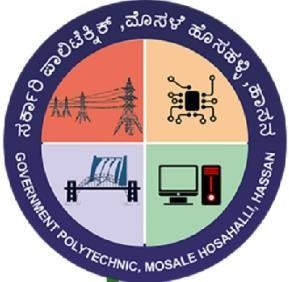
Government of Karnataka Departmentof Technical Education Banglore - 560001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GOVT. POLYTECHNIC

Mosalehosahalli-573212

**2025-202****6**

**SPECIALIZATION PATHWAY**

**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**SUBMITTED BY:**

**NAME:** B S PRAJWAL

**REG NO:** 189CS23009

**SUBMITTED TO:**

H.R Radha BE.,M.Tech

Selection Grade Lecturer/HOD

**Table Contant**

**SPECIALIZATION PATHWAY**

**ARTIFICIAL INTELLIGENCE & MACHINE**

**LEARNING**

**Chapter 1 Page.no**

* 1. Problem statement 3
  2. Project Plan 4
  3. Product Backlog. 5
  4. Implementation. 6
  5. Program. 7
  6. Result and Analysis. 10

**Chapter 2 Page.no**

2.1 Problem statement

2.2 Project Plan

2.3 Product Backlog.

2.4 Implementation.

2.5 Program.

2.6 Result and Analysis.

**Problem Statement :**

Stock market prediction is an important challenge in finance and data science. The goal of this project is to predict stock prices using two approaches:

**Machine Learning (ML):** Random Forest Regressor

**Deep Learning (DL):** Neural Network

* Stock market prices fluctuate daily, making accurate prediction challenging.
* Investors and analysts need reliable tools to estimate future price trends.
* Traditional methods (moving averages, statistical models) often fail on noisy data.
* Machine Learning (ML) can capture non-linear patterns in historical stock data.
* Deep Learning (DL) provides advanced capabilities to learn from complex features.
* This project aims to compare ML and DL models on a regression problem.
* Dataset consists of 100 days of synthetic stock prices with noise added.
* Independent variable: Day, Dependent variable: Price.
* ML approach: Random Forest Regressor.
* DL approach: Neural Network (Dense layers with dropout).

**Project Plan :**

A Project Plan is a structured roadmap that describes how a project will be executed, monitored, and completed.

* Define the objective: Predict stock price using regression.
* Collect/create dataset (synthetic data for demonstration).
* Preprocess data: scaling using StandardScaler.
* Split dataset into training and testing sets (80%-20%).
* Build ML model using Random Forest Regressor.
* Build DL model using a Neural Network with dense layer.
* Train both models on training data.
* Evaluate models using metrics (MSE, MAE, R²).
* Visualize predictions and learning history.
* Compare ML vs DL results to analyze performance**.**
* Train DL model with 100 epochs and validation.
* Compare ML and DL predictions on test set.
* Visualize RF vs DL results with line plots.
* Analyze results using metrics (MSE, R², MAE).
* Document observations and finalize report.

**Product Backlog:**

A project backlog is a list of tasks, features, or work items that need to be completed in a project.

* Define Problem Statement – Clearly describe the task of predicting stock prices using regression models (ML + DL).
* Collect / Generate Dataset – Use a sample dataset (days vs price) or fetch real stock market data.
* Explore Dataset – View first rows, check summary statistics, and understand data structure.
* Preprocess Data – Select features (day) and target (price), handle missing values (if any).
* Feature Scaling – Apply StandardScaler to normalize feature values.
* Split Dataset – Divide into training and testing sets (e.g., 80%-20%).
* Build ML Model – Implement Random Forest Regressor as baseline machine learning model.
* Train ML Model – Fit Random Forest model using training dataset.
* Evaluate ML Model – Calculate metrics like MSE and R² for Random Forest.
* Build DL Model – Create Neural Network with dense layers and dropout for regression.

**Implementation:**

* Dataset: Generated synthetic stock prices (days vs price with noise).
* Preprocessing: StandardScaler applied to normalize features.
* Train-Test Split: 80% training, 20% testing data.
* ML Model: Random Forest Regressor with 100 trees.
* DL Model: Neural Network with 64-32 hidden layers + Dropout.
* Training: DL trained for 100 epochs, batch size = 8.
* Evaluation Metrics: MSE, MAE (DL) and R² (ML).
* Visualization: Scatter plots (RF), Training loss curves (DL).
* Comparison: Overplayed actual vs predicted (RF vs DL).
* Result: Random Forest performed well on small dataset,
* DL showed potential for larger datasets.

**Program to demonstrate stack price prediction using regression model with ML**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor

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

np.random.seed(42)

days = np.arange(1, 101)

price = 50 + 0.5 \* days + np.random.normal(0, 2, size=100)

df = pd.DataFrame({"day": days, "price": price})

print("First 5 rows of dataset:\n", df.head())

X = df[['day']]

y = df['price']

scaler = StandardScaler()

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42

)

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

y\_pred\_rf = rf\_model.predict(X\_test)

print("\n Random Forest Results:")

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

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

plt.scatter(X\_test, y\_test, color='blue', label="Actual")

plt.scatter(X\_test, y\_pred\_rf, color='red', label="RF Predicted")

plt.title("Random Forest Predictions vs Actual")

plt.xlabel("Day (scaled)")

plt.ylabel("Stock Price")

plt.legend()

plt.show()

y\_test\_array = np.array(y\_test)

sorted\_idx = np.argsort(X\_test.flatten())

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

plt.plot(X\_test.flatten()[sorted\_idx], y\_test\_array[sorted\_idx], label="Actual", color="black")

plt.plot(X\_test.flatten()[sorted\_idx], y\_pred\_rf[sorted\_idx], label="RF Predicted", color="red")

plt.title("Stock Price Prediction (Random Forest)")

plt.xlabel("Day (scaled)")

plt.ylabel("Price")

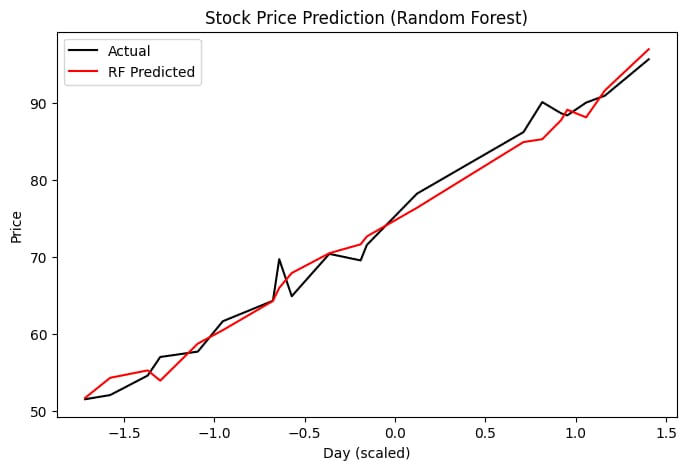
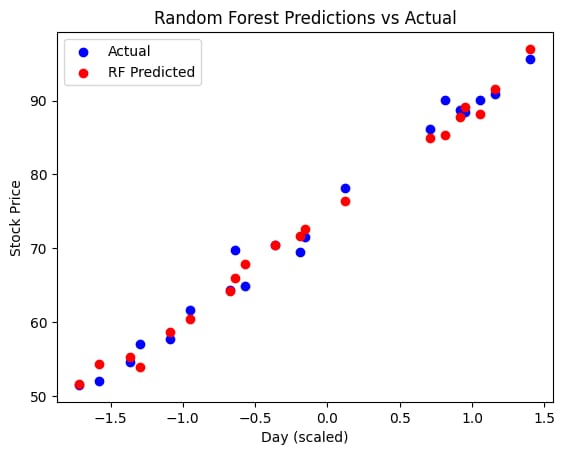
plt.legend()

plt.show()

**OUTPUT :**

Random Forest Results:

MSE: 4.0807103809295615 R2 Score: 0.9802872047844202

**Program to demonstrate stack price prediction using regression model with DL**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

np.random.seed(42)

days = np.arange(1, 101)

price = 50 + 0.5 \* days + np.random.normal(0, 2, size=100)

df = pd.DataFrame({"day": days, "price": price})

print("First 5 rows of dataset:\n", df.head())

X = df[['day']]

y = df['price']

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

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

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42

)

dl\_model = Sequential([

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

Dropout(0.2),

Dense(32, activation='relu'),

Dense(1) # Regression output (no activation)

])

dl\_model.compile(optimizer='adam', loss='mse', metrics=['mae'])

history = dl\_model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test),

epochs=100, batch\_size=8, verbose=0)

dl\_mse, dl\_mae = dl\_model.evaluate(X\_test, y\_test, verbose=0)

print("\n Deep Learning Results:")

print("MSE:", dl\_mse)

print("MAE:", dl\_mae)

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

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

plt.title("DL Model Training Loss (MSE)")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

y\_pred\_dl = dl\_model.predict(X\_test).flatten()

y\_test\_array = np.array(y\_test)

sorted\_idx = np.argsort(X\_test.flatten())

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

plt.plot(X\_test.flatten()[sorted\_idx], y\_test\_array[sorted\_idx], label="Actual", color="black")

plt.plot(X\_test.flatten()[sorted\_idx], y\_pred\_dl[sorted\_idx], label="DL Predicted", color="green")

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

plt.xlabel("Day (scaled)")

plt.ylabel("Price")

plt.legend()

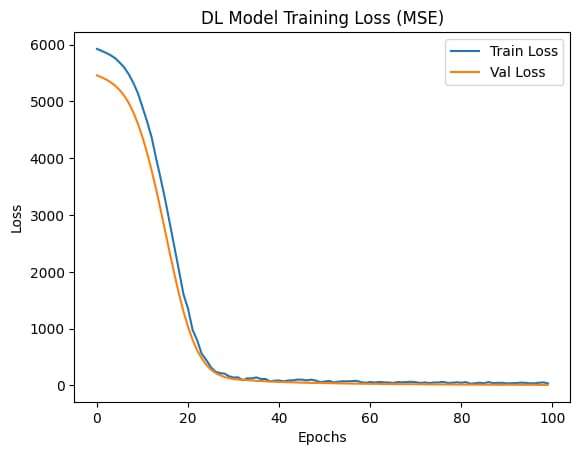
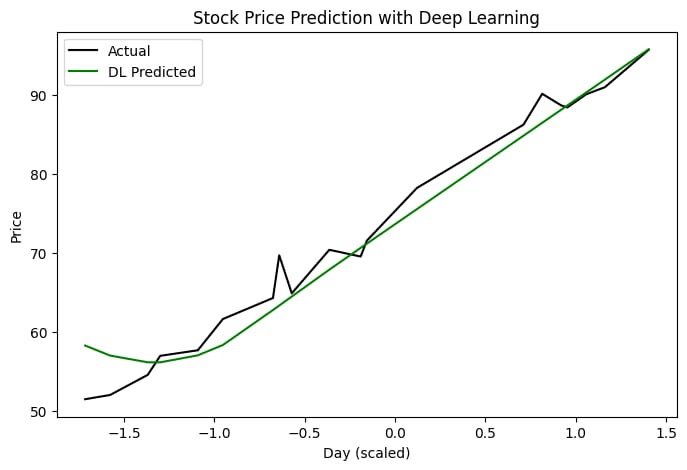
plt.show()

**OUTPUT :**

Deep Learning Results:

MSE: 7.991880893707275

MAE: 2.009674072265625

 ****

**GIT REPOSITORY**

**Problem Statement :**

* The project focuses on predicting student performance as Pass/Fail.
* It uses machine learning (Random Forest) and deep learning (Neural Network) for classification.
* Input features include study hours, attendance, parent support, and previous scores.
* Target variable is binary: pass or fail.
* The project compares ML vs DL approaches.
* Dataset is a small synthetic dataset with 15 student record
* The study aims to help in early identification of at-risk students.
* It uses supervised learning techniques.
* The ML part is implemented with scikit-learn RandomForestClassifier.
* The DL part is implemented with TensorFlow Keras Sequential API.
* Evaluation metrics include accuracy, confusion matrix, classification report.
* Feature scaling ensures fair comparison between models.

**Project Plan:**

* Define the problem statement clearly.
* Collect or simulate student dataset.
* Perform data preprocessing (encoding + scaling).
* Split dataset into training and testing sets.
* Implement Random Forest classifier.
* Train the RF model and evaluate accuracy.
* Visualize feature importance for ML.
* Build a Neural Network model using Keras.
* Train the DL model with proper validation.
* Monitor training vs validation accuracy.
* Compare ML vs DL model performance.
* Document challenges in data preprocessing.
* Prepare visualizations for clarity.
* Analyze which model works best with the dataset.
* Summarize findings in a final report.

**Product Backlog:**

* Dataset preparation (manual or real-world).
* Handle categorical encoding.
* Feature scaling implementation.
* ML model (Random Forest).
* Train-test split design.
* Evaluate ML with metrics.
* Visualize ML feature importance.
* DL model architecture design.
* Compile and train DL model.
* Plot training history (accuracy curves).
* Evaluate DL model accuracy.
* Compare ML and DL results.
* Documentation of results.
* Add visualization for better interpretability.
* Create final project report.

**Implementation:**

* Used pandas and matplotlib for dataset handling and visualization.
* Encoded categorical data (parental support, pass/fail labels).
* Applied StandardScaler for normalization of numeric features.
* Splitted dataset into 80% train, 20% test.
* ML Model: RandomForestClassifier with 100 trees.
* DL Model: Sequential Neural Network with ReLU layers and dropout for regularization.
* Output layer used sigmoid activation for binary classification.
* Trained DL model for 50 epochs with validation monitoring.
* Evaluated models with accuracy, confusion matrix, and classification report.
* Plotted feature importance (ML) and training accuracy (DL).

**Program to demonstrate stack price prediction using Classification model with ML**

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

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

import matplotlib.pyplot as plt

data = {

'study\_hours': [2, 5, 7, 1, 3, 8, 4, 6, 9, 2, 5, 7, 3, 8, 6],

'attendance': [70, 90, 85, 60, 75, 95, 80, 88, 96, 65, 85, 92, 78, 94, 89],

'parent\_support': ['low', 'medium', 'high', 'low', 'medium',

'high', 'medium', 'high', 'high', 'low',

'medium', 'high', 'medium', 'high', 'medium'],

'previous\_score': [40, 65, 78, 30, 55, 85, 60, 70, 90, 35, 68, 80, 58, 87, 72],

'pass\_fail': ['fail', 'pass', 'pass', 'fail', 'fail',

'pass', 'fail', 'pass', 'pass', 'fail',

'pass', 'pass', 'fail', 'pass', 'pass']

}

df = pd.DataFrame(data)

le = LabelEncoder()

df['parent\_support'] = le.fit\_transform(df['parent\_support'])

y = LabelEncoder().fit\_transform(df['pass\_fail'])

X = df.drop('pass\_fail', axis=1)

X = StandardScaler().fit\_transform(X)

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

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

y\_pred = rf\_model.predict(X\_test)

print("\nRandom Forest Results:")

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

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

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

feat\_importances = pd.Series(rf\_model.feature\_importances\_, index=df.drop('pass\_fail', axis=1).columns)

feat\_importances.nlargest(4).plot(kind='barh', color="skyblue")

plt.title("Feature Importance - Random Forest")

plt.show()

**OUTPUT :**

precision recall f1-score support

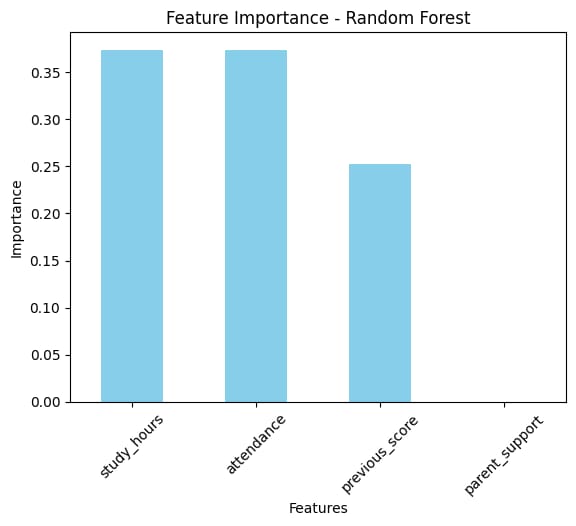
0 1.00 1.00 1.00 2

1 1.00 1.00 1.00 1

accuracy 1.00 3

macro avg 1.00 1.00 1.00 3

weighted avg 1.00 1.00 1.00 3



**Program to demonstrate stack price prediction using Classification model with DL**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

data = {

'study\_hours': [2, 5, 7, 1, 3, 8, 4, 6, 9, 2, 5, 7, 3, 8, 6],

'attendance': [70, 90, 85, 60, 75, 95, 80, 88, 96, 65, 85, 92, 78, 94, 89],

'parent\_support': ['low', 'medium', 'high', 'low', 'medium',

'high', 'medium', 'high', 'high', 'low',

'medium', 'high', 'medium', 'high', 'medium'],

'previous\_score': [40, 65, 78, 30, 55, 85, 60, 70, 90, 35, 68, 80, 58, 87, 72],

'pass\_fail': ['fail', 'pass', 'pass', 'fail', 'fail',

'pass', 'fail', 'pass', 'pass', 'fail',

'pass', 'pass', 'fail', 'pass', 'pass']

}

df = pd.DataFrame(data)

print("First 5 rows of dataset:\n", df.head())

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

le = LabelEncoder()

df['parent\_support'] = le.fit\_transform(df['parent\_support'])

X = df.drop('pass\_fail', axis=1)

y = df['pass\_fail']

y = LabelEncoder().fit\_transform(y)

scaler = StandardScaler()

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42

)

dl\_model = Sequential([

Dense(32, activation='relu', input\_shape=(X\_train.shape[1],)),

Dropout(0.2),

Dense(16, activation='relu'),

Dropout(0.2),

Dense(1, activation='sigmoid') # Binary classification

])

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

history = dl\_model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test),

epochs=50, batch\_size=4, verbose=0)

dl\_loss, dl\_acc = dl\_model.evaluate(X\_test, y\_test, verbose=0)

print("\n Deep Learning Results:")

print("Accuracy:", dl\_acc)

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

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

plt.title("DL Model Training History")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

**OUTPUT :**

study\_ hours attendance parent\_support previous\_score pass\_fail

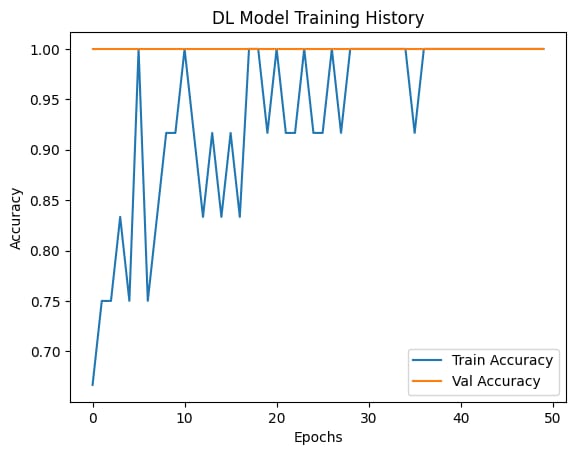
0 2 70 low 40 fail

1 5 90 medium 65 pass

2 7 85 high 78 pass

3 1 60 low 30 fail

4 3 75 medium 55 fail

****

**Git Repository :**