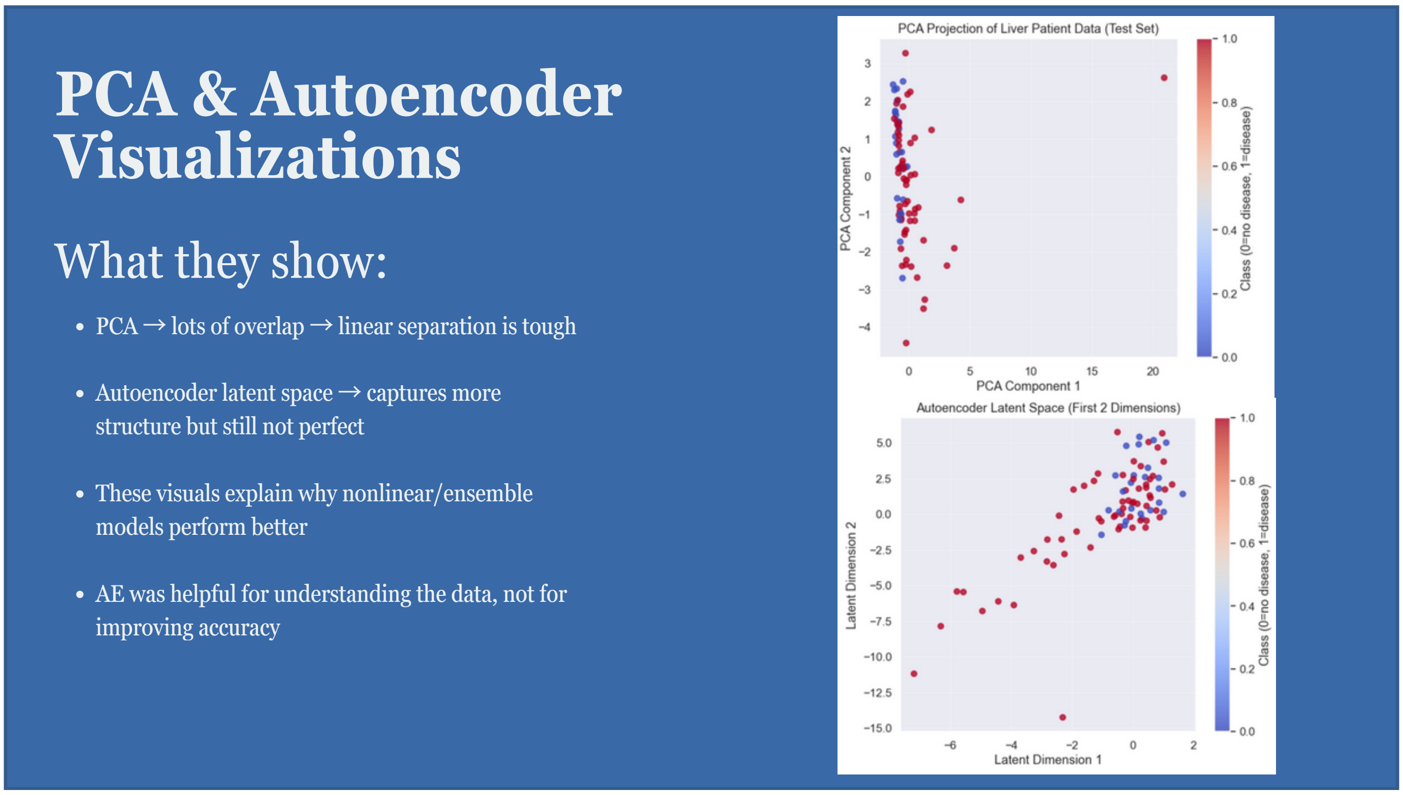
- Both models had very few false negatives
- That’s huge in a medical context (don’t want to miss sick patients)
- NN had slightly fewer false positives
- Overall, both behaved well for screening



These confusion matrices help show exactly where the models got things right or wrong.

Both Logistic Regression and the Neural Network barely had any false negatives. That’s a huge deal, because false negatives are basically the cases where a sick person gets told they're fine — and that’s the last thing you want.

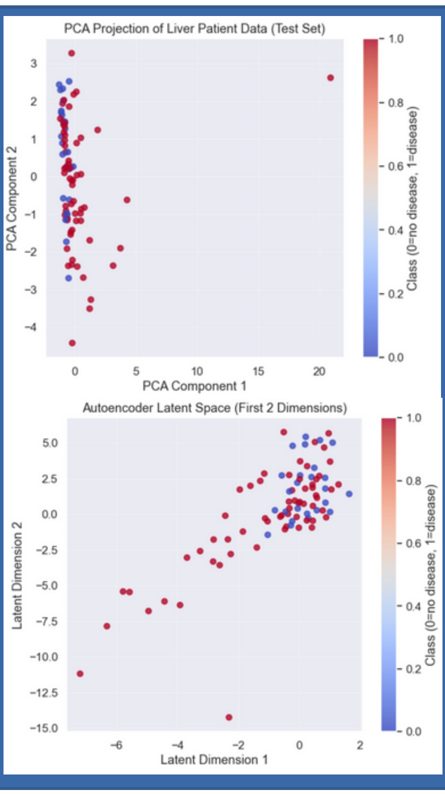
The Neural Network was a little better in terms of false positives, but both models did really well overall.



PCA & Autoencoder Visualizations

What they show:

- PCA → lots of overlap → linear separation is tough
- Autoencoder latent space → captures more structure but still not perfect
- These visuals explain why nonlinear/ensemble models perform better
- AE was helpful for understanding the data, not for improving accuracy



This part was actually one of my favorites.

The PCA plot shows the data kind of all mixed together — there isn't a clear line you can draw to separate the two classes. So that explains why some models hit a performance ceiling.

The autoencoder takes things a step further, showing a little more structure in the data because it's nonlinear. The clusters aren't perfect, but it helps visualize what the neural network is picking up on during training.

Main Takeaways

- Simple models (like LR) can be surprisingly strong
- Neural Networks + ensembles perform well with noisy medical data
- Autoencoders are great for insight, not always accuracy
- How we evaluate models matters a LOT under imbalance

This project helped me appreciate how ML could support early detection – not replace doctors, but help thousands of ppl that otherwise could have never known until it was too late.

Doing this project honestly made me appreciate how powerful even simple models can be — Logistic Regression did amazing on this task.

The Neural Network and the ensembles gave really strong results too, especially in terms of recall, which is something I really cared about because of my dad's experience.

The autoencoder didn't improve accuracy, but it helped me understand the dataset better.

Overall, this project connected everything we learned in class to something that actually matters to me on a personal level. And it made me realize how machine learning could really support doctors one day — not replace them, but help them catch things earlier, and possibly saving thousands of lives.