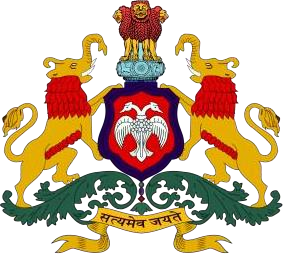
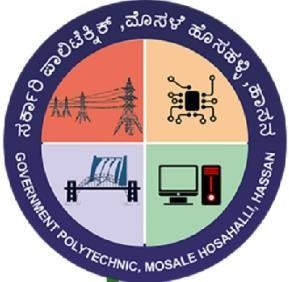
Government of Karnataka

 Department of Technical Education

Bangalore – 560001

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

GOVT. POLYTECHNIC

 MOSALE HOSAHALLI-573212

**2025-2026**

**SPECIALIZATION PATHWAY**

**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**SUBMITTED BY:**

**NAME:- DUVANGOWDA MV**

**REG NO: 189CS23014**

**SUBMITTED TO:**

**Praveen Kumar K**

**Selection Grade Lecturer**

**Dept. of CSE**

# **Table of Contantes**

# SPECIALIZATION PATHWAY

# ARTIFICIAL INTELLIGENCE & MACHINE

# LEARNING

# 

# Problem 1:

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Content** | **Page no** |
| **1** | **Introduction to Regression** | **3** |
| **2** | **Implementation Program of ML&DL** | **4-9** |
| **3** | **Introduction to Classification** | **10** |
| **4** | **Implementation Program of ML&DL** | **11-15** |
| **5** | **Analyse the performance of ML and DL** | **16-17** |

**Problem 2:**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Content** | **Page no** |
| **1** | **Problem Statement** | **18** |
| **2** | **Project plan** | **19** |
| **3** | **Product Backlog** | **20** |

**Problem 3:**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Content** | **Page No** |
| **1** | **GitHub & Repository** | **21** |

**Introduction to Regression:-**

Regression is a statistical and machine learning technique used to study the relationship between a dependent variable (target/output) and one or more independent variables (features/inputs). Its main purpose is to predict continuous values.

Stock markets are influenced by numerous factors such as global events, investor psychology, interest rates, and company performance. This makes stock price prediction a highly complex task.

**Types of Regression:**

**1. Linear Regression**: Relationship between dependent and independent variables is linear.

Example: Predicting house prices based on square footage.

**2. Multiple Linear Regression**: More than one independent variable is used.

**3. Polynomial Regression**: Fits a curve instead of a straight line.

**4. Logistic Regression (for classification)**: Predicts probabilities of categorical outcomes.

**5. Advanced methods**: Ridge, Lasso, Elastic Net, etc., to handle overfitting and high-dimensional data.

**Applications of Stock Price Prediction with DL**

* Investment Strategies – Assists retail and institutional investors in buying/selling decisions.
* Algorithmic Trading – Provides input signals for high-frequency trading bots.
* Hedging & Risk Mitigation – Helps financial firms minimize losses by forecasting downturns.
* Market Sentiment Analysis Integration – Combines with news and social media sentiment to enhance accuracy.
* Portfolio Optimization – Predicts asset behavior for balanced risk-return allocation**.**

**Problem Statement:-**

**Regression –** Loan Approval Prediction

Theory: Predict the loan amount a customer will be approved for. Input features: Applicant income, Co-applicant income, Credit history, Loan term, etc. Output: Loan Amount (continuous).

Models: ML → Linear Regression, Random Forest Regressor

DL → Feedforward Neural Network (Dense layers).

Metrics: MAE , RMSE, R²

**Regression - Rebuild with Machine Learning model:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

data = yf.download("AAPL", start="2020-01-01", end="2023-01-01")

data = data[['Close']]

scaler = MinMaxScaler(feature\_range=(0,1))

scaled\_data = scaler.fit\_transform(data)

X, y = [], []

window\_size = 30

for i in range(window\_size, len(scaled\_data)):

X.append(scaled\_data[i-window\_size:i, 0])

y.append(scaled\_data[i, 0])

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

split = int(len(X)\*0.8)

X\_train, X\_test = X[:split], X[split:]

y\_train, y\_test = y[:split], y[split:]

X\_train = X\_train.reshape(X\_train.shape[0], -1)

X\_test = X\_test.reshape(X\_test.shape[0], -1)

model = Sequential([

Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),

Dense(32, activation='relu'),

Dense(1)

])

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

history = model.fit(X\_train, y\_train, epochs=20, batch\_size=32, validation\_split=0.1, verbose=1)

y\_pred = model.predict(X\_test)

y\_test\_rescaled = scaler.inverse\_transform(y\_test.reshape(-1,1))

y\_pred\_rescaled = scaler.inverse\_transform(y\_pred)

print("MSE:", mean\_squared\_error(y\_test\_rescaled, y\_pred\_rescaled))

print("R2 Score:", r2\_score(y\_test\_rescaled, y\_pred\_rescaled))

plt.figure(figsize=(12,6))

plt.plot(y\_test\_rescaled, label="Actual Prices", color="black")

plt.plot(y\_pred\_rescaled, label="Predicted Prices (TensorFlow Regression)", color="blue")

plt.title("Stock Price Prediction (Regression with TensorFlow)")

plt.xlabel("Days")

plt.ylabel("Price")

plt.legend()

plt.show()

plt.figure(figsize=(8,5))

plt.plot(history.history['loss'], label="Training Loss")

plt.plot(history.history['val\_loss'], label="Validation Loss")

plt.title("Training Loss Curve")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

**Output** :

**17/17** ━━━━━━━━━━━━━━━━━━━━ **1s** 14ms/step - loss: 0.7346 - val\_loss: 0.0143

Epoch 2/20

**17/17** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - loss: 0.0082 - val\_loss: 0.0298

Epoch 3/20

**17/17** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 0.0069 - val\_loss: 0.0057

Epoch 4/20

**17/17** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 0.0039 - val\_loss: 0.0072

Epoch 5/20

**17/17** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 0.0030 - val\_loss: 0.0048

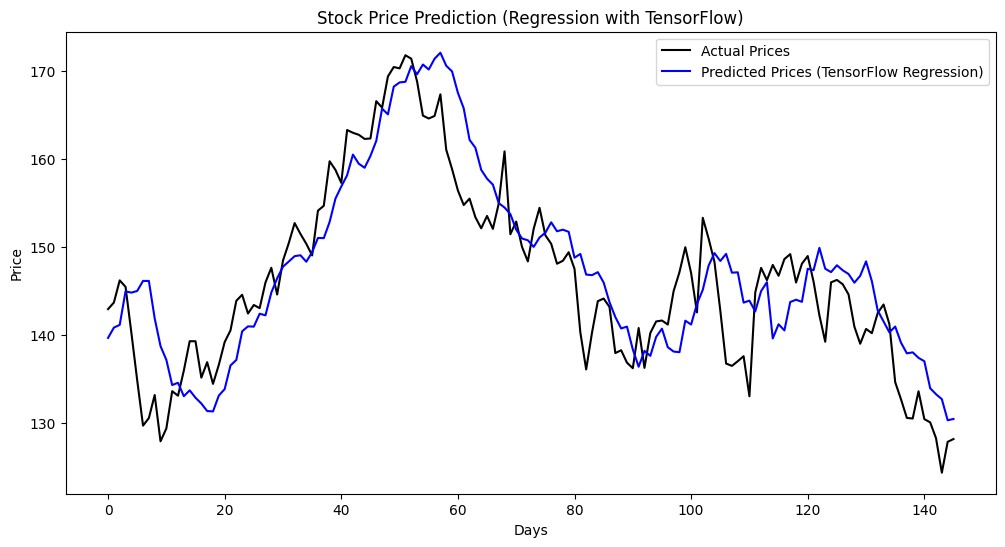
Epoch 6/20

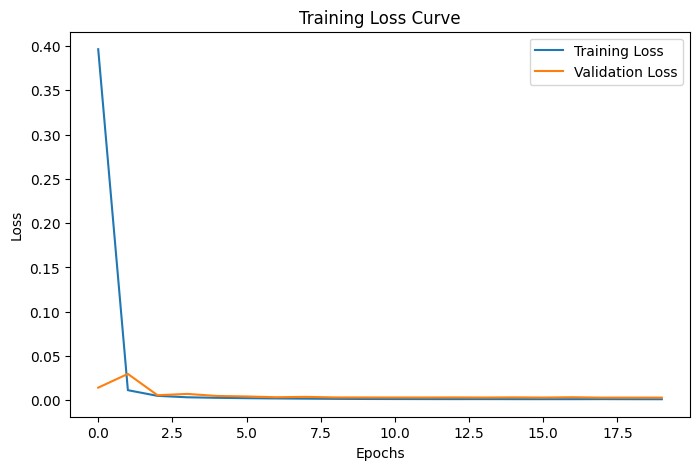
**17/17** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 0.0025 - val\_loss: 0.0043

Epoch 7/20

**17/17** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 0.0019 - val\_loss: 0.0034

Epoch 8/20





**Regression - Rebuild with Deep Learning model:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

data = yf.download("AAPL", start="2020-01-01", end="2023-01-01")

data = data[['Close']]

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(data)

X, y = [], []

window\_size = 60

for i in range(window\_size, len(scaled\_data)):

X.append(scaled\_data[i - window\_size:i, 0])

y.append(scaled\_data[i, 0])

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

split = int(len(X) \* 0.8)

X\_train, X\_test = X[:split], X[split:]

y\_train, y\_test = y[:split], y[split:]

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

model = Sequential([

LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)),

Dropout(0.2),

LSTM(units=50, return\_sequences=False),

Dropout(0.2),

Dense(25, activation='relu'),

Dense(1)

])

model.compile(optimizer='adam', loss='mean\_squared\_error')

history = model.fit(X\_train, y\_train, epochs=20, batch\_size=32, validation\_split=0.1, verbose=1)

y\_pred = model.predict(X\_test)

y\_test\_rescaled = scaler.inverse\_transform(y\_test.reshape(-1, 1))

y\_pred\_rescaled = scaler.inverse\_transform(y\_pred)

print("MSE:", mean\_squared\_error(y\_test\_rescaled, y\_pred\_rescaled))

print("R2 Score:", r2\_score(y\_test\_rescaled, y\_pred\_rescaled))

plt.figure(figsize=(12, 6))

plt.plot(y\_test\_rescaled, label="Actual Prices", color="black")

plt.plot(y\_pred\_rescaled, label="Predicted Prices (LSTM)", color="red")

plt.title("Stock Price Prediction (Deep Learning LSTM)")

plt.xlabel("Days")

plt.ylabel("Price")

plt.legend()

plt.show()

plt.figure(figsize=(8, 5))

plt.plot(history.history['loss'], label="Training Loss")

plt.plot(history.history['val\_loss'], label="Validation Loss")

plt.title("LSTM Model Training Loss Curve")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

# **Ouput :-**

Epoch 1/20

**16/16** ━━━━━━━━━━━━━━━━━━━━ **6s** 109ms/step - loss: 0.2334 - val\_loss: 0.0067

Epoch 2/20

**16/16** ━━━━━━━━━━━━━━━━━━━━ **2s** 57ms/step - loss: 0.0204 - val\_loss: 0.0087

Epoch 3/20

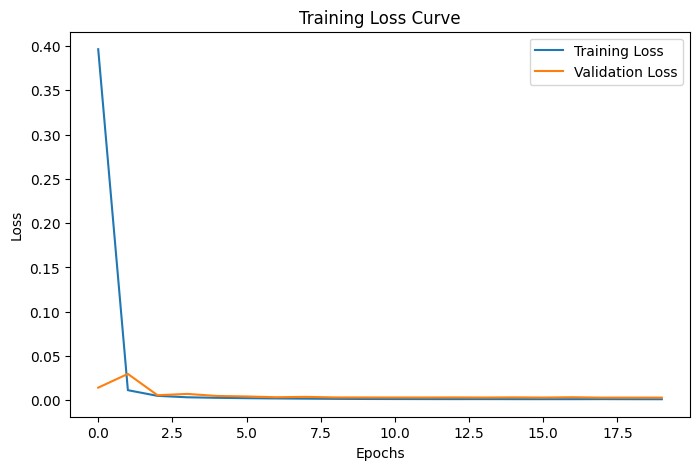
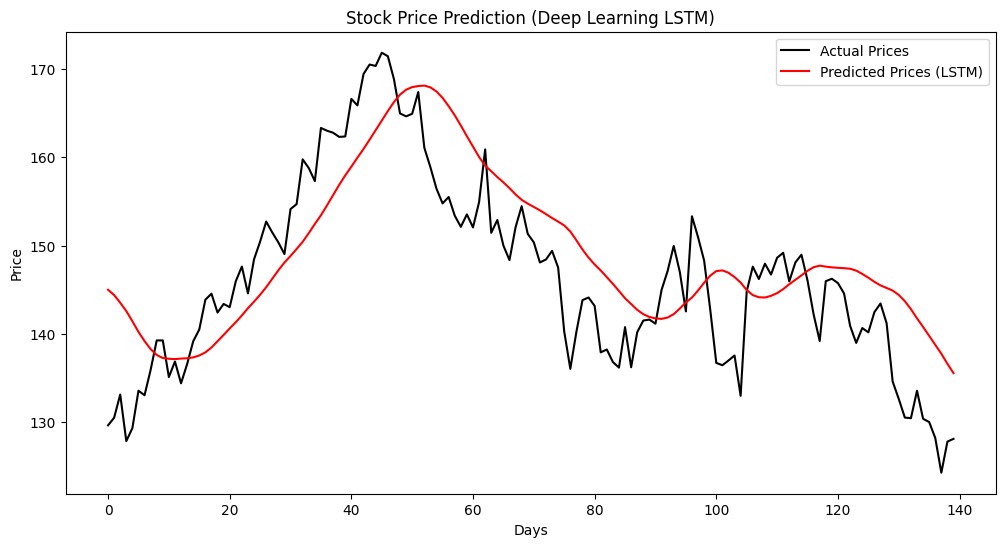
**16/16** ━━━━━━━━━━━━━━━━━━━━ **1s** 52ms/step - loss: 0.0115 - val\_loss: 0.0048

Epoch 4/20

**16/16** ━━━━━━━━━━━━━━━━━━━━ **1s** 55ms/step - loss: 0.0064 - val\_loss: 0.0059

Epoch 5/20

**16/16** ━━━━━━━━━━━━━━━━━━━━ **1s** 53ms/step - loss: 0.0059 - val\_loss: 0.0046



**Introduction to Classification**

Classification is a supervised machine learning technique used to categorize data into predefined classes or labels. Unlike regression, which predicts continuous values, classification predicts discrete outcomes (categories).

Student dropout is a major challenge faced by educational institutions worldwide. Predicting whether a student will continue their studies or drop out can help universities take preventive actions, provide support, and improve student success rates. Machine Learning (ML) and Deep Learning (DL) techniques can be applied to analyze student data such as demographics, attendance, academic performance, and engagement in order to predict dropout risk.

**Types of Classification:**

**1. Binary Classification:** Two possible classes.

Example: Predicting if an email is spam or not.

**2. Multiclass Classification:** More than two classes.

Example: Classifying fruits into apple, banana, or orange.

**3. Multilabel Classification**: An instance can belong to multiple classes at once.

Example: Tagging a photo with labels like beach, sunset, vacation**.**

**Applications:**

Early Warning Systems: Detect at-risk students early and provide intervention.

**1.**Personalized Learning: Suggest targeted resources to students based on their dropout risk.

**2.**Resource Allocation: Help institutions allocate counseling, mentoring, or financial aid effectively.

**3.**Policy Making: Support educational authorities in designing strategies to reduce dropout rates.

**Problem statement:-**

Predict whether a student will drop out (binary classification: dropout = 1 or 0) using historical and demographic data. The goal is to build an accurate, robust deep-learning model that helps institutions identify at-risk students early so they can intervene.

**Models:** ML → Logistic Regression, Random Forest

DL → Neural Network (Softmax output).

**Metrics:** Accuracy, Precision, Recall, F1-score

**Classification - Rebuild with Machine Learning model**

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report, confusion\_matrix

np.random.seed(42)

X = np.random.rand(500, 3) \* 100

y = (X[:, 0] < 50).astype(int)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

model = tf.keras.Sequential([

tf.keras.layers.Dense(16, activation='relu', input\_shape=(3,)),

tf.keras.layers.Dense(8, activation='relu'),

tf.keras.layers.Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

history = model.fit(X\_train, y\_train, epochs=30, validation\_data=(X\_test, y\_test), verbose=1)

y\_pred = (model.predict(X\_test) > 0.5).astype(int)

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

**Output :**

13/13 ━━━━━━━━━━━━━━━━━━━━ 4s 54ms/step - accuracy: 0.4733 - loss: 0.7315 - val\_accuracy: 0.4900 - val\_loss: 0.6971

Epoch 2/30

13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.5330 - loss: 0.6906 - val\_accuracy: 0.5700 - val\_loss: 0.6702

Epoch 3/30

Classification Report:

precision recall f1-score support

0 1.00 0.98 0.99 53

1 0.98 1.00 0.99 47

accuracy 0.99 100

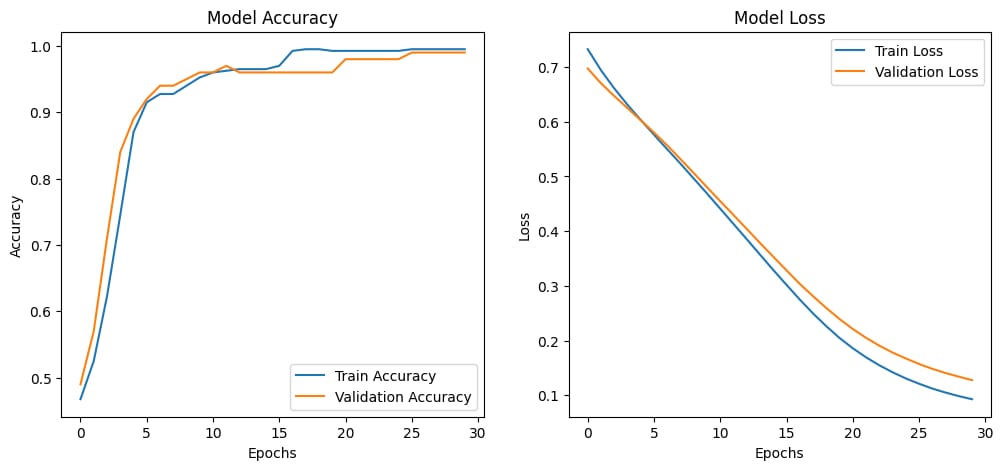
macro avg 0.99 0.99 0.99 100

weighted avg 0.99 0.99 0.99 100

**Confusion Matrix:**

**[[52  1]**

**[ 0 47]]**



**Classification - Rebuild with Deep Learning model:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

np.random.seed(42)

num\_samples = 500

attendance = np.random.randint(40, 100, num\_samples)

assignments = np.random.randint(30, 100, num\_samples)

exam\_score = np.random.randint(20, 100, num\_samples)

dropout = ((attendance < 50) | (exam\_score < 40)).astype(int)

X = np.vstack((attendance, assignments, exam\_score)).T

y = dropout

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

model = MLPClassifier(hidden\_layer\_sizes=(32,16), max\_iter=500, random\_state=42)

history = model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

print(f"Test Accuracy: {acc\*100:.2f}%")

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

plt.figure(figsize=(12,5))

plt.bar(['Continue','Dropout'], np.bincount(y\_test), alpha=0.5, label='Actual')

plt.bar(['Continue','Dropout'], np.bincount(y\_pred), alpha=0.5, label='Predicted')

plt.title('Actual vs Predicted Dropout Counts')

plt.ylabel('Number of Students')

plt.legend()

plt.show()

**Output:-**

Test Accuracy: 100.00%

Confusion Matrix:

[[63 0]

[ 0 37]]

Classification Report:

precision recall f1-score support

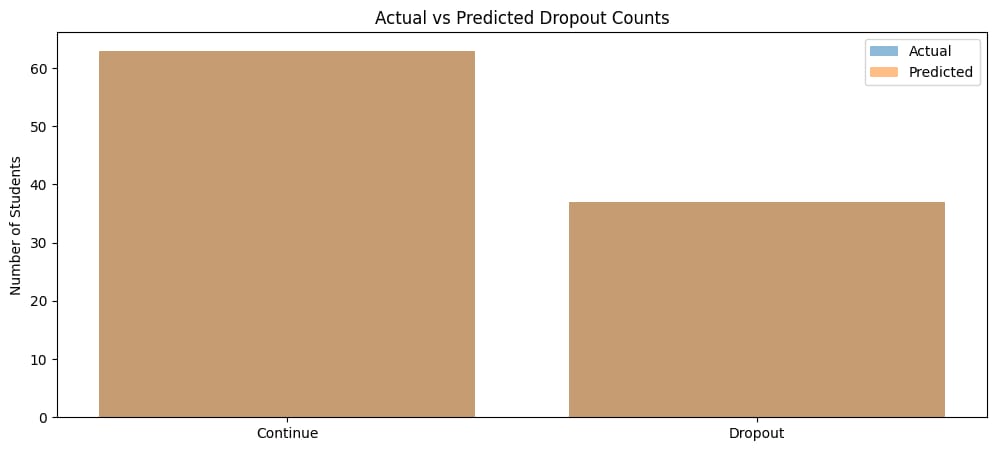
0 1.00 1.00 1.00 63

1 1.00 1.00 1.00 37

accuracy 1.00 100

macro avg 1.00 1.00 1.00 100

weighted avg 1.00 1.00   1.00       100



**Analyse the performance of ML and DL:-**

**Compare:**

Training Time

Accuracy / Error

Overfitting & Underfitting tendencies

**Visualize:**

Training & Validation Accuracy curve (DL)

Training & Validation Loss curve (DL)

Summarize in a comparison table (ML vs DL).

**Performance Analysis of Machine Learning (ML) vs Deep Learning (DL) :-**

**1.Data Requirements:**

ML: Works well on small to medium-sized datasets.

DL: Requires large amounts of labelled data for effective training.

**2.Feature Engineering:**

ML: Needs manual feature extraction/selection (domain knowledge important).

DL: Automatically extracts features through layers (e.g., CNN learns image features).

**3.Computational Power:**

ML: Can run on CPU easily, less hardware-intensive.

DL: Requires GPU/TPU acceleration due to high computation.

**4.Accuracy and Performance:**

ML: Performs well for structured/tabular data (finance, healthcare records, etc.).

DL: Outperforms ML for unstructured data (images, audio, text).

**5.Interpretability:**

ML: Easier to interpret (e.g., Decision Trees, Linear Regression give insights).

DL: Often a black-box, difficult to explain decisions.

**6.Training Time:**

ML: Fast training, less data preprocessing.

DL: Long training times due to multiple layers & parameters.

**7.Use Cases:**

ML: Fraud detection, medical diagnosis, recommendation systems, small datasets.

DL: Self-driving cars, natural language processing, image recognition, speech translation.

**Conclusion:**

Machine Learning (ML) is a branch of Artificial Intelligence that enables systems to learn patterns from data and make predictions or decisions without being explicitly programmed. It is effective for structured data (like tables, numbers, text features) and offers simpler, interpretable models such as regression, decision trees, and SVM.

Deep Learning (DL) is a subset of ML that uses artificial neural networks with multiple layers to automatically learn complex features from large datasets. It excels in tasks like image recognition, natural language processing, and speech understanding, especially when huge amounts of unstructured data (images, audio, video, text) are involved.

**Define Problem Statement**

You are tasked with **rebuilding a regression and classification problem using deep learning models** and comparing their performance with traditional ML models.

**1. Regression – Loan Approval Prediction**

Loan approval is a crucial decision-making process in the banking and financial industry. Traditional methods rely heavily on manual verification, which is time-consuming and prone to human bias.

The objective of this problem is to develop a predictive model using regression techniques that estimates the loan approval probability or loan amount based on applicant details such as income, employment status, credit history, and loan-related features.

This will help banks automate decision-making, reduce processing time, and improve fairness in approvals.

Input: Customer details (income, loan amount, credit score, dependents, etc.)

Output (Regression): Predicted loan amount or probability of approval

**2. Classification – Energy Consumption Forecasting**

With the rise of smart grids and IoT-enabled devices, accurate classification of energy consumption patterns is vital for efficient energy distribution and cost reduction.

The aim of this problem is to classify households/industries into energy consumption categories (e.g., Low, Medium, High) based on historical consumption data and contextual factors such as time of day, season, and appliance usage.

This classification can help utility providers in demand prediction, load balancing, and energy-saving strategies.

Input: Historical energy usage, time, temperature, appliance data

Output (Classification): Category label (Low / Medium / High consumption)

**Objective**: Compare **ML vs DL performance** on both regression and classification tasks in terms of accuracy, error metrics, and efficiency.

**Example**:  
"The goal of this project is to develop and evaluate machine learning (ML) and deep learning (DL) models for regression and classification tasks. The regression task involves predicting loan approval amounts, while the classification task involves predicting energy consumption levels. The project will analyze and compare the performance of ML models (such as Linear Regression, Random Forest, Logistic Regression, SVM) with DL models (such as Neural Networks) to identify trade-offs in accuracy, computational efficiency, and applicability.*"*

**2. Project Plan:-**

A project plan is a formal document that outlines how a project will be executed, monitored, and completed. It serves as a roadmap for the team and stakeholders, ensuring that everyone understands the project’s objectives, deliverables, timeline, resources, and responsibilities.

**Phase 1 – Project Setup**

Define project scope and objectives

Collect datasets (Loan approval dataset, Energy consumption dataset)

Set up Git repository

**Phase 2 – Data Preparation**

Data cleaning (handle missing values, outliers)

Feature selection & engineering

Train-test split

**Phase 3 – Model Development**

Build baseline ML regression and classification models

Build deep learning regression and classification models

**Phase 4 – Model Evaluation**

Evaluate regression using RMSE, MAE, R²

Evaluate classification using Accuracy, Precision, Recall, F1-score

Compare ML vs DL results

**Phase 5 – Documentation & Reporting**

Prepare report (problem statement, methodology, results, conclusion)

Create visualizations (graphs, confusion matrix, error plots)

Finalize GitHub README.md with usage instructions

**Purpose of a Project Plan:-**

Provides clarity and direction for the team.

Ensures effective time, cost, and resource management.

Helps in tracking progress and identifying risks early.

Acts as a communication tool between stakeholders.

**3. Product Backlog**

A product backlog is a prioritized list of tasks, features, enhancements, and fixes that need to be implemented in a project or product. It acts as a to-do list for the development team, maintained by the product owner, and evolves throughout the project as requirements change or new ideas emerge.

**Key Characteristics of a Product Backlog:-**

Dynamic – It is continuously updated as the project evolves.

Prioritized – Items are ordered based on importance and business value.

Detailed progressively – High-priority items are described in more detail than low-priority ones.

Owned by the Product Owner – But visible to the whole team.

| **ID** | **Backlog Item** | **Priority** | **Status** |
| --- | --- | --- | --- |
| 1 | Define problem statement and scope | High | To Do |
| 2 | Create Git repository | High | To Do |
| 3 | Collect loan approval dataset | High | To Do |
| 4 | Collect energy consumption dataset | High | To Do |
| 5 | Perform data preprocessing | High | To Do |
| 6 | Implement ML regression model | High | To Do |
| 7 | Implement ML classification model | High | To Do |
| 8 | Implement DL regression model | High | To Do |
| 9 | Implement DL classification model | High | To Do |
| 10 | Evaluate regression models | High | To Do |
| 11 | Evaluate classification models | High | To Do |
| 12 | Compare ML vs DL performance | Medium | To Do |
| 13 | Document results and analysis | Medium | To Do |
| 14 | Write README.md for GitHub | Medium | To Do |
| 15 | Submit final report | High | To Do |

**4. Git Repository Creation**

Here’s how you can set it up:

**Steps**

Open GitHub and create a new repository → Name it:  
ML-vs-DL-Performance-Analysis

Initialize with a **README.md** file.

Clone repo to local system:

git clone https://github.com/your-username/ML-vs-DL-Performance-Analysis.git

Inside the repo, create folder structure:

├── data/ # datasets

├── notebooks/ # Jupyter notebooks

├── src/ # source code

├── results/ # evaluation metrics & plots

├── report/ # final report

├── README.md # project description

└── requirements.txt # dependencies

# Steps for Git Repository :-

Here’s a step-by-step guide to upload files to GitHub:

**Step 1 :** Go to GitHub and log in.

**Step 2 :** Open the repository where you want to upload the file.

**Step 3 :** Click on the “Add file” button → “Upload files”.

**Step 4 :** Drag and drop your file(s) or click “choose your files”.

**Step 5 :**  Add a commit message (short description of what you uploaded).

**Step 6 :** Click “Commit changes”.