

MACHINE LEARNING – COMPLETE NOTES (WITH EXPLANATION & CODE)

1. What is Machine Learning?

Machine Learning is a branch of Artificial Intelligence in which a computer system learns from data and improves its performance without being explicitly programmed. Instead of writing rules for every situation, we provide data and the machine automatically learns patterns from it.

For example, if we give past student marks to a machine learning model, it can learn the pattern and predict marks for new students.

2. Why Machine Learning is Important?

Machine Learning is used because:

- Data is very large and complex
- Writing manual rules is difficult
- Machines can automatically find patterns
- Predictions become more accurate with experience

Machine Learning helps in automation, decision-making, and prediction.

3. Types of Machine Learning

3.1 Supervised Learning

Supervised Learning uses labeled data, which means both input and correct output are provided.

Example:

Study hours → Marks

Email text → Spam or Not Spam

Supervised Learning is mainly used for prediction.

There are two types:

Regression – Output is a number Examples:
salary, marks, price

Classification – Output is a category
Examples: pass/fail, yes/no

3.2 Unsupervised Learning

Unsupervised Learning uses data without labels. The machine does not know the correct output and tries to find hidden patterns or groups in the data.

Example:

- Grouping customers based on behavior
-

3.3 Reinforcement Learning

Reinforcement Learning works on reward and punishment. The model learns by interacting with the environment and improves its performance based on feedback.

Example:

- Game playing AI
 - Robotics
-

4. Machine Learning Workflow

Every Machine Learning program follows the same basic steps:

1. Collect the dataset
 2. Clean the data
 3. Separate input and output
 4. Split data into training and testing
 5. Choose an algorithm
 6. Train the model
 7. Test the model
 8. Make predictions
-

5. Important Machine Learning Terms

- **Dataset:** Collection of data
 - **Feature (X):** Input values
 - **Label (y):** Output value
 - **Training:** Teaching the model
 - **Testing:** Checking performance
 - **Prediction:** Output for new data
-

6. Linear Regression

Linear Regression is a supervised learning algorithm used to predict a numerical value. It works by fitting a straight line that best represents the relationship between input and output.

The equation of a straight line is:

$$\underline{y = mx + c}$$

Where:

- x is the input
- y is the output
- m is the slope
- c is the intercept

Linear Regression is used when the output is continuous, such as marks, salary, or price

In the equation $y = mx + c$:

Slope (m)

The **slope** tells us **how fast y changes when x changes**.

- It shows the **rate of change**
- It tells the **direction of the line**

Meaning:

- If **m is positive (+)** → y increases when x increases (line goes upward)
- If **m is negative (-)** → y decreases when x increases (line goes downward) □ If **m = 0** → y does not change (horizontal line)

Example:

If **m = 2**, it means

☞ when **x increases by 1**, **y increases by 2**

So slope answers the question:

“For each 1 unit change in x, how much does y change?”

📌 Intercept (c)

The **intercept** tells us **where the line cuts the y-axis**.

Meaning:

- It is the **value of y when $x = 0$**
- It shows the **starting value of y**

Example:

If $c = 3$, it means

₹ when $x = 0$, $y = 3$

So the line starts from $y = 3$ on the y-axis.

💡 Simple Real-Life Example

Imagine:

- x = number of hours worked
- y = total money earned
- $m = ₹100$ per hour
- $c = ₹500$ fixed salary

Equation: $y =$

$100x + 500$

- **Slope (100):** You earn ₹100 for each hour

Python Code: Linear Regression Example

```
import pandas as pd
from sklearn.linear_model import LinearRegression
```

```
data = {
    'Hours': [1, 2, 3, 4, 5],
    'Marks': [20, 30, 40, 50, 60]}
```

```

}

df = pd.DataFrame(data)

X = df[['Hours']]
y = df['Marks']

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

prediction = model.predict([[6]]) print(prediction)

```

The model learns the relationship between study hours and marks and predicts marks for 6 study hours.

from sklearn.linear_model import LinearRegression

LinearRegression → A machine learning model used to find a linear relationship between input and output

7. Train-Test Split

If we train and test a model on the same data, it may memorize the data instead of learning. To avoid this, the dataset is divided into training data and testing data.

```

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2
)

```

Training data is used to teach the model, and testing data is used to check accuracy.

Python Code: Logistic Regression Example

```

from sklearn.linear_model import LogisticRegression

```

```

X = [[30], [40], [50], [60]]
y = [0, 0, 1, 1]

```

```

model = LogisticRegression() model.fit(X,
y)

```

```
print(model.predict([[45]]))
```

8. Logistic Regression

Logistic Regression is used for classification problems where the output has only two possible values.

Examples:

- Pass or Fail
- Yes or No
- Spam or Not Spam

It returns probabilities between 0 and 1 and then classifies the result.

Python Code: Logistic Regression Example

```
from sklearn.linear_model import LogisticRegression
```

```
X = [[30], [40], [50], [60]]
```

```
y = [0, 0, 1, 1]
```

```
model = LogisticRegression()
```

```
model.fit(X, y)
```

```
print(model.predict([[45]]))
```

9. K-Nearest Neighbors (KNN)

KNN is a classification algorithm that predicts the result based on the nearest data points. The output depends on the majority class among the nearest neighbors.

Python Code: KNN Example

```
from sklearn.neighbors import KNeighborsClassifier
```

```
X = [[1], [2], [3], [6], [7], [8]]
```

```
y = [0, 0, 0, 1, 1, 1]
```

```
model = KNeighborsClassifier(n_neighbors=3)
```

```
model.fit(X, y)
```

```
print(model.predict([[5]]))
```

10. Decision Tree

A Decision Tree is a tree-like model that makes decisions based on conditions. Each node represents a question, and each branch represents an answer.

Python Code: Decision Tree Example

```
from sklearn.tree import DecisionTreeClassifier
```

```
X = [[25, 50000], [40, 80000], [30, 60000]]
```

```
y = [0, 1, 1]
```

```
model = DecisionTreeClassifier()
```

```
model.fit(X, y)
```

```
print(model.predict([[35, 70000]]))
```

11. Model Evaluation (Accuracy)

Accuracy tells how many predictions are correct.

```
from sklearn.metrics import accuracy_score
```

```
y_pred = model.predict(X)
```

```
print(accuracy_score(y, y_pred))
```

12. Overfitting and Underfitting

- **Overfitting:** Model learns the training data too well and performs poorly on new data.
- **Underfitting:** Model does not learn enough from the data.

Proper data size, correct algorithm, and train-test split help avoid these problems.

13. Machine Learning Libraries

- **NumPy** – Numerical operations
- **Pandas** – Data handling
- **Matplotlib** – Visualization
- **Seaborn** – Advanced graphs
- **Scikit-learn** – Machine Learning algorithms