UNIT 1: INTRODUCTION TO MACHINE LEARNING

Diploma-Level Study Material (Syllabus-Aligned)

📌 QUICK OVERVIEW

Unit 1 has 4 topics:

1. **1.1 Basics of ML** - Definition, Traditional Prog vs ML, Role in AI/DS
2. **1.2 Types of ML** - Supervised, Unsupervised, Reinforcement Learning
3. **1.3 Applications & Challenges** - Real-world use cases, challenges
4. **1.4 Python for ML** - Python basics, libraries, simple scripts

Total Learning Time: 8 hours | **Exam Weight:** ~25% of first semester

MindforgeAi Publications

Technical Publication Series

Chatake Innoworks Pvt. Ltd.

Authored by: Akash Shivadas Chatake

1.1 BASICS OF MACHINE LEARNING

1.1.1 What is Machine Learning?

Simple Definition

Machine Learning is a method of making computers learn patterns from data and improve their performance automatically, **without being explicitly programmed for every scenario**.

Traditional Programming vs Machine Learning

Aspect	Traditional Programming	Machine Learning
How it works	Programmer writes explicit rules → Computer executes code	Programmer shows examples → Computer learns rules automatically
Input to system	Input data	Input data + expected output (examples)
Process	Code execution	Learning from data
Output	Results based on coded rules	Learned model that makes predictions
Can it adapt?	No - needs reprogramming	Yes - learns from new data
Best for	Well-defined problems (e.g., banking, billing)	Pattern recognition (e.g., face detection, recommendations)
Example	If age > 18, allow entry	Learn from past: “Who usually enters?”

Why Machine Learning Matters?

Three key reasons:

1. **Too Complex to Code**
   - Can't write rules for face recognition
   - Can't define rules for movie recommendations
   - Patterns are too complex for humans to code
2. **Data is Abundant**
   - Modern world generates massive amounts of data
   - ML uses this data to discover patterns
   - More data → Better learning
3. **Systems Need to Adapt**
   - Rules that work today may fail tomorrow
   - ML models learn and update automatically
   - Can handle changing conditions

Real-World Example: Email Spam Filter

Traditional Approach:

```
If (contains "free money") → Mark as SPAM
    If (contains "winner") → Mark as SPAM
        ... (write 1000s of rules manually)
    Problem: Spammers change text, all rules break
```

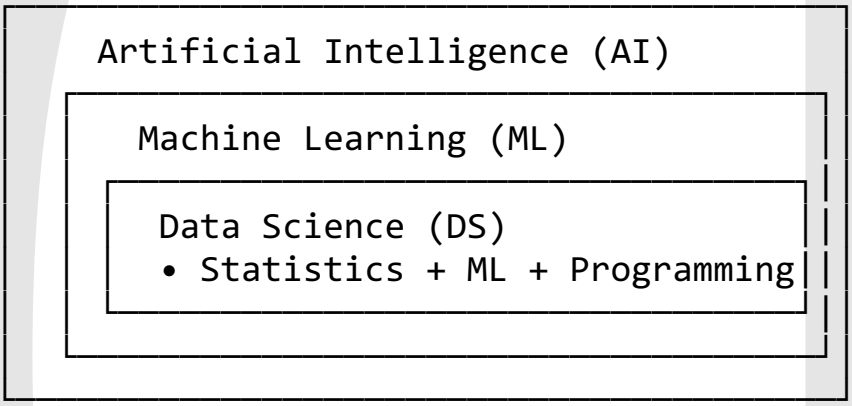
ML Approach:

```
Show system: 10,000 spam emails + 10,000 legitimate emails
System learns patterns: "Legitimate emails rarely say 'free money'"
    New email arrives → System predicts SPAM/NOT SPAM automatically
        Spammers change text → System adapts from new data
```

Role of ML in AI and Data Science

Artificial Intelligence (AI) - Broad field of creating intelligent machines - Includes robotics, planning, decision-making, etc. - **Machine Learning is a tool within AI**

Data Science - Uses data to extract insights and make decisions - Combines statistics, programming, domain knowledge - **Machine Learning is a key component of Data Science**



1.1.2 Key Components of ML System

Every ML system has **three essential parts:**

1. DATA

- **What:** Raw information/facts
- **Examples:** Student marks, images, customer records
- **Quality > Quantity:** One clean dataset > Many messy datasets
- **Types:** Numbers, text, images, video, audio

2. MODEL

- **What:** Mathematical structure that learns patterns
- **Analogy:** Empty vessel that "learns" from data
- **Examples:** Decision Tree, Linear Regression, Neural Network
- **Role:** Captures patterns to make predictions

3. LEARNING ALGORITHM

- **What:** Step-by-step procedure to teach the model
- **Process:** Show data → Calculate error → Improve model → Repeat
- **Examples:** Gradient Descent, k-means, Decision Tree Algorithm

How they work together:

```
DATA → [LEARNING ALGORITHM] → MODEL → Can predict new data
```

1.1.3 Three Key Processes in ML

1. TRAINING

- **Purpose:** Teach the model
- **Data Used:** Training set (60-80% of data)
- **Process:** Show model examples with correct answers
- **What model does:** Adjusts internal parameters to minimize errors
- **Duration:** Hours to weeks (depending on data size)

2. VALIDATION

- **Purpose:** Check if model is learning well
- **Data Used:** Validation set (10-20% of data)
- **Process:** Test model on data it hasn't seen
- **What it checks:** Is model generalizing or just memorizing?
- **When:** During training to tune hyperparameters

3. TESTING

- **Purpose:** Final performance evaluation
- **Data Used:** Test set (20% of data, never seen during training)
- **Process:** Evaluate on completely new data
- **What it tells us:** How model will perform in real world
- **When:** After training is complete

Exam Summary for 1.1

Key Points to Remember:

- ML ≠ AI (ML is within AI)
- ML learns patterns; Traditional programming executes coded rules
- ML system needs: Data, Model, Learning Algorithm
- Three phases: Training (learn), Validation (check), Testing (evaluate)
- Data quality matters more than quantity

1.2 TYPES OF MACHINE LEARNING

1.2.1 SUPERVISED LEARNING

Definition

Learning from **labeled examples** where both input AND correct output are provided.

How It Works

```
Input Data (Features)
        ↓
    Model
        ↓
    Prediction
        ↓
    Compare with ACTUAL LABEL
        ↓
    Calculate Error
        ↓
    Adjust Model
        ↓
    Repeat until accurate
```

Requirements

- Labeled data (input-output pairs)
- Clear definition of what to predict
- Usually, 100s to 1000s of examples

Two Main Tasks

A. CLASSIFICATION - Predict Categories

B. REGRESSION - Predict Numbers

Two Main Tasks

A. CLASSIFICATION - Predict Categories

Definition: Predict which category an item belongs to

Examples: -

Email:	**Spam or Not Spam?**	2 categories
Disease:	**Present or Absent?**	2 categories
Handwritten **digit**:	**0, 1, 2, ..., 9?**	10 categories
Student performance:	**Pass or Fail?**	2 categories
Fruit type:	**Apple, Orange, Banana?**	3 categories

Output: Discrete category (not continuous number)

How model learns:

```
Training data:
  Image of 3 → Label: "Three"
  Image of 5 → Label: "Five"
  Image of 8 → Label: "Eight"

Model learns patterns of each digit shape

New image → Model → "This looks like a Five" (prediction)
```

B. REGRESSION - Predict Numbers

Definition: Predict a continuous numerical value

Examples: -

House **price**	(e.g., ₹45,00,000)
Temperature tomorrow	(e.g., 28°C)
Student **GPA**	(e.g., 3.8)
Stock **price** next week	(e.g., ₹1,250)
Age of person from photo	(e.g., 35 years)

Output: Any continuous number within range

How model learns:

```
Training data:
  House 100 sq m, 2 rooms → Price: ₹30 lakhs
  House 200 sq m, 4 rooms → Price: ₹60 lakhs
  House 150 sq m, 3 rooms → Price: ₹45 lakhs

Model learns: "Larger house = higher price"

New house 180 sq m, 3 rooms → Model → Predicts: ₹52 lakhs
```

Advantages of Supervised Learning

✅ Clear feedback (know if prediction is right/wrong)
✅ Usually high accuracy
✅ Well-established methods

Disadvantages of Supervised Learning

❌ Requires labeled data (expensive and time-consuming)
❌ Can't use unlabeled data (most real-world data is unlabeled)
❌ Models can memorize instead of learning (overfitting)

Real-World Applications

- Medical diagnosis (labeled: sick/healthy patients)
- Credit approval (labeled: approved/rejected loans)
- Spam detection (labeled: spam/legitimate emails)
- Student grade prediction (labeled: student data + grades)

1.2.2 UNSUPERVISED LEARNING

Definition

Finding hidden patterns and structure in data **WITHOUT labels** or expected outputs.

How It Works

```
Raw Data (No Labels)
        ↓
    Algorithm
        ↓
    Discovers Patterns
        ↓
    Groups or Structures
        ↓
    Human Interprets Results
```

Requirements

- ✅ Raw data (no labels needed)
- ✅ Large amounts of data
- ✅ No predetermined answers

Two Main Tasks

A. ***CLUSTERING - Group Similar Items***
B. ***DIMENSIONALITY REDUCTION - Simplify Data***

Two Main Tasks

A. CLUSTERING - Group Similar Items

Definition: Group similar items together

Examples: - **Customer Segmentation:**

Group customers by shopping behavior

Group A: "Budget buyers" - prefer discounts

Group B: "Premium buyers" - prefer quality

Group C: "Occasional buyers"

- **Document Grouping:** Categorize news articles
  - Sports cluster: "Cricket", "Football", "Tennis"
  - Political cluster: "Government", "Elections", "Laws"
  - Business cluster: "Markets", "Stocks", "Companies"

- **Image Clustering:** Group similar photos
  - Cluster 1: All dog photos
  - Cluster 2: All cat photos
  - Cluster 3: All bird photos

How it works:

```
Algorithm measures similarity between items
                Groups similar items together
                        Results in "clusters"
        Human then interprets what each cluster means
```

Challenge: Algorithm doesn't know what clusters mean - human must interpret

B. DIMENSIONALITY REDUCTION - Simplify Data

Definition: Reduce number of features while keeping important information

Simple Analogy: Compressing a photo - smaller file, same content

Examples: -

Sensor data: 1000 sensors → Compress to 10 important values

Visualizing data: High-dimension data → 2D plot for visualization

Removing noise: Keep important patterns, remove noise

Data compression: Store less data, faster processing

Why useful: -

- Faster model training
- Easier to visualize
- Less storage needed
- Removes irrelevant features

Advantages of Unsupervised Learning

- ✅ No need for labels (huge advantage!)
- ✅ Works with abundant unlabeled data
- ✅ Discovers unexpected patterns
- ✅ Cheaper than supervised learning

Disadvantages of Unsupervised Learning

- ❌ Hard to validate results (no “correct” answer)
- ❌ May find meaningless patterns
- ❌ Requires human interpretation
- ❌ Results can be unpredictable

Real-World Applications

- Customer segmentation for targeted marketing
- Document recommendation systems
- Gene clustering in biological research
- Anomaly detection (finding unusual/suspicious items)
- Social network analysis (finding communities)

1.2.3 REINFORCEMENT LEARNING

Definition

Learning through **interaction and rewards** - agent learns best actions by trial-and-error.

How It Works

```
Agent (Decision-maker)
    ↓
    Takes Action
    ↓
    Environment Responds
    ↓
    Receives Reward/Penalty
    ↓
    Learns: "Good action → more reward"
    ↓
    Improves Strategy
```

Key Concepts

Concept	Meaning	Example
Agent	The learner	Robot, game player, car
Environment	The world agent acts in	Game board, real world, simulation
Action	Decision agent makes	Move, press button, steer
Reward	Feedback signal	+1 for winning, -1 for losing
Policy	Strategy agent learns	“When I see X, do Y”

Simple Example: Training a Dog

```
Dog (Agent) is in living room (Environment)

Iteration 1:
                    Dog jumps on sofa (Action)
                    Owner scolds (Negative reward: -1)
                    Dog learns: "Jumping = bad"

Iteration 2:
                    Dog lies down (Action)
                    Owner gives treat (Positive reward: +1)
                    Dog learns: "Lying down = good"

After many iterations:

        Dog has learned optimal policy: "Lie down to get rewards"
```

Real-World Applications

1. Game Playing -

AlphaGo:	Learned to beat world champion at Go
Video game agents:	Learn optimal strategies
Chess engines:	Self-play improvement

2. Robotics –

Robot arm learning to pick objects

Walking/balance learning

Navigation and obstacle avoidance

3. Autonomous Systems -

Self-driving cars learning safe driving

Trading systems learning profit strategies

Resource allocation and scheduling

Advantages of Reinforcement Learning

- ✅ Works in complex, dynamic environments
- ✅ Can learn optimal long-term strategies
- ✅ No need for labeled data
- ✅ Naturally handles trial-and-error learning

Disadvantages of Reinforcement Learning

- ❌ Requires many interactions (slow learning)
- ❌ Dangerous in real-world (learning by failure)
- ❌ Hard to design good reward system
- ❌ Can learn unexpected behaviors
- ❌ Requires simulation environment or safe testing

1.2.4 COMPARISON TABLE - Three Types of ML

Aspect	Supervised	Unsupervised	Reinforcement
Data Required	Labeled (input + output)	Unlabeled data only	Reward signals
Learning Style	With teacher	Self-discovery	Trial-and-error
Feedback	Immediate (right/wrong)	None (human interprets)	Reward/penalty
Ease of Setup	Hard (need labels)	Easy (no labels needed)	Hard (need simulation)
Typical Tasks	Classify, predict values	Group, compress data	Optimize actions
Output	Predictions/categories	Groups or patterns	Policy/strategy
Real-world Speed	Fast to deploy	Medium	Slow (needs training)
Example	Email spam filter	Customer segmentation	Game-playing AI
Success Metric	Accuracy on test data	Human interpretation	Cumulative reward

Exam Summary for 1.2

Key Points to Remember: -

- **Supervised:**

  Labeled data → Classification (categories) or Regression (numbers)

- **Unsupervised:**

  Unlabeled data → Clustering (groups) or Dimensionality Reduction (simplify)

- **Reinforcement:**

  Rewards/penalties → Agent learns optimal policy

- Each type suited for different problems

- Supervised needs labels (expensive); Unsupervised needs no labels (cheap)

1.3 APPLICATIONS AND CHALLENGES IN ML

1.3.1 Real-World Applications by Domain

1. HEALTHCARE

Applications: -

Disease Diagnosis:	Predict disease from patient data (X-ray analysis, blood tests)
Drug Discovery:	Identify promising drug compounds
Treatment Planning:	Recommend personalized treatment
Patient Risk Prediction:	Identify high-risk patients for early intervention
Example:	ML model analyzes chest X-ray → → Detects tuberculosis with 95% accuracy

---

2. FINANCE

Applications: -

Credit Risk Assessment:	Predict loan default probability
Fraud Detection:	Identify fraudulent transactions
Stock Price Prediction:	Forecast market movements
Customer Credit Scoring:	Determine creditworthiness
Example:	ML system detects unusual spending pattern → → Blocks fraudulent transaction

---

3. E-COMMERCE

Applications: -

Product Recommendations:	"Customers who bought X also bought Y"
Price Optimization:	Set dynamic prices based on demand

Customer Segmentation: Group customers for targeted marketing

Demand Forecasting: Predict product demand

Example: Amazon recommends products based on browsing history

---

4. OTHER DOMAINS

- **Transportation:** - Self-driving cars (autonomous navigation) - Route optimization for delivery - Traffic prediction
- **Manufacturing:** - Equipment failure prediction (preventive maintenance) - Quality control (defect detection) - Production optimization
- **Agriculture:** - Crop yield prediction - Soil analysis for fertilizer optimization - Pest/disease detection
- **Education:** - Student performance prediction - Personalized learning paths - Dropout risk identification

---

1.3.2 Challenges in Machine Learning

Challenge 1: DATA QUALITY

Problem: “Garbage in, garbage out” - Incomplete data (missing values) - Incorrect data (errors in recording) - Inconsistent data (different formats) - Biased data (not representative)

Impact: Poor data → Poor model, regardless of algorithm quality

Solution: - Data cleaning and validation - Handle missing values - Remove outliers - Ensure representative sample

---

Challenge 2: DATA AVAILABILITY

Problem: Sufficient labeled data is expensive - Requires manual labeling - Time-consuming process - Expert knowledge often needed

Impact: Can’t train model without enough data

Solution: - Use unsupervised learning (no labels needed) - Use transfer learning (leverage pre-trained models) - Use synthetic data generation

---

Challenge 3: OVERFITTING

Problem: Model learns training data too well (including noise) - Memorizes examples - Fails on new, unseen data - High training accuracy, low test accuracy

Example: - Training accuracy: 99% - Test accuracy: 60% - Problem: Model memorized training data, not learning patterns

Solution: - Use more training data - Simpler model - Early stopping - Regularization techniques

---

Challenge 4: UNDERFITTING

Problem: Model too simple to learn underlying patterns - Fails on both training and test data - Misses important patterns

Example: - Training accuracy: 65% - Test accuracy: 62% - Problem: Model too simple to understand data

Solution: - More complex model - More features - longer training

Challenge 5: BIAS IN DATA

Problem: Training data doesn't represent entire population - Historical bias (past discrimination encoded) - Sampling bias (not representative) - Measurement bias (systematic errors)

Example: - Trained on faces of one ethnicity → performs poorly on others - Trained on male-dominated hiring data → discriminates against women

Impact: Model discriminates against certain groups

Solution: - Diverse training data - Fairness metrics - Regular audits - Remove biased features

Challenge 6: COMPUTATIONAL RESOURCES

Problem: Training large models requires significant resources - High CPU/GPU cost - Long training time - Memory limitations

Impact: Some models take weeks to train; expensive infrastructure needed

Solution: - Use distributed computing - Cloud computing platforms - Optimize algorithms - Use simpler models where possible

Challenge 7: ETHICAL CONCERNS

Problems: - **Privacy:** Training data may contain personal information - **Transparency:** "Black box" models hard to explain - **Fairness:** Models may discriminate - **Accountability:** Who's responsible if model makes wrong decision?

Examples: - Facial recognition invading privacy - ML used for predicting criminal behavior (unreliable, biased) - Loan approval algorithm discriminating against minorities

Solution: - Ethical guidelines - Transparency requirements - Privacy-preserving techniques - Regular bias audits

Summary: Challenges at a Glance

Challenge	Cause	Impact	Solution
Data Quality	Incomplete, incorrect, inconsistent	Poor model	Clean data
Data Availability	Expensive labeling	Can't train	Transfer learning, unsupervised
Overfitting	Model too complex	Works train, fails test	Simpler model, more data
Underfitting	Model too simple	Fails both	Complex model, more features
Bias	Non-representative data	Discrimination	Diverse data, fairness audit
Computation	Heavy resources needed	Expensive, slow	Cloud computing, optimization
Ethics	Privacy, fairness, transparency	Harmful decisions	Guidelines, transparency, audit

Exam Summary for 1.3

Key Points to Remember: -

- ✅ ML applications across healthcare, finance, e-commerce, etc.
- ✅ Data quality > data quantity
- ✅ Overfitting = memorization; Underfitting = too simple
- ✅ Bias in data → Biased model (ethical issue)
- ✅ Privacy and ethics are critical concerns
- ✅ Computational resources can be limiting

1.4 INTRODUCTION TO PYTHON FOR ML

1.4.1 Python Basics Relevant to ML

Why Python for ML?

- ✅ Easy to learn and read
- ✅ Rich ecosystem of ML libraries
- ✅ Wide community support
- ✅ Industry standard for ML/Data Science

Python Fundamentals

1. ***Variables and Data Types***
2. ***Lists (Collections)***
3. ***Conditionals (If-Else)***
4. ***Loops***
5. ***Functions***

Python Fundamentals

1. Variables and Data Types

```
# Numbers
age = 25            # Integer
height = 5.9        # Float
price = 1000.50     # Float

# Strings
name = "Alice"      # String
city = 'Mumbai'     # Single or double quotes

# Booleans
is_student = True   # True or False
has_loan = False

# Print values
print(age)          # Output: 25
print(name)         # Output: Alice
```

2. Lists (Collections)

```
# Create list
marks = [85, 90, 78, 92, 88]
students = ["Alice", "Bob", "Charlie"]

# Access elements (0-indexed)
print(marks[0])     # Output: 85 (first element)
print(students[1]) # Output: Bob (second element)

# Add to list
marks.append(95)    # Add 95 at end
marks.insert(0, 80) # Add 80 at position 0

# List length
print(len(marks))  # Output: 6

# Loop through list
for mark in marks:
    print(mark)     # Prints each mark
```

NOTES:

3. Conditionals (If-Else)

```
# Simple if
age = 20
if age >= 18:
    print("You are adult")

# If-else
score = 45
if score >= 50:
    print("Pass")
else:
    print("Fail")

# If-elif-else
marks = 75
if marks >= 90:
    print("Grade A")
elif marks >= 80:
    print("Grade B")
elif marks >= 70:
    print("Grade C")
else:
    print("Grade D")
```

4. Loops

```
# For loop
for i in range(5):      # 0, 1, 2, 3, 4
    print(i)

# For loop with list
fruits = ["apple", "banana", "orange"]
for fruit in fruits:
    print(fruit)

# While loop
count = 0
while count < 3:
    print(count)
    count = count + 1
```

NOTES:

5. Functions

```python
# Simple function
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")      # Output: Hello, Alice!

# Function with return
def add(a, b):
    result = a + b
    return result

sum_value = add(5, 3)
print(sum_value)    # Output: 8

# Function with default value
def welcome(name="Guest"):
    print(f"Welcome, {name}!")

welcome()           # Output: Welcome, Guest!
welcome("Bob")      # Output: Welcome, Bob!
```

NOTES:

1.4.2 Overview of ML Libraries

1. NumPy - Numerical Computing

Purpose: Work with numerical arrays (lists of numbers)

```python
import numpy as np

# Create array
arr = np.array([1, 2, 3, 4, 5])
print(arr)          # Output: [1 2 3 4 5]

# Create 2D array (matrix)
matrix = np.array([[1, 2, 3],
                   [4, 5, 6]])
print(matrix)       # 2x3 matrix

# Basic operations
print(arr + 10)     # Add 10 to each element
print(arr * 2)      # Multiply each by 2
print(np.mean(arr)) # Average: 3.0
print(np.sum(arr))  # Sum: 15

# Useful functions
print(arr.shape)    # (5,) - size of array
print(arr.dtype)    # int64 - data type
```

When to use: Mathematical operations, matrix multiplication, scientific computing

---

2. Pandas - Data Manipulation

Purpose: Work with tabular data (like Excel spreadsheets)

```python
import pandas as pd

# Create DataFrame (like Excel table)
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
}

df = pd.DataFrame(data)
print(df)

#      Name  Age  Salary
# 0  Alice   25   50000
# 1    Bob   30   60000
# 2  Charlie 35   70000

# Access columns
print(df['Name'])       # All names
print(df['Age'].mean()) # Average age: 30

# Access rows
print(df.loc[0])        # First row

# Filter data
print(df[df['Age'] > 28]) # Only age > 28

# Basic statistics
print(df.describe())    # Summary statistics
```

When to use: Load data, clean data, explore data, transform data

---

3. Matplotlib - Data Visualization

Purpose: Create charts and graphs

```python
import matplotlib.pyplot as plt

# Create plot data
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
sales = [100, 150, 120, 180, 200]

# Line plot
plt.plot(months, sales)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Sales')
plt.show()

# Bar chart
plt.bar(months, sales)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Sales')
plt.show()

# Histogram (distribution)
marks = [65, 70, 75, 80, 85, 85, 90, 95]
plt.hist(marks, bins=5)
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.title('Distribution of Marks')
plt.show()
```

When to use: Visualize data patterns, check data quality, communicate results

---

4. Scikit-learn - Machine Learning

Purpose: Build ML models easily

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Sample data: student features and pass/fail
X = [[2.5, 80],    # Study hours, marks
     [3.0, 85],
     [1.5, 60],
     [4.0, 90],
     [2.0, 75]]

y = [1, 1, 0, 1, 1]  # 1 = Pass, 0 = Fail

# Split into training (80%) and testing (20%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Create and train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)
print(predictions)

# Check accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

When to use: Build classification/regression models, train models, make predictions

---

1.4.3 Writing and Executing Simple ML Scripts

Example 1: Simple Classification Model

```python
# Step 1: Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Step 2: Load data
data = {
    'Age': [20, 25, 30, 35, 40, 45],
    'Income': [30000, 40000, 50000, 60000, 70000, 80000],
    'BuysProduct': [0, 0, 1, 1, 1, 1]  # 0=No, 1=Yes
}
df = pd.DataFrame(data)

# Step 3: Prepare data
X = df[['Age', 'Income']]  # Features
y = df['BuysProduct']       # Target

# Step 4: Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Step 5: Create and train model
model = DecisionTreeClassifier(max_depth=3)
model.fit(X_train, y_train)

# Step 6: Make predictions
predictions = model.predict(X_test)
print(f"Predictions: {predictions}")

# Step 7: Evaluate
accuracy = accuracy_score(y_test, predictions)
print(f"Model Accuracy: {accuracy:.2f}")
```

Example 2: Regression Model

```python
# Predict house price from size
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Data: house size (sq ft) and price (₹)
data = {
    'Size': [1000, 1500, 2000, 2500, 3000],
    'Price': [3000000, 4500000, 6000000, 7500000, 9000000]
}
df = pd.DataFrame(data)

# Prepare
X = df[['Size']]
y = df['Price']

# Train model
model = LinearRegression()
model.fit(X, y)

# Predict new house
new_size = [[2200]]
predicted_price = model.predict(new_size)
print(f"Predicted price for 2200 sq ft: ₹{predicted_price[0]:,.0
f}")

# Check error
predictions = model.predict(X)
mse = mean_squared_error(y, predictions)
print(f"Mean Squared Error: {mse}")
```

Example 3: Data Exploration with Pandas

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load data
data = {
    'StudentID': [1, 2, 3, 4, 5],
    'StudyHours': [2, 3, 5, 4, 6],
    'Marks': [50, 60, 85, 75, 90]
}
df = pd.DataFrame(data)

# Explore data
print("First few rows:")
print(df.head())

print("\nBasic statistics:")
print(df.describe())

# Visualize relationship
plt.scatter(df['StudyHours'], df['Marks'])
plt.xlabel('Study Hours')
plt.ylabel('Marks')
plt.title('Study Hours vs Marks')
plt.show()

# Find students with high marks
high_performers = df[df['Marks'] > 75]
print("\nHigh performers (Marks > 75):")
print(high_performers)
```

1.4.4 Running Python Scripts

Method 1: Command Line

```
# Save as: my_script.py

# Run from terminal
python my_script.py

# Output appears in terminal
```

Method 2: Interactive Notebook

```
# Jupyter Notebook or Google Colab
# Write code in cells
# Run each cell separately
# See output immediately
```

Method 3: IDE (PyCharm, VS Code)

```
# Write script in editor
# Click "Run" button
# See output in console
```

Exam Summary for 1.4

Key Points to Remember: -

- ✅ Python: Variables, Lists, Conditionals, Loops, Functions
- ✅ **NumPy:** Numerical arrays, matrix operations
- ✅ **Pandas:** DataFrames, data manipulation, exploration
- ✅ **Matplotlib:** Visualize data with plots and charts
- ✅ **Scikit-learn:** Build ML models, train, predict
- ✅ ML workflow: Load → Prepare → Train → Predict → Evaluate
- ✅ Simple scripts: Few lines of code, clear structure

UNIT 1: QUICK REVISION SHEET (2-Page Summary)

Quick Reference: Definitions

Term	Definition
Machine Learning	Systems that improve from experience without explicit programming
Supervised Learning	Learning from labeled data (input + output pairs)
Unsupervised Learning	Finding patterns in unlabeled data
Reinforcement Learning	Learning through rewards and penalties
Classification	Predicting discrete categories
Regression	Predicting continuous numerical values
Clustering	Grouping similar items without labels
Overfitting	Model memorizes training data, fails on new data
Training Data	Data used to teach the model (60-80%)
Test Data	Data used to evaluate model (20-40%, never seen before)

Key Comparisons at a Glance

ML Types

```
SUPERVISED: Label → Yes
            Task → Classification (categories) or Regression (numbers)
            Cost → High (expensive labels)
            Example → Email spam filter

UNSUPERVISED: Label → No
              Task → Clustering (groups) or Dimensionality Reduction
              Cost → Low (no labels needed)
              Example → Customer segmentation

REINFORCEMENT: Label → Rewards/Penalties
               Task → Learn optimal policy/strategy
               Cost → High (needs simulation/interaction)
               Example → Game playing AI
```

Problems and Solutions

Problem	Cause	Fix
Low accuracy	Bad data	Clean data, get more data
Overfitting	Model too complex	Simpler model, more data
Underfitting	Model too simple	Complex model, more features
Bias	Non-representative data	Diverse training data
Slow training	Large data/model	Cloud computing, simpler model

Python Libraries Quick Reference

```python
# NumPy - Numerical operations
import numpy as np
arr = np.array([1,2,3])
print(np.mean(arr))  # Average

# Pandas - Data manipulation
import pandas as pd
df = pd.DataFrame(data)
df['column'].mean()  # Column average

# Matplotlib - Visualization
import matplotlib.pyplot as plt
plt.plot(x, y)
plt.show()

# Scikit-learn - ML models
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Real-World Applications Summary

Domain	Application	ML Type
Healthcare	Disease diagnosis, drug discovery	Supervised
Finance	Fraud detection, credit scoring	Supervised
E-commerce	Recommendations, price optimization	Supervised + Unsupervised
Transportation	Autonomous vehicles, route optimization	Reinforcement
Manufacturing	Equipment failure prediction, quality control	Supervised

Common Challenges

1. **Data Quality** → Need clean, error-free data
2. **Labeling Cost** → Supervised learning requires expensive labels
3. **Overfitting** → Model memorizes instead of learning
4. **Bias** → Training data not representative
5. **Computational Resources** → Training takes time and power
6. **Ethics** → Privacy, fairness, transparency concerns

Typical ML Workflow

```
1. DATA COLLECTION
   ↓
2. DATA CLEANING & PREPARATION
   ↓
3. EXPLORATORY ANALYSIS (Visualize, understand)
   ↓
4. SPLIT: Train (80%) / Test (20%)
   ↓
5. CHOOSE MODEL
   ↓
6. TRAIN MODEL on training data
   ↓
7. PREDICT on test data
   ↓
8. EVALUATE (Calculate accuracy/error)
   ↓
9. DEPLOYMENT (Use in real world)
```

Exam Focus Areas

Must Know: -

Definition of ML, AI, Data Science (differences)
Three types: Supervised, Unsupervised, Reinforcement
Classification vs Regression (difference + examples)
Clustering and dimensionality reduction (concept only)
Challenges: Data quality, overfitting, bias, ethics
Python: Variables, loops, functions, libraries
NumPy, Pandas, Matplotlib, Scikit-learn basics
Typical ML workflow (load → train → predict → evaluate)

Common Exam Questions:

1. “Define Machine Learning and distinguish it from traditional programming”
2. “Explain three types of ML with examples”
3. “What is the difference between classification and regression?”
4. “Describe the overfitting problem and its solution”
5. “List real-world applications of ML in healthcare/finance”
6. “What libraries would you use for ML in Python?”
7. “Explain the data preparation process before building an ML model”
8. “What ethical concerns exist in ML? How to address them?”

END OF UNIT 1 STUDY MATERIAL

Total Topics Covered: 4 (1.1 Basics, 1.2 Types, 1.3 Applications, 1.4 Python)

Reading Time: 1.5 - 2 hours
Practice Time: 2 - 3 hours
Total Study Time: 4 hours recommended

Next Steps: - Review each section’s summary - Practice Python code examples - Try implementing simple models from Section 1.4 - Prepare answers to common exam questions

---

Compiled for: Diploma-level ML Course
Last Updated: December 2025
Suitable for: MSBTE and similar diploma examinations