Government of Karnataka

Department of Technical

Education Bangalore – 560001

DEPARTMENT OF COMPUTER SCIENCE AND

ENGINEERING GOVT. POLYTECHNIC

MOSALEHOSAHALLI-573212

2025-2026

SPECIALIZATION PATHWAY

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

SUBMITTED BY:

NAME: INCHANA

REG NO: 189CS23022

SUBMITTED TO:

H. R .Radha, B.E., M. Tech

Selection Grade

Lecturer/ HOD Department

Head Of Department of CSE

Table Content

SPECIALIZATION PATHWAY

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Introduction:

Machine Learning (ML) and Deep Learning (DL) are two major branches of Artificial Intelligence (AI).

- **Machine Learning (ML):** Uses algorithms that learn from data without being explicitly programmed.
- **Deep Learning (DL):** A subset of ML that uses Artificial Neural Network (ANNs) with multiple hidden layers.

In this project :

- **Regression Task:** Loan Amount Detection
- **Classification Task:** Fraud Detection In Transaction

Regression – Rebuild loan amount prediction with deep learning model

Rebuild a **loan amount detection (regression)** model using a deep learning approach and give you a complete, ready-to-run Python script that:

- creates a realistic synthetic dataset and saves it to loan_data.csv
- preprocesses numeric & categorical features
- builds and trains a Keras neural network for regression
- saves the model and shows training / evaluation plots (loss curve, predicted vs actual, residuals)
- prints evaluation metrics (MAE, RMSE, R^2)

Program of Machine Learning :-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Load CSV Dataset
df = pd.read_csv("C:/Users/INCHANA/Downloads/loan_data (3).csv")  # Replace with your actual file path
print("Dataset Head:\n", df.head())  # Show first few rows of the dataset

# Step 2: Data Preprocessing
```

```
# Handle missing values (if any)
df = df.dropna()  # Or you can fill missing values with df.fillna() if required
print("\nAfter Handling Missing Values:\n", df.isnull().sum())  # Check if any missing values
remain

# Check column types and features
print("\nData Info:\n", df.info())

# Step 3: Exploratory Data Analysis (EDA)
# Check the distribution of the loan amount
plt.figure(figsize=(8,6))
sns.histplot(df['LoanAmount'], kde=True)
plt.title("Distribution of Loan Amounts")
plt.xlabel("Loan Amount")
plt.ylabel("Frequency")
plt.show()

# Step 4: Prepare data for modeling
# Assume 'LoanAmount' is the target variable
# And the rest are the features
X = df.drop('LoanAmount', axis=1)  # Features (independent variables)
y = df['LoanAmount']  # Target (dependent variable)

# Handle categorical variables with one-hot encoding (if necessary)
X = pd.get_dummies(X, drop_first=True)  # Drop the first column to avoid multicollinearity

# Step 5: Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Feature Scaling (if necessary)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 7: Train a simple model (Linear Regression)
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Step 8: Make predictions
y_pred = model.predict(X_test_scaled)

# Step 9: Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

```
print(f"\nMean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R^2 Score: {r2}")

# Step 10: Visualize predictions vs actual values
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([0, max(y_test)], [0, max(y_test)], '--r', label="Perfect Prediction")
plt.title("Actual vs Predicted Loan Amounts")
plt.xlabel("Actual Loan Amount")
plt.ylabel("Predicted Loan Amount")
plt.legend()
plt.show()
```

Output:

```
Dataset Head:
   ApplicantIncome  CoapplicantIncome  ...  Property_Area  Loan_Status
0            17795               1687  ...      Semiurban            Y
1             2860               6833  ...          Urban            N
2             7390               2427  ...          Rural            Y
3            23575               9760  ...          Rural            Y
4            13964               4000  ...          Urban            Y

[5 rows x 11 columns]

After Handling Missing Values:
 ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Gender               0
Married              0
Education            0
Self_Employed        0
Property_Area        0
Loan_Status          0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ApplicantIncome    500 non-null    int64
 1   CoapplicantIncome  500 non-null    int64
```

2 LoanAmount 500 non-null int64
3 Loan_Amount_Term 500 non-null int64
4 Credit_History 500 non-null int64
5 Gender 500 non-null object
6 Married 500 non-null object
7 Education 500 non-null object
8 Self_Employed 500 non-null object
9 Property_Area 500 non-null object
10 Loan_Status 500 non-null object
dtypes: int64(5), object(6)
memory usage: 43.1+ KB

Data Info:
None

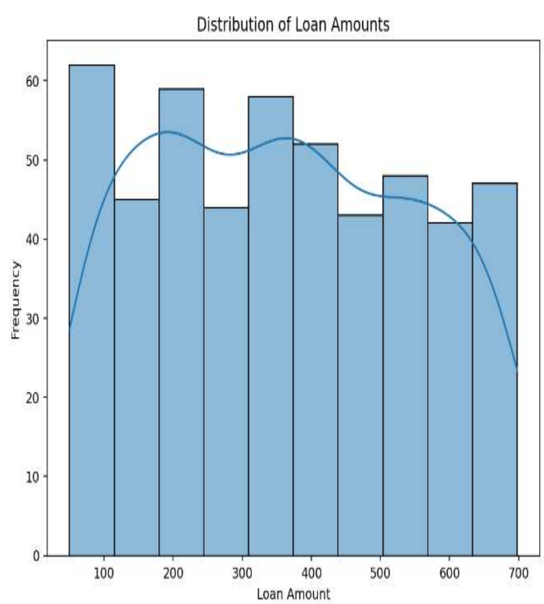
Important

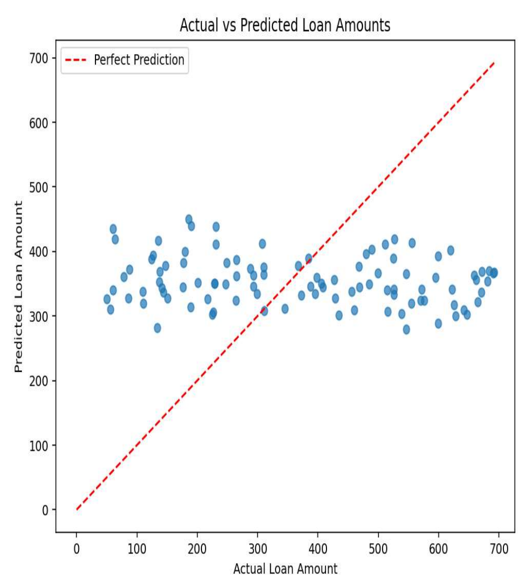
Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you neAed to uncheck "Mute inline plotting" under the options menu of Plots.

Mean Squared Error: 42446.87722969311

Root Mean Squared Error: 206.02639935137708

R^2 Score: -0.10456849387168199





Program of Deep Learning :-

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

import tensorflow as tf
from tensorflow.keras import layers, models

# -----------------------------
# 1. Load dataset
# -----------------------------
df = pd.read_csv("C:/Users/INCHANA/Downloads/loan_data (3).csv")

print("Dataset Shape:", df.shape)
print(df.head())

# -----------------------------
# 2. Features and Target
# -----------------------------
# Example: assuming 'LoanAmount' is the target
X = df.drop(columns=["LoanAmount"])
y = df["LoanAmount"]

# -----------------------------
# 3. Preprocessing
# -----------------------------
# Separate categorical and numeric columns
categorical_cols = X.select_dtypes(include=["object"]).columns.tolist()
numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()

# Column Transformer
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_cols),
```

```
        ("cat", OneHotEncoder(handle_unknown="ignore"), categorical_cols)
    ]
)

# -----------------------------
# 4. Train-test split
# -----------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Fit the transformer
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

print("Transformed Shape:", X_train.shape)

# -----------------------------
# 5. Build Deep Learning Model
# -----------------------------
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1)  # Regression output
])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# -----------------------------
# 6. Train Model
# -----------------------------
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=50,
    batch_size=32,
    verbose=1
)

# -----------------------------
# 7. Evaluate Model
```

```python
# -----------------------------
loss, mae = model.evaluate(X_test, y_test, verbose=0)
print(f"Test MAE: {mae:.2f}")

# -----------------------------
# 8. Plots
# -----------------------------

# Plot training & validation loss
plt.figure(figsize=(8,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.xlabel("Epochs")
plt.ylabel("MSE Loss")
plt.title("Training vs Validation Loss")
plt.legend()
plt.show()

# Plot MAE
plt.figure(figsize=(8,5))
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Val MAE')
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.title("Training vs Validation MAE")
plt.legend()
plt.show()

# -----------------------------
# 9. Predictions vs Actual
# -----------------------------
y_pred = model.predict(X_test).flatten()

plt.figure(figsize=(6,6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.xlabel("Actual Loan Amount")
plt.ylabel("Predicted Loan Amount")
plt.title("Predicted vs Actual Loan Amounts")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')  # 45-degree line
plt.show()
```

Output :-

13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 161176.9062 - mae: 356.1888 - val_loss: 174327.0938 - val_mae: 368.6962

Epoch 2/50

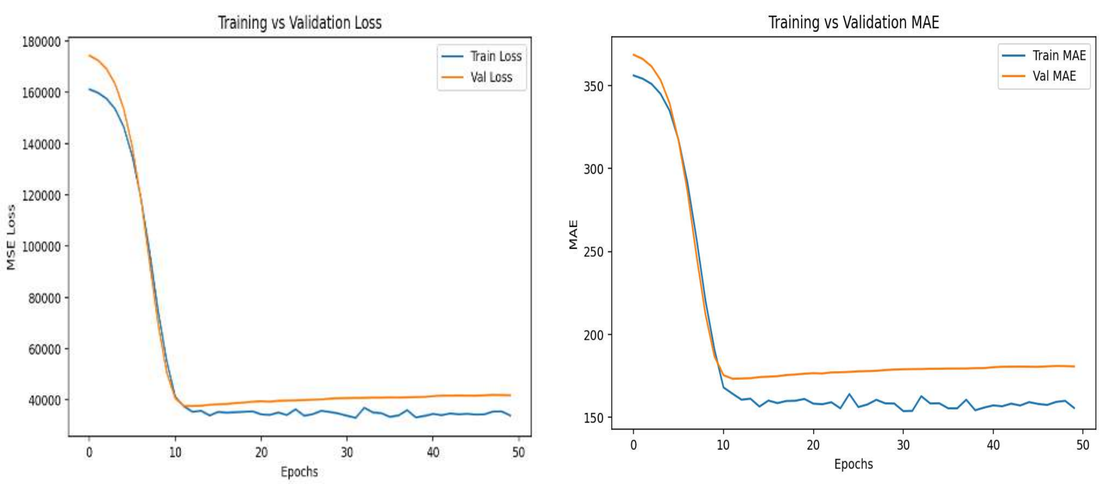
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 159822.7656 - mae: 354.2943 - val_loss: 172481.7188 - val_mae: 366.2214

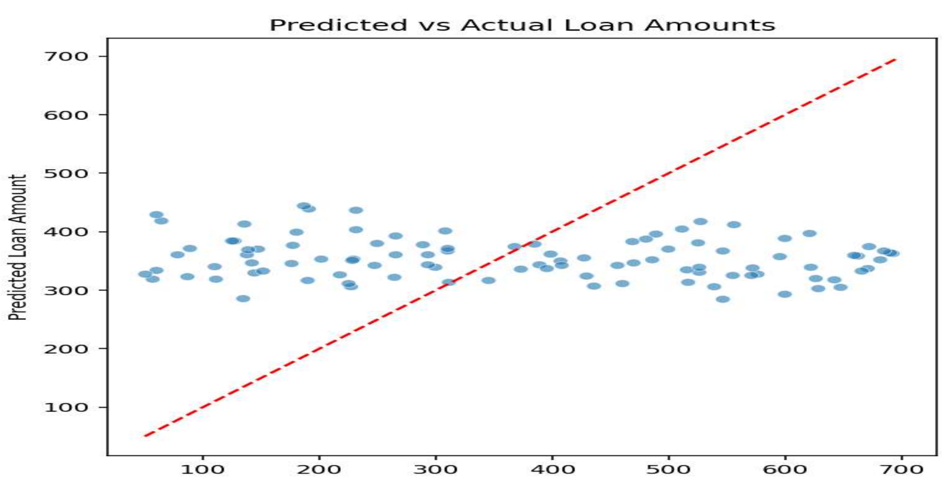
Epoch 3/50

13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 157502.6094 - mae: 351.0763 - val_loss: 169118.1406 - val_mae: 361.6621

Epoch 4/50

13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 153466.7031 - mae: 345.1176 - val_loss: 163192.7188 - val_mae: 353.4714





Classification – Rebuild with deep learning model

Classification (DL): A supervised learning task where a neural network learns to assign input data into one of two or more categories.

- Objective: Predict a discrete label (e.g., fraud vs. non-fraud, spam vs. ham, disease vs. healthy).
- Input: Features (numeric, categorical, text, images, etc.).
- Model: Deep learning architectures (e.g., fully connected networks, CNNs, RNNs, Transformers).

Program of Machine Learning :-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
from sklearn.ensemble import RandomForestClassifier

# -----------------------------
# 1. Load dataset
# -----------------------------
df = pd.read_csv("C:/Users/INCHANA/Downloads/transactions.csv")

print("First rows of dataset:")
print(df.head())

# -----------------------------
# 2. Features & Target
# -----------------------------
X = df.drop(["transaction_id", "is_fraud"], axis=1)
y = df["is_fraud"]

# -----------------------------
# 3. Preprocessing
# -----------------------------
numeric_features = ["amount", "time"]
categorical_features = ["location"]
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cat", OneHotEncoder(), categorical_features)
    ]
)

# -----------------------------
# 4. Train-Test Split
# -----------------------------
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
stratify=y)

# -----------------------------
# 5. Build Model
# -----------------------------
model = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", RandomForestClassifier(random_state=42))
])

model.fit(X_train, y_train)

# -----------------------------
# 6. Evaluation
# -----------------------------
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))

# -----------------------------
# 7. Plots
# -----------------------------

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Fraud", "Fraud"],
yticklabels=["Not Fraud", "Fraud"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_auc_score(y_test, y_proba)))
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

# Feature Importance
rf_model = model.named_steps["classifier"]
preprocessor_fit = model.named_steps["preprocessor"]

# Extract feature names after preprocessing
ohe = preprocessor_fit.named_transformers_["cat"]
feature_names = numeric_features + list(ohe.get_feature_names_out(categorical_features))

importances = rf_model.feature_importances_
feat_imp = pd.Series(importances, index=feature_names).sort_values(ascending=False)

plt.figure(figsize=(8,4))
sns.barplot(x=feat_imp.values, y=feat_imp.index)
plt.title("Feature Importance")
plt.show()

# Class Distribution
plt.figure(figsize=(6,4))
sns.countplot(x="is_fraud", data=df, palette="Set2")
plt.title("Class Distribution")
plt.xticks([0,1], ["Not Fraud", "Fraud"])
plt.show()
```

Output :

First rows of dataset:

	transaction_id	amount	time	location	is_fraud
0	1	120.5	12	New York	0
1	2	560.0	3	Los Angeles	1
2	3	75.0	18	Chicago	0
3	4	3000.0	2	Houston	1

```
4           5   45.0   16     Phoenix       0
```

Classification Report:

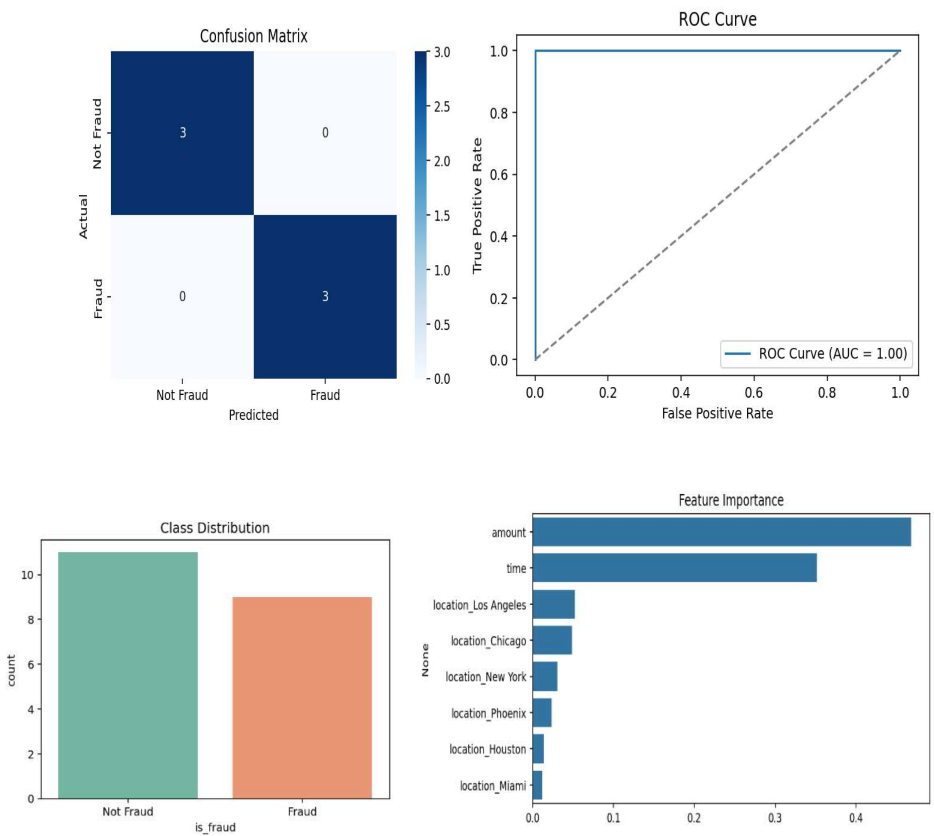
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         3

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

ROC AUC Score: 1.0

c:\users\inchana\fraud detection in transaction mi.py:107: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

sns.countplot(x="is_fraud", data=df, palette="Set2")



Program of Deep Learning:

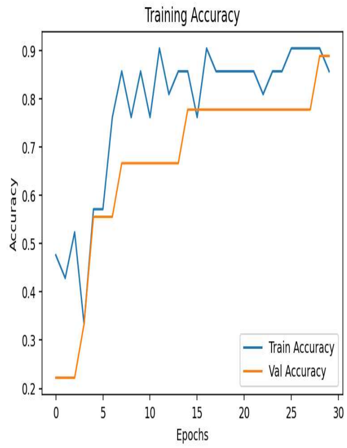
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve


import tensorflow as tf
from tensorflow.keras import layers, models

# -----------------------------
# 1. Load dataset
# -----------------------------
df = pd.read_csv("C:/Users/INCHANA/Downloads/transactions (1).csv")

print("First rows of dataset:")
print(df.head())

# -----------------------------
# 2. Features & Target
# -----------------------------
X = df.drop(["transaction_id", "is_fraud"], axis=1)
y = df["is_fraud"]

# -----------------------------
# 3. Preprocessing
# -----------------------------
numeric_features = ["amount", "time"]
categorical_features = ["location"]

preprocessor = ColumnTransformer(

 transformers=[
```

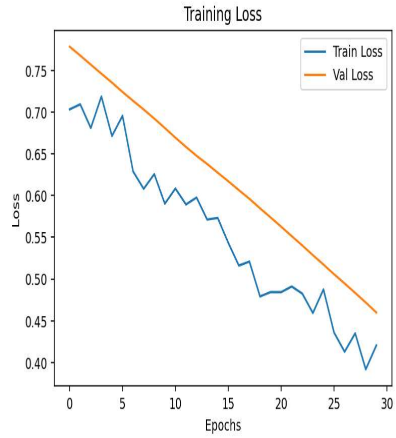
```
        ("num", StandardScaler(), numeric_features),
        ("cat", OneHotEncoder(), categorical_features)
]
)

X_processed = preprocessor.fit_transform(X)

# -----------------------------
# 4. Train-Test Split
# -----------------------------
X_train, X_test, y_train, y_test = train_test_split(X_processed, y, test_size=0.3,
random_state=42, stratify=y)

# -----------------------------
# 5. Build Deep Learning Model
# -----------------------------
model = models.Sequential([
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(32, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")  # binary classification])

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

# -----------------------------
# 6. Train Model
# -----------------------------
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=30,
batch_size=8, verbose=1)

# -----------------------------
# 7. Evaluation
# -----------------------------
y_proba = model.predict(X_test).ravel()
y_pred = (y_proba > 0.5).astype(int)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))

# -----------------------------
# 8. Plots
# -----------------------------

# Training Curves
```

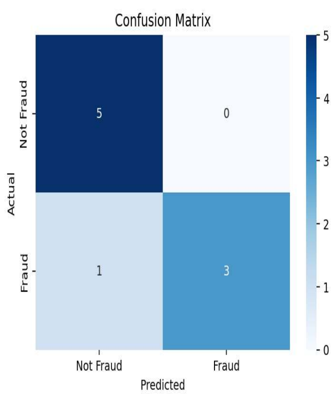
```
plt.figure(figsize=(6,4))
plt.plot(history.history["accuracy"], label="Train Accuracy")
plt.plot(history.history["val_accuracy"], label="Val Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Training Accuracy")
plt.show()

plt.figure(figsize=(6,4))
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.title("Training Loss")
plt.show()

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Fraud", "Fraud"],
yticklabels=["Not Fraud", "Fraud"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_auc_score(y_test, y_proba)))
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

# Class Distribution
plt.figure(figsize=(6,4))
sns.countplot(x="is_fraud", data=df, palette="Set2")
plt.title("Class Distribution")
plt.xticks([0,1], ["Not Fraud", "Fraud"])
plt.show()
```

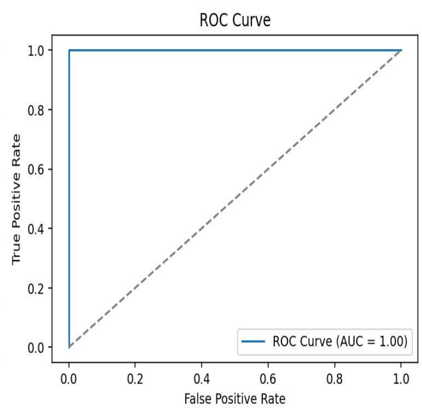
Output :

```
First rows of dataset:
   transaction_id  amount  time     location  is_fraud
0               1   120.5    12     New York         0
1               2   560.0     3  Los Angeles         1
2               3    75.0    18      Chicago         0
3               4  3000.0     2      Houston         1
4               5    45.0    16      Phoenix         0
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 93ms/step - accuracy: 0.4286 - loss:
0.6723 - val_accuracy: 0.5556 - val_loss: 0.6252
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4762 - loss:
0.6890 - val_accuracy: 0.7778 - val_loss: 0.6166
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6190 - loss:
0.6634 - val_accuracy: 0.7778 - val_loss: 0.6077
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6667 - loss:
0.6384 - val_accuracy: 0.7778 - val_loss: 0.5998
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.6667 - loss:
0.6233 - val_accuracy: 0.8889 - val_loss: 0.5917
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.6667 - loss:
0.6419 - val_accuracy: 0.8889 - val_loss: 0.5839
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         4

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9

ROC AUC Score: 1.0
c:\users\inchana\fraud detectin in transaction dl.py:122: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x="is_fraud", data=df, palette="Set2")
```


Training Accuracy
Accuracy
Epochs
Train Accuracy
Val Accuracy


Training Loss
Loss
Epochs
Train Loss
Val Loss


Confusion Matrix
Actual
Not Fraud
Fraud
Predicted
Not Fraud
Fraud
5
0
1
3


ROC Curve
True Positive Rate
False Positive Rate
ROC Curve (AUC = 1.00)


Class Distribution
count
Not Fraud
Fraud
is_fraud

Analyse the performance of ML and 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

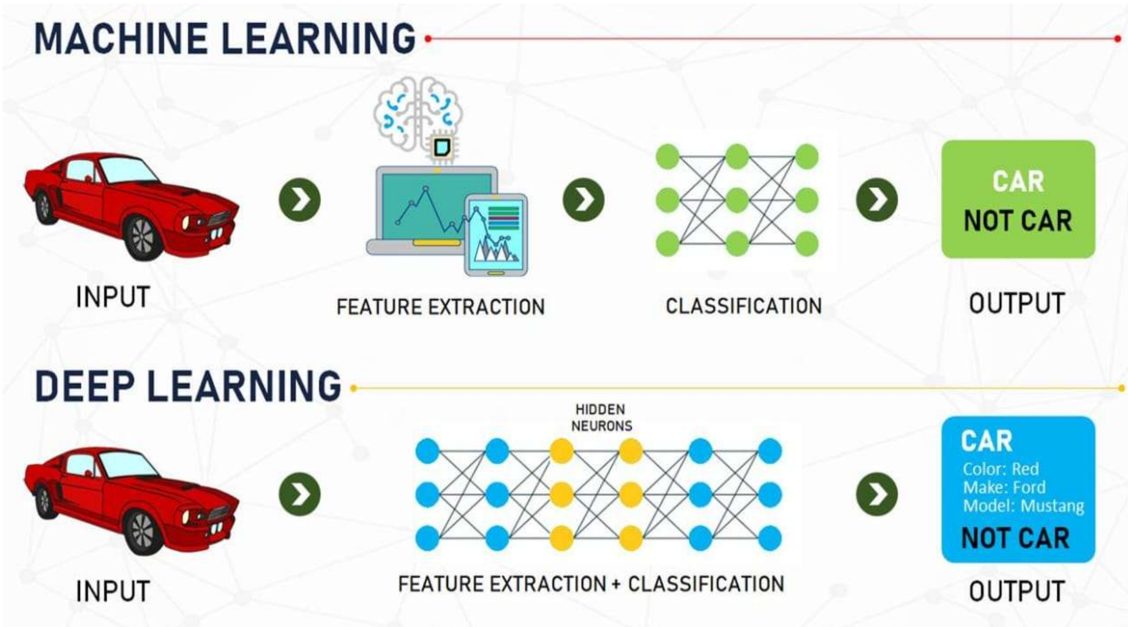
ML: Fast training, less data preprocessing.

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:

The projects on loan amount prediction and fraud detection in transactions highlight the critical role of machine learning and deep learning in the financial sector. Loan amount prediction enables institutions to estimate suitable loan values for applicants by analyzing historical data, customer profiles, and financial indicators, which improves decision-making and reduces credit risks. On the other hand, fraud detection systems use classification techniques to identify suspicious transactions, ensuring the safety and trustworthiness of digital financial operations. Together, these applications not only enhance efficiency and accuracy but also strengthen the reliability and security of financial services, supporting better customer experiences and sustainable business growth.

1.Define Problem Statement

Loan Amount Prediction

- Loan amount prediction is the task of using applicant details such as income, credit score, employment type, loan history, and property value to **predict the loan amount** a person is eligible for. It is generally treated as a **regression problem** where the goal is to estimate a continuous loan amount value, but in some cases, it can also be framed as a **classification problem** by categorizing applicants into predefined loan ranges.
- **Objective** → To build a predictive model that helps financial institutions make **accurate and efficient loan decisions**, reducing risk while improving customer satisfaction.

Fraud Detection in Transaction

- Fraud detection in transactions is the process of analysing user and transaction data (such as amount, location, time, and payment method) to identify whether a transaction is legitimate or fraudulent. This is typically framed as a classification problem, where the goal is to classify each transaction into two classes: *fraud* or *not fraud*.
- Objective → To develop a model that can accurately detect fraudulent transactions in real time, thereby reducing financial losses and improving trust in financial systems.

2.Project Plan

Project 1: Loan Amount Prediction

1.Project Overview

- The project aims to develop a predictive model that estimates the loan amount an applicant is eligible for based on factors such as income, employment type, credit score, property value, and loan history.
- By leveraging Machine Learning (Regression) or Deep Learning models, the system will provide accurate prediction to help financial institutions reduce risk, automate decision-making, and improve customer satisfaction.

2. Scope

- Data collection preprocessing (income, demographics, credit history, etc.).
- Building and training regression/deep learning models for loan amount prediction.

- Model evaluation using metrics like MAE, RMSE, and R^2 score.
- Visualization of results and insights for business decision-making.

- **Out of Scope:**
- Real-time deployment on production systems.
- Integration with existing banking software.
- Handling of live transaction data or external APIs.

3.Deliverables

- Dataset (cleaned and pre-processed loan applicant data).
- Data Analysis (EDA) report with insights on applicant features.
- Predictive Model (ML/DL) for estimating loan amount.
- Performance Evaluation Report (comparisons of models and metrics).
- Visualization Dashboard / plots showing predicted vs. actual loan amounts.
- Final Project Report with methodology, results, and business recommendations.

Project 2: Fraud Detection in Transaction

1. Project Overview

- The project aims to design and implement a fraud detection system that identifies suspicious or fraudulent financial transaction in real-time.
- It leverages **machine learning (ML) and deep learning (DL)** models to detect anomalies and predict fraud based on historical transaction data.
- The solution will enhance security, reduce financial losses, and improve trust in digital payment system.

2.Project Scope

- **In- Scope:**
- Collecting and preprocessing transaction datasets (e.g., transaction amount, time, location, user behavior).
- Applying **ML algorithms** (Logistic Regression, Random Forest,) and **DL models** (Neural Networks, LSTM) for fraud detection.
- Feature engineering (user spending patterns, frequency of transactions, location mismatches).
- Model evaluation using metrics like **accuracy, precision, recall, F1-score, ROC-AUC**.
- Visual dashboards to monitor fraud trends.
- Integration with real-time systems for alert generation.
- Legal/forensic investigation of fraud.
- **Out-of-Scope:**
- Manual fraud recovery processes.
- Deployment into production banking systems (only prototype/demo level).

3. Deliverables

- **Dataset Preparation:**
  A clean and labelled dataset of transaction (fraudulent vs. non-fraudulent).

- **Exploratory Data Analysis (EDA):**
  Visual insights into fraud patterns and transaction anomalies.
- **Fraud Detection Models:**
  Machine learning models trained on transaction data

3.Product Backlog

Project 1: Loan Amount Prediction

Epic 1: Data Collection & Preparation

- **User story 1:** As a data scientist, I want to gather historical loan datasets so that I can build predictive models.
  - Task: collect datasets (loan applications, customer profiles, approved loan amounts).
  - Task: Clean and preprocess missing/duplicate values.
  - Task: Performance feature encoding (categorical numerics).

- **User Story 2:** As a data engineer, I want to normalize and scale fetures so that model training is consistent.
  - Task: Apply Standardization/Scaling.
  - Task: Split data into train/test sets.

Epic 2: Model Development

- **User Story 3:** As a data scientist, I want to train regression models so that I can predict loan amounts.
  - Task: Implement Linear Regression, Decision Trees, Random Forest.
  - Task: Train models and record performance metrics.

- **User Story 4:** As a researcher, I want to test deep learning models so that I can compare performance with ML.
  - Task: Build a Neural Network model.
  - Task: Train & validate using the same dataset.

Epic 3: Model Evaluation

- **User Story 5:** As a product owner, I want compare ML vs DL results so that I can choose the best approach.
  - Task: Evaluate using MAE, RMSE, R^2 Score.
  - Task: Generate performance plots.

Epic 4: Deployment & Reporting

- **User Story 6:** As a stakeholder, I want a simple interface to input applicant details and get a loan amount prediction.
  - Task: Provide documentation & demo.
  - Task: Develop a Flask/streamlet web app.

Project 2: Fraud Detection in Transaction

Epic 1: Data Collection & Preparation

- **User 1:** As a data analyst, I want to collect labelled transaction data so that fraud patterns can be identified.
  - Task: Gather transaction datasets (amount, time, location, device, is fraud).
  - Task: Handle class imbalance (fraud cases << normal cases).
- **User Story 2:** As a data engineer, I want to preprocess and encode transaction data so that models can learn effectively.
  - Task: One-hot encode categorical variables.
  - Task: Normalize numeric features.

Epic 2: Model Development

- **User Story 3:** As a data scientist, I want to build classification models so that fraudulent transaction can be flagged.
  - Task: Train ML models (Logistic Regression, Random Forest,).
  - Task: Compare model accuracy, precision, recall.

- **User Story 4:** As a research, I want to explore deep learning so that sequential/behavioural fraud can be detected.
  - Task: Implement Neural Network.
  - Task: Implement LSTM/Autoencoder for anomalv detection.

Epic 3: Model Evaluation

- **User Story 5:** As a compliance officer, I want to ensure fraud is detected accurately so that false positive are minimized.
  - Task: Evaluate models using Confusion Matrix, ROC-AUC, Precision-Recall.
  - Task: Plot fraud distribution and detection accuracy.

Epic 4: Deployment & Reporting

- **User Story 6:** As a fraud analyst, I want a dashboard to monitor transaction so that fraud alerts are visible in real-time.
  - Task: Build visualization dashboard (Stramlite/PowerBI).
  - Task: Create real-time alerting system (prototype).

4.Git Repository Creation

1. Install Git on your system if not already installed.
2. Create a new repository on GitHubLog in to GitHub.
   Click the “+” button → New repository.
   Enter a repository name, choose public/private, and create it.
3. Open your project folder on your computer.
4. Initialize Git in that folder (this makes it a Git repository).
5. Add your project files to the Git staging area.
6. Commit the changes with a message (for example: “First upload”).
7. Connect your local project to the GitHub repository by adding its URL.
8. Push (upload) your project from your computer to GitHub.
9. Refresh your GitHub repository page — your files should now appear there.

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
```

Commit and push changes:
git add .
git commit -m "Initial project setup"
git push origin main