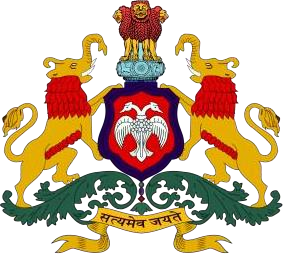
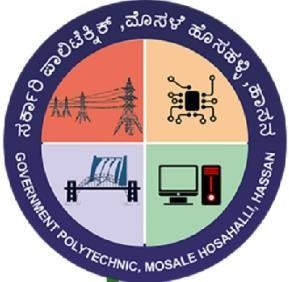
Government of Karnataka

 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: DEEKSHA**

**REG NO: 189CS23013**

**SUBMITTED TO:**

**Praveen Kumar K**

**Selection Grade Lecturer**

**Dept. of CSE**

**Tabel Content**

**SPECIALIZATION PATHWAY**

**ARTFICIAL INTELLIGEANCE & MACHINE LEARNING**

**Problem 1:**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Content** | **Page no** |
| **1** | **Introduction to Regression** | **3** |
| **2** | **Implementation Program of ML&DL** | **4-8** |
| **3** | **Introduction to Classification** | **9** |
| **4** | **Implementation Program of ML&DL** | **10-16** |
| **5** | **Analyse the performance of ML and DL** | **17-18** |

**Problem 2:**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Content** | **Page no** |
| **1** | **Problem Statement** | **19** |
| **2** | **Project plan** | **20-21** |
| **3** | **Product Backlog** | **22-23** |

**Problem 3:**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Content** | **Page No** |
| **1** | **GitHub & Repository** | **24** |

**Regression Problem – Car Price Prediction**

**Introduction**

The automotive industry has witnessed rapid growth in recent years, leading to an increase in the sale and resale of vehicles. One of the most significant challenges in the used car market is determining the fair market value of a car. Car price prediction plays a crucial role in addressing this challenge by using data-driven techniques to estimate the selling price of a vehicle based on various attributes.

Car price prediction involves analysing a wide range of factors such as brand, model, year of manufacture, mileage, fuel type, transmission type, engine capacity, and previous ownership. These features, when combined with historical sales data, help machine learning and deep learning models capture patterns and relationships that influence car prices.

Traditional methods of car valuation often rely on expert judgment or static depreciation rules, which may not reflect real market dynamics. In contrast, predictive modelling provides more accurate, reliable, and scalable solutions. With the help of regression algorithms (like Linear Regression, Random Forest, or Gradient Boosting) and deep learning models (like Neural Networks), one can forecast the approximate value of a car in seconds.

Car price prediction is valuable not only for buyers and sellers but also for automotive companies, dealerships, and financial institutions that provide car loans and insurance. By leveraging machine learning, stakeholders gain insights into pricing trends, customer preferences, and market demand, thereby improving decision-making and transparency in the vehicle resale market.

**Problem Statement**

Car price prediction is a regression problem where the goal is to predict the price of a car based on features like brand, model year, mileage, fuel type, horsepower, etc.

**Machine Learning Approach**

Algorithms used: Linear Regression, Decision Tree Regressor, Random Forest, Gradient Boosting.

**Process:**

Data preprocessing (handling missing values, encoding categorical features).

Feature scaling.

Training on ML models.

Evaluation using metrics like R² score, MAE (Mean Absolute Error), MSE (Mean Squared Error).

**Program to demonstrate Regression Model in Machine Learning Problem**

# Import necessary libraries

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

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# Load CSV dataset

# Replace 'car\_data.csv' with your dataset path

data = pd.read\_csv(r"C:\Users\deeks\Downloads\car\_price\_dataset.csv")

# Preview dataset

print(data.head())

# Assuming the CSV has columns: 'Year', 'Mileage', 'EngineSize', 'Price'

# Features and target

X = data[['Year', 'Mileage', 'Engine\_Size']]

y = data['Price']

# Split into training and testing sets

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

# Scale features

scaler = StandardScaler()

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

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

# Build TensorFlow regression model

model = Sequential([Dense(64, activation='relu',input\_shape=(X\_train\_scaled.shape[1],)),Dense(32, activation='relu'), Dense(1) ])

# Output layer for regression

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

# Train the model

history = model.fit(X\_train\_scaled, y\_train, validation\_split=0.2, epochs=100, batch\_size=16)

# Predict on test data

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

# Plot 1: Actual vs Predicted Prices

plt.figure(figsize=(8,6))

sns.scatterplot(x=y\_test, y=y\_pred.flatten())

plt.xlabel('Actual Price')

plt.ylabel('Predicted Price')

plt.title('Actual vs Predicted Car Prices')

plt.show()

# Plot 2: Training Loss over Epochs

plt.figure(figsize=(8,6))

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

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

plt.xlabel('Epochs')

plt.ylabel('Mean Squared Error')

plt.title('Training and Validation Loss')

plt.legend()

plt.show()

**Output**:

Epoch 1/100

400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 85526984.0000 - mae: 8715.8311 - val\_loss: 80823104.0000 - val\_mae: 8449.9941

Epoch 2/100

400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 61460548.0000 - mae: 7378.3164 - val\_loss: 38673060.0000 - val\_mae: 5919.3467

Epoch 3/100

400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 85526984.0000 - mae: 8715.8311 - val\_loss: 80823104.0000 - val\_mae: 8449.9941

Epoch 4/100

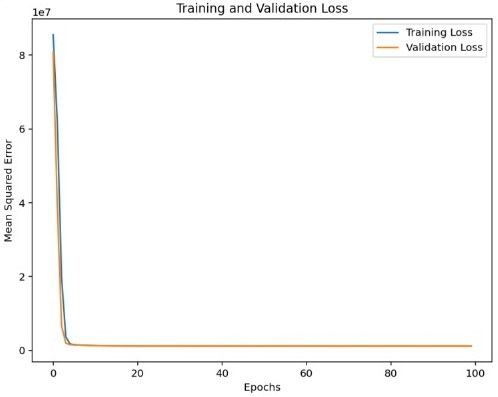
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 61460548.0000 - mae: 7378.3164 - val\_loss: 38673060.0000 - val\_mae: 5919.3467

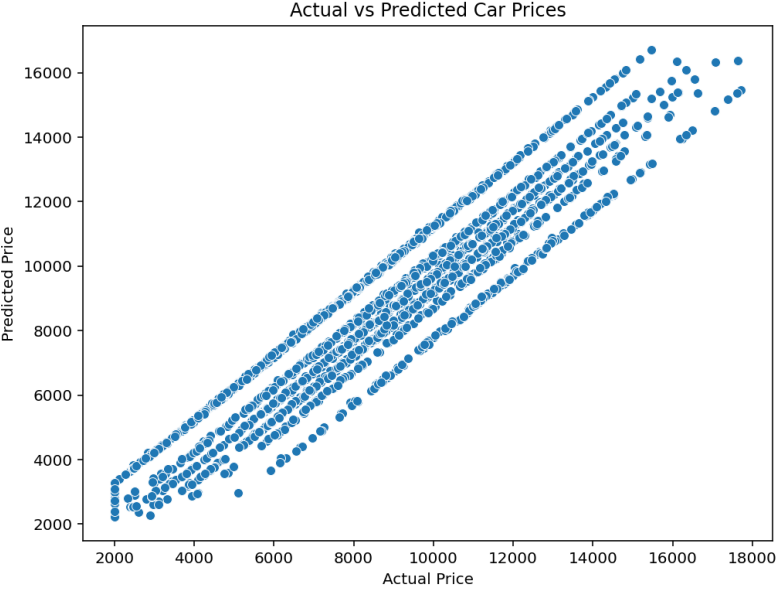
Epoch 5/100

400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 85526984.0000 - mae: 8715.8311 - val\_loss: 80823104.0000 - val\_mae: 8449.9941

Epoch 6/100

400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 61460548.0000 - mae: 7378.3164 - val\_loss: 38673060.0000 - val\_mae: 5919.3467





**Deep Learning Approach:**

Architecture:

Input layer → Hidden dense layers (with ReLU activation) → Output layer (linear activation).

Loss Function: Mean Squared Error (MSE).

Optimizer: Adam optimizer (adaptive learning).

Advantages: Handles complex feature interactions better than traditional ML.

**Program to demonstrate Regression Model in Deep Learning Problem**

# Import required libraries

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

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

# 1. Load the dataset (CSV file)

# Example: "car\_data.csv" must have a column 'price' and features like mileage, year, engine size, etc.

data = pd.read\_csv(r"C:\Users\deeks\Downloads\car\_price\_dataset.csv")

print("Dataset Head:\n", data.head())

print("Dataset Info:\n", data.info())

# 2. Preprocess the data

# Drop rows with missing values (optional, depending on dataset quality)

data = data.dropna()

# Suppose 'price' is the target variable

X = data.drop("Doors", axis=1) # Features

y = data["Doors"] # Target

# Convert categorical features if any

X = pd.get\_dummies(X, drop\_first=True)

#Train-test split

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

# Scale features

scaler = StandardScaler()

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

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

# 3. Build Deep Learning Model

model = keras.Sequential([

layers.Dense(128, activation="relu", input\_shape=(X\_train.shape[1],)),

layers.Dense(64, activation="relu"),

layers.Dense(32, activation="relu"),

layers.Dense(1) # Regression output (no activation)])

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

# 4. Train the Model

history = model.fit(X\_train, y\_train, validation\_split=0.2, epochs=100,

batch\_size=32, verbose=1)

# 5. Evaluate the Model

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

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"Model Performance:")

print(f"RMSE: {rmse}")

print(f"R² Score: {r2}")

# 6. Plot Training History

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

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

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

plt.xlabel("Epochs")

plt.ylabel("MSE Loss")

plt.legend()

plt.title("Training vs Validation Loss")

**Output:**

[5 rows x 10 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -------- ------ -----

0 Brand 10000 non-null object

1 Model 10000 non-null object

2 Year 10000 non-null Int64

3 Engine\_Size 10000 non-null float64

4 Fuel\_Type 10000 non-null object

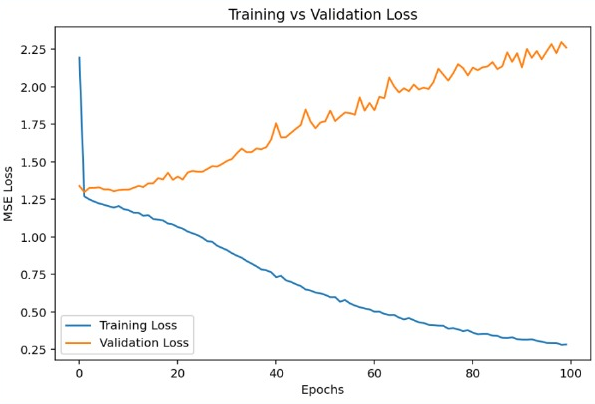
5 Transmission 10000 non-null object

dtypes: float64(1), int64(5), object(4)

Model Performance:

RMSE: 1.5241097556463805

R² Score: -0.9053386449813843



**Classification Problem – Credit Card Fraud Detection**

**Introduction**

With the rapid growth of online transactions and digital payments, credit cards have become one of the most widely used modes of payment worldwide. However, this convenience also brings a major challenge: credit card fraud. Fraudulent transactions cause billions of dollars in financial losses each year, affecting banks, merchants, and customers. Detecting such fraud in real time is a critical task to ensure financial security and trust in digital payment systems.

Credit card fraud detection involves identifying unusual spending patterns and suspicious activities that differ from a customer’s normal behavior. Traditional manual methods are not sufficient due to the large volume of transactions occurring every second. This is where machine learning and artificial intelligence play an important role. By analyzing historical transaction data, these models can classify transactions as genuine or fraudulent based on features such as transaction amount, location, time, and purchase behavior.

**Problem Statement**

Credit card fraud detection is a binary classification problem where transactions are labeled as fraud (1) or non-fraud (0).

The Credit Card Fraud Detection problem can be framed as a binary classification task in machine learning, where each transaction needs to be classified as either:

* Legitimate (non-fraudulent), or
* Fraudulent (fraud).

**Machine Learning Approach**

Algorithms used: Logistic Regression, Decision Trees, Random Forest, XGBoost.

Process:

Deal with class imbalance (fraud cases are rare).

Train ML models.

Evaluate using Accuracy, Precision, Recall, F1-Score, AUC-ROC.

**Program to demonstrate Classification Model in Machine Learning Problem**

# Credit Card Fraud Detection using Machine Learning (TensorFlow)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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, roc\_auc\_score

# Load dataset

# Dataset available: https://www.kaggle.com/mlg-ulb/creditcardfraud

df = pd.read\_csv(r"C:\Users\deeks\Downloads\creditcard.csv")

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

print(df["Class"].value\_counts())

# Features and target

X = df.drop("Class", axis=1)

y = df["Class"]

# Train-test split

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

# Scale features

scaler = StandardScaler()

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

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

# Build Deep Learning Model

model = tf.keras.Sequential([

tf.keras.layers.Dense(32, input\_dim=X\_train.shape[1], activation="relu"),

tf.keras.layers.Dropout(0.3),

tf.keras.layers.Dense(16, activation="relu"),

tf.keras.layers.Dropout(0.3),

tf.keras.layers.Dense(1, activation="sigmoid") # Binary classification])

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

# Train the model

history = model.fit(X\_train, y\_train,validation\_split=0.2,epochs=10,

batch\_size=2048,verbose=1)

# Evaluate on test set

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

y\_pred\_classes = (y\_pred > 0.5).astype(int)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred\_classes))

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred\_classes))

print("ROC-AUC Score:", roc\_auc\_score(y\_test, y\_pred))

# Plot training history

plt.figure(figsize=(12,4))

plt.subplot(1,2,1)

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

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

plt.legend()

plt.title("Model Loss")

plt.subplot(1,2,2)

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

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

plt.legend()

plt.title("Model Accuracy")

plt.show()

**Output:**

Dataset shape: (284807, 31)

Class

0 284315

1 492

Name: count, dtype: int64

Epoch 1/10

90/90 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9967 - loss: 0.0684 - val\_accuracy: 0.9982 - val\_loss: 0.0154

Epoch 2/10

90/90 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9980 - loss: 0.0299 - val\_accuracy: 0.9984 - val\_loss: 0.0077

Epoch 3/10

90/90 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9984 - loss: 0.0189 - val\_accuracy: 0.9989 - val\_loss: 0.0056

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 56864

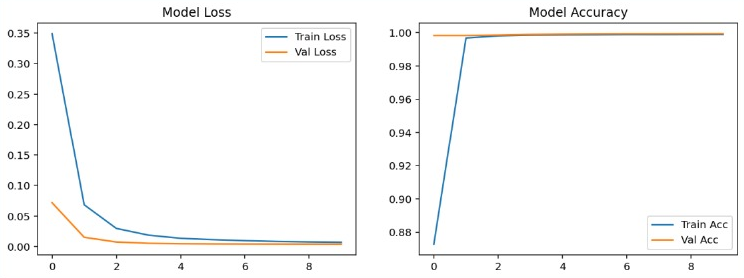
1 0.83 0.72 56982

accuracy 1.00 56962

macro avg 0.91 0.86 0.89 56962

weighted avg 1.00 1.00 1.00 56962

ROC-AUC Score: 0.9616982661100456



**Deep Learning Approach**

Architecture:

Input layer → Hidden dense layers (ReLU + Dropout for regularization) → Output layer (Sigmoid activation).

Loss Function: Binary Cross-Entropy.

Optimizer: Adam optimizer.

Techniques: SMOTE (Synthetic Minority Over-sampling), Class Weights.

**Program to demonstrate Classification Model in Deep Learning Problem**

# Credit Card Fraud Detection using Deep Learning (Keras)

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.metrics import classification\_report, confusion\_matrix, roc\_auc\_score, roc\_curve, precision\_recall\_curve, average\_precision\_score

import tensorflow as tf

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.callbacks import EarlyStopping

# Load Dataset

# Download from: https://www.kaggle.com/mlg-ulb/creditcardfraud

df = pd.read\_csv(r"C:\Users\deeks\Downloads\creditcard.csv") # ensure the file is in same folder

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

print(df['Class'].value\_counts())

# Preprocessing

X = df.drop("Class", axis=1)

y = df["Class"]

# Scale only 'Amount' and 'Time' (others are PCA-transformed already)

scaler = StandardScaler()

X[['Time','Amount']] = scaler.fit\_transform(X[['Time','Amount']])

# Train/test split (stratified to keep fraud ratio consistent)

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

print("Train shape:", X\_train.shape, " Test shape:", X\_test.shape)

# Build Deep Learning Model

def build\_model(input\_dim):

model = Sequential([Dense(128, activation='relu', input\_dim=input\_dim),

BatchNormalization(),Dropout(0.5),Dense(64, activation='relu'),

BatchNormalization(),Dropout(0.3),Dense(32, activation='relu'),

Dense(1, activation='sigmoid') # binary classification])

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

# return model

model = build\_model(X\_train.shape[1])

model.summary()

# Training

es = EarlyStopping(monitor='val\_loss', patience=6, restore\_best\_weights=True)

history = model.fit( X\_train, y\_train, validation\_split=0.1,epochs=50, batch\_size=2048, callbacks=[es],verbose=2)

#Evaluation

y\_probs = model.predict(X\_test).ravel()

y\_pred = (y\_probs >= 0.5).astype(int)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred, digits=4))

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

print("ROC AUC:", roc\_auc\_score(y\_test, y\_probs))

print("Average Precision (PR AUC):", average\_precision\_score(y\_test, y\_probs))

# Plots

# Training history

plt.figure(figsize=(6,4))

plt.plot(history.history['loss'], label='train\_loss')

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

plt.title("Training vs Validation Loss")

plt.xlabel("Epoch")

plt.ylabel("Loss")

plt.legend()

plt.show()

# Precision-Recall Curve

precision, recall, \_ = precision\_recall\_curve(y\_test, y\_probs)

plt.figure(figsize=(6,5))

plt.plot(recall, precision, label=f"AP={average\_precision\_score(y\_test,y\_probs):.4f}")

plt.xlabel("Recall")

plt.ylabel("Precision")

plt.title("Precision-Recall Curve")

plt.legend()

plt.show()

**Output:**

Name: count, dtype: int64

Train shape: (227845, 30) Test shape: (56962, 30)

Model: "sequential\_2"

|  |  |  |
| --- | --- | --- |
| Layer(type) | Output Shape | Param# |
| dense (Dense) | (None,128) | 3,968 |
| batch\_normalization  (BatchNormalization) | (None,128) | 512 |
| dropout (Dropout) | (None,128) | 0 |
| dense (Dense) | (None,64) | 8,256 |
| batch\_normalization\_1  (BatchNormalization) | (None,64) | 256 |
| dropout\_1 (Dropout) | (None,64) | 0 |
| dense\_1(Dense) | (None,32) | 2,080 |
| dense\_2Dense) | (None,1) | 33 |

ROC AUC: 0.9818684824802177

Average Precision (PR AUC): 0.8517949599243462

A graph of a curve and a graph of a curve

AI-generated content may be incorrect.

**Analyse the performance of ML and DL**

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

🌐 **Machine Learning (ML)**

Machine Learning is a subset of Artificial Intelligence (AI) that focuses on building algorithms that allow systems to learn from data and improve performance over time without being explicitly programmed.

**🤖 Deep Learning (DL)**

Deep Learning is a subset of Machine Learning inspired by the structure of the human brain. It uses Artificial Neural Networks (ANNs) with multiple hidden layers to automatically extract and learn complex patterns from large datasets.

**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**

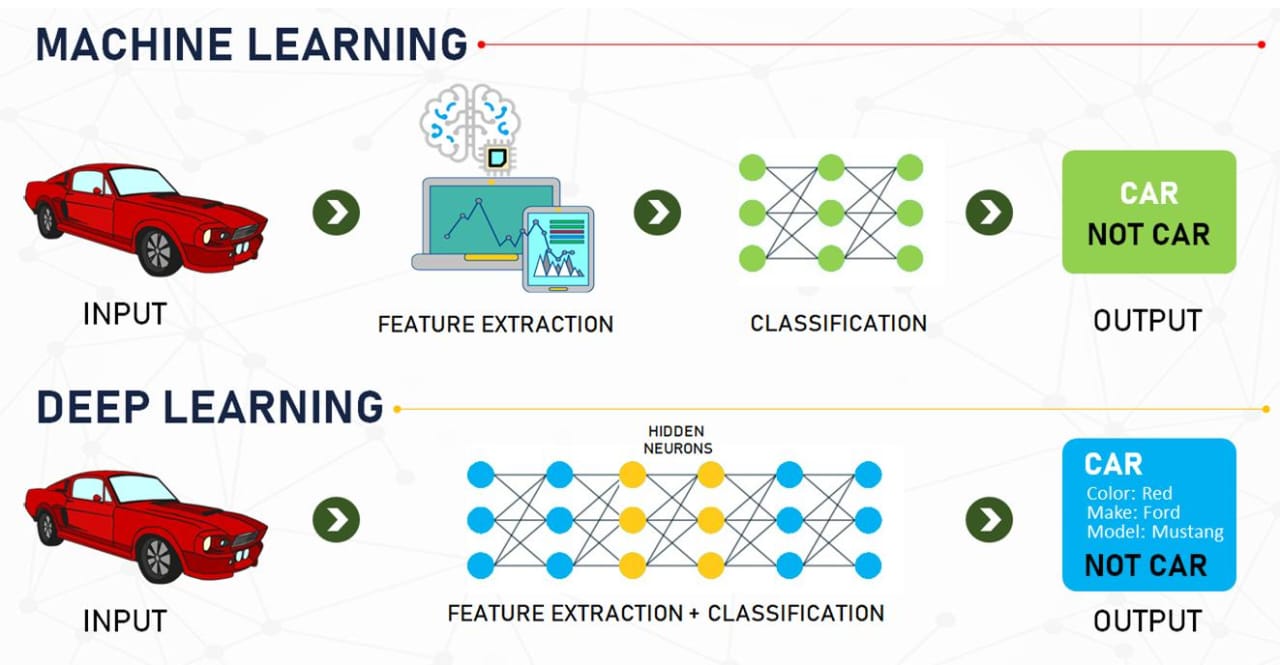
ML: Fast training, less data preprocessing.

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) and Deep Learning (DL) have emerged as two of the most powerful technologies driving innovation in artificial intelligence. ML provides algorithms that learn from data to make predictions or decisions without being explicitly programmed, while DL, a subset of ML, uses artificial neural networks with multiple layers to automatically extract complex patterns and features from large datasets.

ML is highly effective for tasks where data is limited, models need interpretability, and training time must be efficient. In contrast, DL excels in handling massive amounts of unstructured data such as images, videos, and natural language, enabling breakthroughs in fields like computer vision, speech recognition, and healthcare diagnostics.

Both approaches have their own advantages and challenges: ML offers simplicity, speed, and interpretability, while DL offers high accuracy, scalability, and automation of feature engineering at the cost of requiring vast computational resources and large datasets.

In summary, ML and DL complement each other: ML provides strong baseline models for structured problems, and DL pushes the boundaries of what AI systems can achieve in complex, real-world scenarios. As data availability.

**1.Define Problem Statement**

🚗 **Car Price Prediction – Introduction**

The automobile industry is one of the fastest-growing markets worldwide, with millions of cars being bought and sold every year. A key challenge in this ecosystem is predicting the fair resale price of a car. Car prices depend on a wide variety of factors including brand, model, manufacturing year, mileage, fuel type, engine capacity, transmission, number of previous owners, and market conditions.

Traditionally, buyers and sellers relied on manual evaluation, market knowledge, or dealership recommendations, which can often lead to subjective and inconsistent pricing. This creates problems such as overpricing (making the car harder to sell) or underpricing (causing financial loss to the seller).

Machine learning techniques provide a data-driven solution by analyzing historical car sales data and identifying patterns that influence car pricing. With regression models and deep learning approaches, we can build a system that predicts the approximate value of a car, ensuring transparency, fairness, and efficiency in the automobile resale market.

**💳 Credit Card Fraud Detection – Introduction**

The rapid growth of online banking and digital transactions has made financial systems more efficient, but it has also led to an increase in fraudulent activities. Credit card fraud, in particular, has become a global issue causing billions of dollars in annual losses for banks, merchants, and customers.

Fraudulent transactions often resemble genuine transactions, making them difficult to detect using traditional rule-based methods. Moreover, fraudsters continuously adapt their strategies, which makes static detection systems outdated very quickly.

Machine learning and deep learning techniques offer a promising solution by analyzing large-scale transaction data to detect unusual or suspicious patterns in real time. Using features such as transaction amount, location, time, frequency, and customer behavior, a classification model can be trained to differentiate between legitimate and fraudulent transactions.

By implementing such systems, banks and financial institutions can reduce financial losses, protect customer trust, and improve the security of the digital payment ecosystem.

**2.Project Plan**

**Introduction**

**Car Price Prediction:** The goal is to build a regression model that predicts the selling price of a car based on features like brand, model, mileage, year, fuel type, and other attributes.

**Credit Card Fraud Detection:** The aim is to develop a classification model to detect fraudulent transactions in financial datasets, distinguishing between legitimate and fraudulent activities.

**2. Objectives**

1. Collect and preprocess datasets for both projects.

2. Build Machine Learning (ML) and Deep Learning (DL) models.

3. Compare model performance using suitable evaluation metrics.

4. Deploy models for real-world usability (optional).

**3. Scope**

**Car Price Prediction:**

Predict resale values of used cars.

Support dealerships and individuals in decision-making.

**Credit Card Fraud Detection:**

Detect fraudulent transactions in real time.

Reduce financial losses for banks and customers.

**Project 1: Car Price Prediction**

**1. Project Overview**

* **Objective:** Predict the price of used cars based on features such as make, model, year, mileage, fuel type, etc., using machine learning regression techniques.
* **Outcome:** A model that can estimate car prices with high accuracy and a user-friendly interface or API.

**2. Scope**

* **In Scope:**
* Data collection from online sources (Kaggle datasets, APIs)
* Data cleaning and preprocessing
* Exploratory Data Analysis (EDA)
* Feature selection and engineering
* Model selection: Linear Regression, Decision Tree, Random Forest, XGBoost
* Model evaluation and hyperparameter tuning
* Deployment (optional: Streamlit app or Flask API)
* **Out of Scope:**
* Real-time price updates
* Integration with commercial marketplaces

**3. Deliverables**

* Cleaned dataset
* EDA report with insights
* Machine learning models and evaluation metrics
* Prediction interface or API
* Documentation

**Project 2: Credit Card Fraud Detection**

**1. Project Overview**

* **Objective:** Detect fraudulent credit card transactions using historical transaction data.
* **Outcome:** A classification model that flags fraudulent transactions with high precision and recall.

**2. Scope**

* **In Scope:**
* Data collection (e.g., Kaggle credit card dataset)
* Model selection: Logistic Regression, Random Forest, XGBoost, Neural Networks
* Model evaluation with precision, recall, F1-score, ROC-AUC
* Optional: Deployment of fraud detection API
* **Out of Scope:**
* Real-time transaction monitoring
* Integration with banking systems

**3. Deliverables**

* Processed dataset
* Data analysis report
* Classification model with evaluation metrics
* Fraud detection dashboard or API
* Documentation

**3.Product Backlog**

**Introduction to Product Backlog**

A Product Backlog is a prioritized list of all the tasks, features, enhancements, bug fixes, and technical work that need to be completed for a project or product. It acts as the single source of truth for everything that is required to improve the product and deliver value to stakeholders.

The backlog is dynamic and continuously refined; items may be added, removed, or reprioritized based on changing business needs, customer feedback, or market conditions.

In Agile and Scrum methodologies, the product backlog is maintained by the Product Owner and serves as the foundation for planning iterations (sprints). Each backlog item, often called a User Story or Product Backlog Item (PBI), should be clear enough so that the development team can understand what needs to be done and why it is valuable.

**A well-structured backlog ensures:**

* Transparency in project progress.
* Alignment between business goals and development efforts.
* Flexibility to adapt to changes.
* Continuous delivery of value to users.

**Project 1: Car Price Prediction**

**Epic 1: Data Collection & Preprocessing**

* **User Story 1:** As a data scientist, I want to collect car datasets so that I can train a prediction model.
* Task: Identify reliable datasets (Kaggle, APIs).
* Task: Import datasets into project.
* **User Story 2:** As a developer, I want to preprocess data so that it is clean and ready for modeling.
* Task: Handle missing values.
* Task: Encode categorical features (brand, fuel type, transmission).
* Task: Normalize/standardize numerical features.

**Epic 2: Exploratory Data Analysis (EDA)**

* **User Story 3:** As a data analyst, I want to visualize data to understand feature relationships.
* Task: Plot distributions of car prices.
* Task: Correlation analysis of features.
* Task: Detect and remove outliers.

**Epic 3: Model Development**

* **User Story 4:** As a data scientist, I want to build baseline models to predict car prices.
* Task: Train Linear Regression model.
* Task: Evaluate with RMSE, MAE, R².

**Epic 4: Documentation & Testing**

* **User Story 5:** As a developer, I want to document my work so that others can understand and use it.
* Task: Write project documentation.
* Task: Perform unit testing and validation.

**Project 2: Credit Card Fraud Detection**

**Epic 1: Data Collection & Preprocessing**

* **User Story 1:** As a data scientist, I want to gather transaction datasets to detect fraud.
* Task: Import Kaggle dataset.
* Task: Check data imbalance.
* **User Story 2:** As a developer, I want to preprocess the dataset for training.
* Task: Normalize/scale features.
* Task: Handle imbalanced dataset (SMOTE/undersampling).

**Epic 2: Exploratory Data Analysis (EDA)**

* **User Story 3:** As a data analyst, I want to understand patterns in fraudulent vs. non-fraudulent transactions.
* Task: Plot transaction amount distributions.
* Task: Correlation heatmap of features.

**Epic 3: Model Development**

* **User Story 4:** As a data scientist, I want to build a baseline classifier.
* Task: Train Logistic Regression.
* Task: Evaluate with Precision, Recall, F1-score.
* **User Story 5:** As a developer, I want to improve detection accuracy using advanced models.
* Task: Train Random Forest, XGBoost, Neural Network.
* Task: Apply cross-validation.

**Epic 4: Documentation & Testing**

* **User Story 6:** As a developer, I want to document and test the fraud detection system.
* Task: Write model usage guide Create test cases for new transactions.

**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