JACINTA IZUNDU

Artificial Neural Networks (ANNs) with Python

ARTIFICIAL NEURAL NETWORKS (ANNS) WITH PYTHON: PREDICTING HEART DISEASE

clemsjacy@gmail.com

JACINTA IZUNDU

Introduction

“In this project, I built machine learning and neural network models in Python to predict heart disease.
It’s a blend of data science, AI, and healthcare showing how algorithms can learn from patient data and support better medical decisions.”
This project uses Artificial Intelligence (AI) and Machine Learning (ML) techniques in Python to predict whether a person is likely to have heart disease based on their medical data.

I explored the dataset, cleaned and prepared it, and then applied three different models: Decision Tree, a Support Vector Machine (SVM), and an Artificial Neural Network (ANN) and I then compared how well each one could detect heart disease patterns. The ANN had the best performance, but the SVM was more interpretable and reliable for healthcare use.
This project shows my technical ability and my understanding of how AI can be applied responsibly in the real world.

PURPOSE

- To understand how data can be processed and used to train AI systems.
- Compare model performance using accuracy, precision, recall, and AUC scores.
- To learn how AI can assist doctors and researchers in making early, data-driven health predictions as fast as possible.

Tools & Libraries used

Python, Pandas, Scikit-learn, TensorFlow/Keras, Matplotlib, Seaborn

Logos of each tool

Dataset Overview

For the heart disease dataset, I used a real-world dataset containing 303 patient records from the Cleveland Clinic Heart Disease Database. Each row represents one person, and each column gives a health-related measurement or test result for example:

- age – the person's age in years
- sex – 1 for male, 0 for female
- cp (chest-pain type) – four categories of chest-pain symptoms
- trestbps (resting blood pressure) – in mm Hg
- chol (cholesterol) – blood-cholesterol level in mg/dl
- fbs (fasting blood sugar) – 1 if > 120 mg/dl, else 0
- restecg (resting ECG results)
- thalach (max heart rate achieved)
- exang (exercise-induced angina) – 1 = yes, 0 = no
- oldpeak (ST depression induced by exercise)
- slope, ca, thal – ECG-related measures
- target – the main label (1 = heart-disease present, 0 = no disease)

Dataset Overview II

what I did here:

- I Loaded the data directly from the UCI repository using pandas.read_csv() — so I didn't need to download or convert files manually.
- I Named each column clearly (age, sex, cp, chol, etc.).
- I Displayed the first few rows (df.head()) to confirm it loaded correctly.
- I Checked data structure with df.info() and df.describe() to see what kinds of numbers and ranges were present.
- I Prepared for cleaning and exploration spotting missing values (often '?') and outliers that would be handled in next Step
-

Why this dataset overview matters

This step gave me a first look at the information my AI models will learn from and understanding the dataset ensures that:

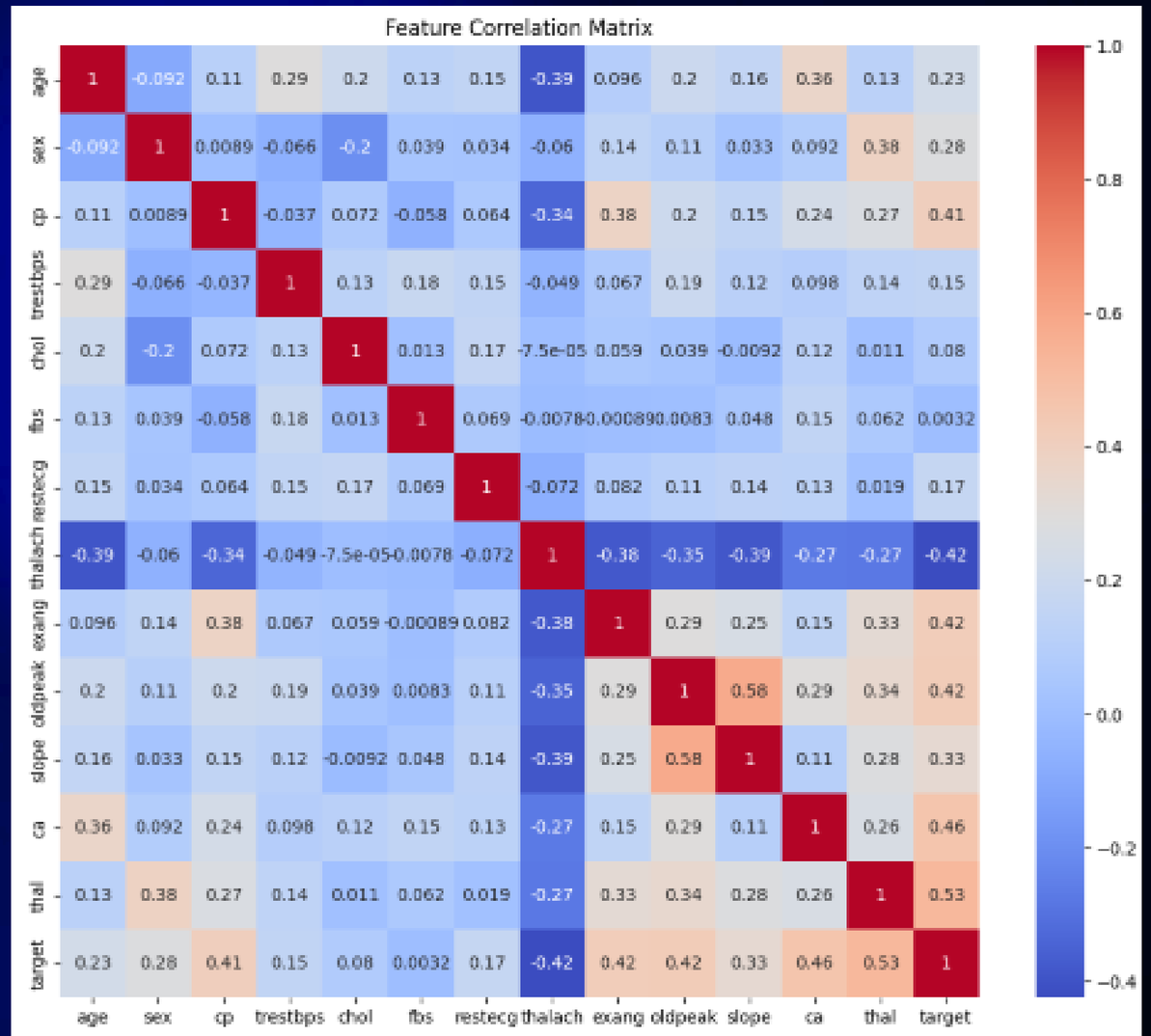
- I know what kind of data I am working with (numbers vs categories).
- I can identify and fix errors or missing values before training.
- I see which features like chest-pain type or cholesterol level, might have a strong link to heart disease.

Data Exploration

This step was about cleaning and understanding the data identifying key features like chest pain and thalassemia that strongly influence heart disease predictions

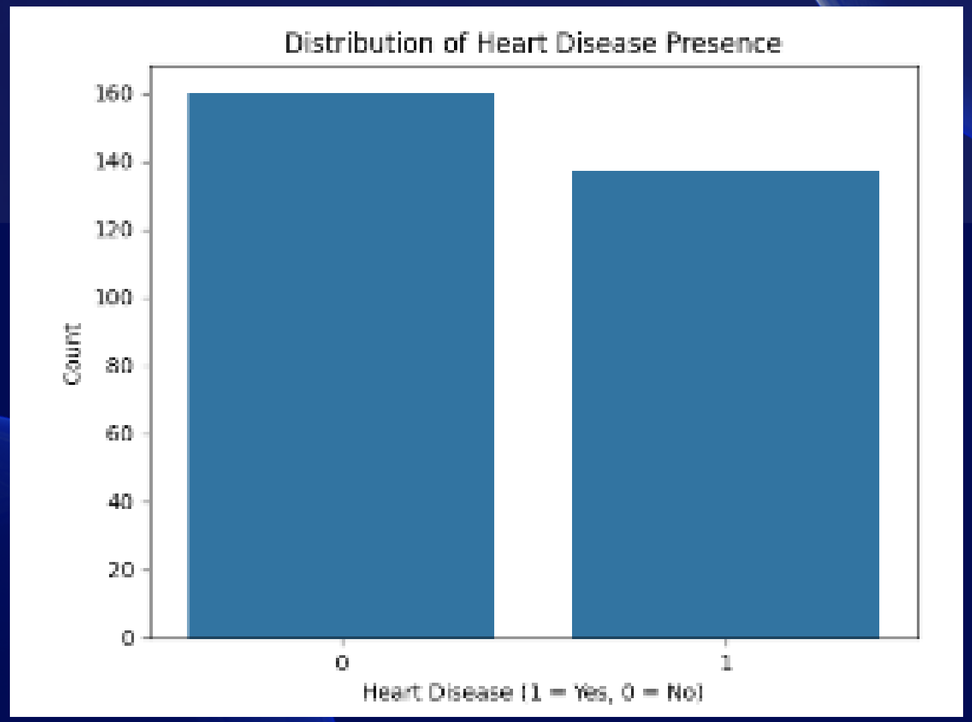
Key Action Performed

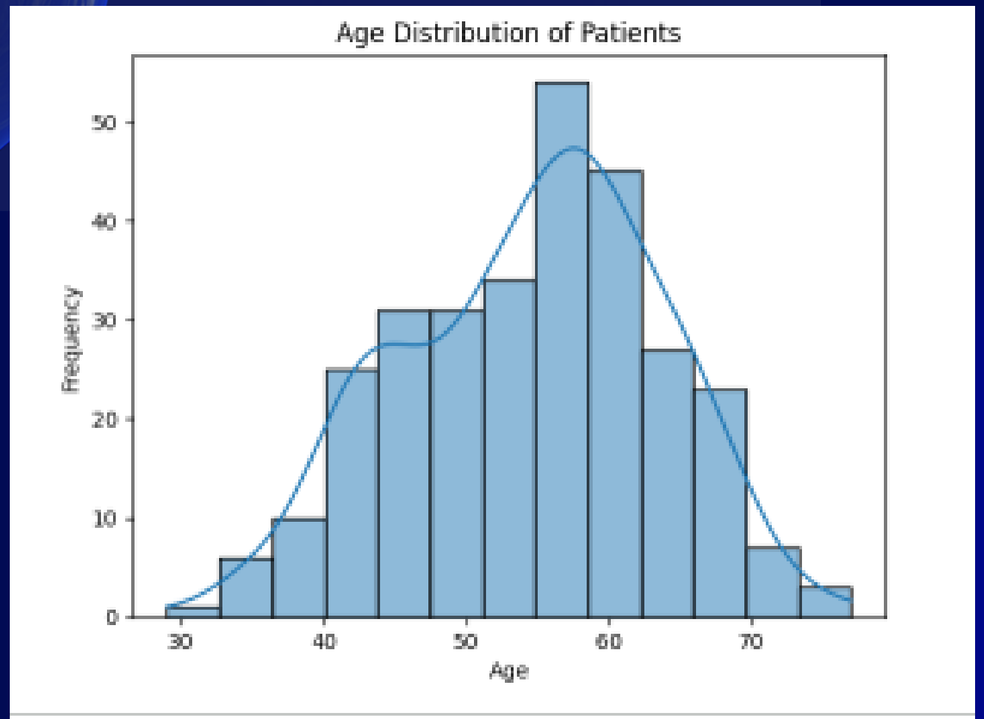
- *Reviewed data structure (.head(), .info(), .describe()) to confirm proper formatting.*
- *Replaced "?" symbols with missing values and cleaned numeric columns (ca, thal).*
- *Converted the target column into a simple binary outcome: → 1 = Heart Disease, 0 = No Disease.*
- *Explored feature relationships using correlation heatmap and count plots.*
- *Verified balanced target classes (almost equal number of disease/no-disease patients).*



Insight

- Chest pain type (cp), thal, and ca was most correlated with heart disease.
- The data was balanced, helping ensure unbiased training.
- No major missing values after cleaning.





Data Preparation

In this step, I transformed and standardized the data so that every variable was numeric, on the same scale, and ready for machine learning.
To get the dataset ready for modeling I ensure all data is in the right format, scaled properly, and split into training and testing sets and To get the dataset ready for modeling I ensured that all data is in the right format, scaled properly, and split into training and testing sets.

```
df.head()
```

```
All specified categorical columns have already been dummified or do not exist in the DataFrame.
```

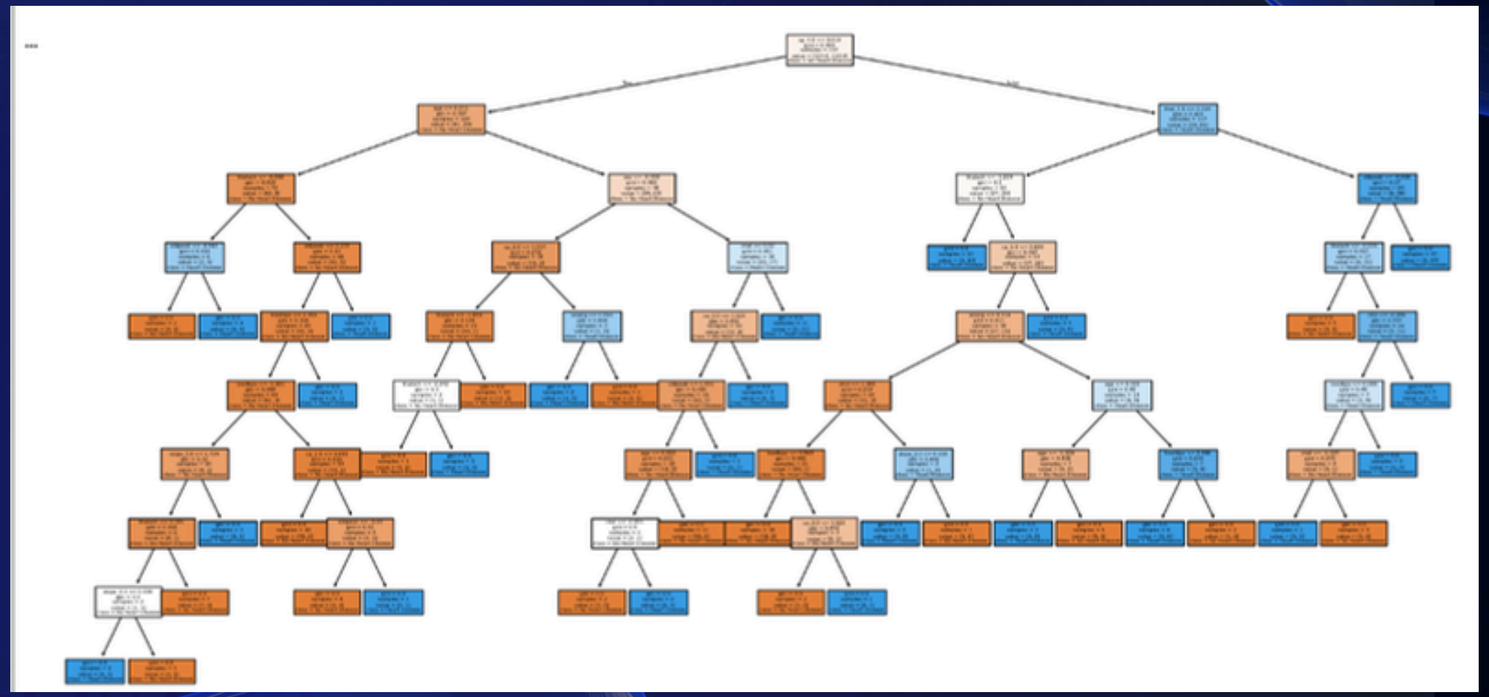
	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	target	cp_2.0	...	cp_4.0	restecg_1.0	restecg_2.0	slope_2.0	slope_3.0	thal_6.0	thal_7.0	ca_1.0	ca_2.0	ca_3.0
0	63.0	1.0	145.0	233.0	1.0	150.0	0.0	2.3	0	False	...	False	False	True	False	True	True	False	False	False	False
1	67.0	1.0	160.0	286.0	0.0	108.0	1.0	1.5	1	False	...	True	False	True	True	False	False	False	False	False	True
2	67.0	1.0	120.0	229.0	0.0	129.0	1.0	2.6	1	False	...	True	False	True	True	False	False	True	False	True	False
3	37.0	1.0	130.0	250.0	0.0	187.0	0.0	3.5	0	False	...	False	False	False	False	True	False	False	False	False	False
4	41.0	0.0	130.0	204.0	0.0	172.0	0.0	1.4	0	True	...	False	False	True	False	False	False	False	False	False	False

```
X_train[:2]  # quick peek
```

```
array([[-1.74167853, -1.52906121,  0.31928358, -0.54421318, -0.43698372,
         0.15179817, -0.73413966, -0.91504143, -0.40925259,  1.59289467,
        -0.98742088, -0.1132277 , -1.02998523,  1.03872391, -0.26906912,
        -0.24090603, -0.825137  , -0.52363494, -0.38069349, -0.27797972],
       [ 0.60111446, -1.52906121,  0.98598146, -0.1617713 , -0.43698372,
         0.97139911, -0.73413966, -0.12163356, -0.40925259, -0.6277879 ,
        -0.98742088, -0.1132277 , -1.02998523, -0.96271972, -0.26906912,
        -0.24090603, -0.825137  , -0.52363494, -0.38069349, -0.27797972]])
```

Model 1: Decision Tree

This step is to build a simple, interpretable model that predicts heart disease using a "yes/no" decision pathway.



Key Results:

- Accuracy: 0.67
- Precision: 0.57
- Recall: 0.67
- F1 Score: 0.61

It identifies Chest Pain (cp), Thal, and Ca as important decision points.

The Decision Tree is simple and easy to interpret, but it was the weakest model here, scoring only 67% accuracy."

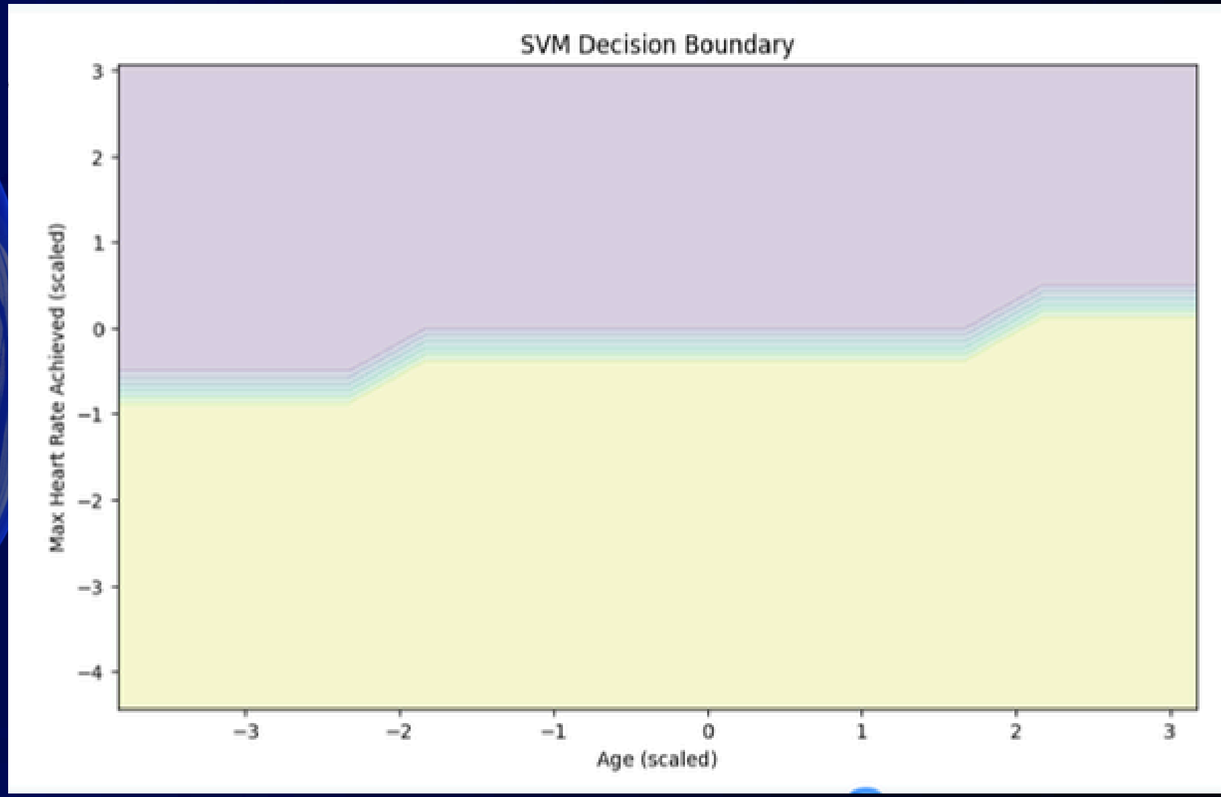
Model 2 — Support Vector Machine (SVM)

In this step is to train a model that finds the best boundary separating patients with and without heart disease

Key Results:

- Accuracy: 0.82
- Precision: 0.76
- Recall: 0.79
- F1 Score: 0.78

- Much stronger performance than the Decision Tree.
- Works especially well after scaling.
- Creates a clear, smooth boundary between classes.
- Good generalization and less likely to overfit.



The SVM performed much better, reaching 82% accuracy with balanced precision and recall. It created a clear decision boundary and generalized well."

Model 3—Artificial Neural Network (ANN)

In this step is to train a deep-learning model that can learn complex, non-linear patterns in the heart-disease dataset.

Key Results:

- *Accuracy: ~0.85*
- *AUC: ~0.92*
- *Best performance among all three models*
- *Learned deeper patterns through hidden layers*

How the ANN Works:

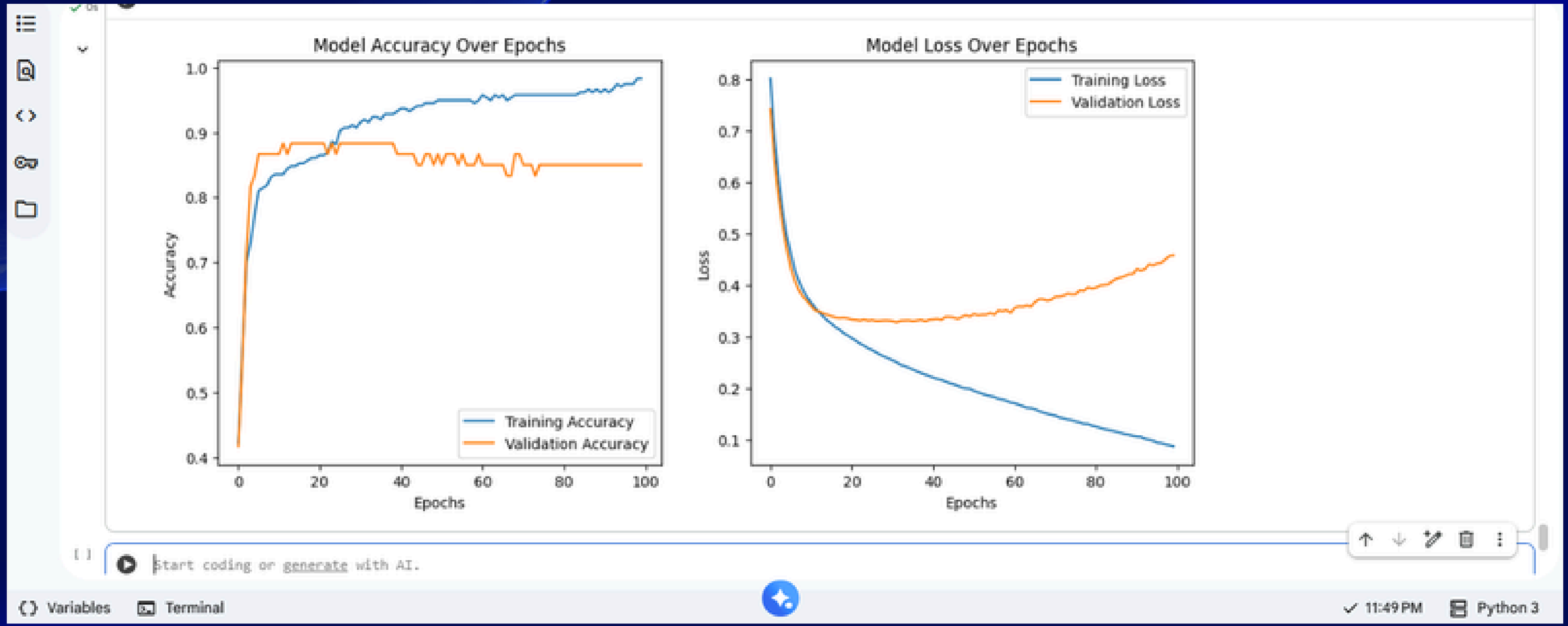
- *Inputs → Hidden Layer 1 (16 neurons) → Hidden Layer 2 (8 neurons) → Output*
- *Used ReLU activation in hidden layers*
- *Used Sigmoid activation for final prediction (0 or 1)*

What This Model Shows:

- *Strong ability to capture complex relationships in medical data*
- *Very high accuracy and AUC*
- *However, it's a "black box" difficult to interpret*

Great predictive performance, but less transparent than SVM or Decision Trees

ANN 2



The ANN gave the highest predictive performance, achieving an AUC of 0.92. It learned deeper patterns but is less interpretable, which is important to consider in healthcare applications.

Model Comparison

This step is to compare the performance of all three models Decision Tree, SVM, ANN, using multiple evaluation metrics.

Performance Summary Table

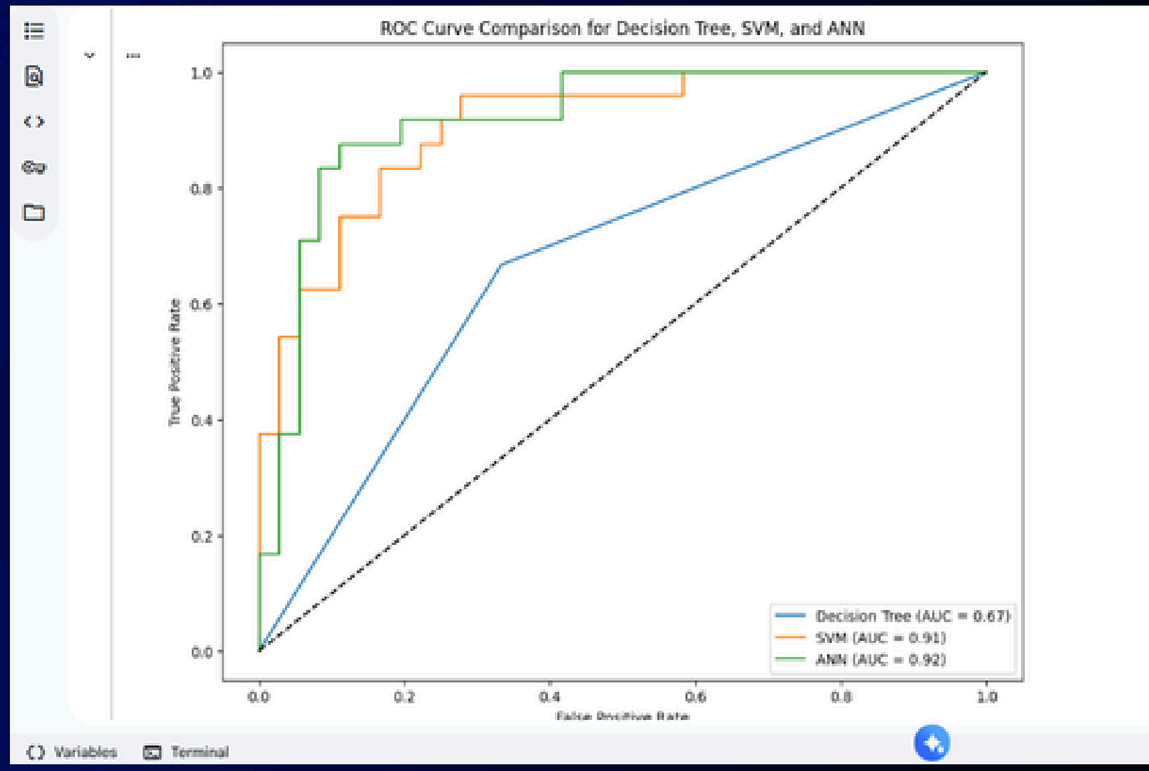
Model	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.67	0.57	0.67	0.61	0.67
SVM	0.82	0.76	0.79	0.78	0.91
ANN	0.85	0.8	0.78	0.79	0.92

Overall Best Model:

ANN (Artificial Neural Network) has the highest accuracy and AUC.

Most Practical for Healthcare:

SVM, because it offers strong accuracy AND better interpretability.



The ANN delivered the best performance overall, but the SVM offers an excellent balance between accuracy and interpretability, making it more suitable for real-world healthcare deployment

What I Achieved

- *I built a complete machine learning pipeline using Python (preprocessing → encoding → scaling → train/test split → modeling → evaluation).*
- *I compared three major ML models: Decision Tree, SVM, and ANN.*
- *I learned how different models capture patterns in health data.*
- *I identified the best performing model for heart disease prediction.*

Technical Skills Gained

- ***Data cleaning and preprocessing***
- ***Correlation analysis and statistical exploration***
- ***Model training and validation***
- ***Performance metrics: Accuracy, Precision, Recall, F1, AUC***
- ***ANN model building using TensorFlow/Keras***
- ***ROC curve analysis and model comparison***

Healthcare & AI Insight

- ***Both ANN and SVM performed strongly, showing that AI can support early medical risk prediction.***
- ***Also learned that model interpretability matters in healthcare doctors must understand the reasoning behind predictions.***

Conclusion

Overall Findings

- *The Artificial Neural Network (ANN) achieved the highest accuracy (≈85%) and the best AUC (≈0.92).*
- *The SVM model performed almost as well and is more interpretable, making it ideal for real-world healthcare use.*
- *The Decision Tree was helpful for understanding the data but not the best predictor.*

What This Means for Employers

This project highlights your ability to:

- *Work with real datasets*
- *Build predictive AI models*
- *Interpret and communicate model results*
- *Apply machine learning to meaningful real-world problems*

Thank You

Get in Touch

+1 217-670-4436

http://linkedin.com/in/jacinta-izundu-b36a20233

clemsjacy@gmail.com

Presented by JACINTA IZUNDU