Unit III

Machine Learning for Data Science:

Introduction to Machine Learning, Regression, Gradient Descent, Supervised Learning-Introduction, Logistic Regression, Softmax Regression, Classification with KNN, Decision Tree, Random Forest,Naive Bayes, SVM, Unsupervised Learning.

- **What is Machine Learning??**

Machine learning is an essential branch of artificial intelligence that enables systems to automatically learn and improve from experience without being explicitly programmed. In the realm of data science, it plays a crucial role in extracting valuable insights and making informed predictions from large datasets.

Supervised Learning: This involves training a model on labeled data, which means that each input comes with a corresponding output. Common algorithms include linear regression, decision trees, and support vector machines.

Unsupervised Learning: In this approach, the model is trained on unlabeled data, and it tries to find hidden patterns or intrinsic structures in the input data. Examples include clustering algorithms like K-means and hierarchical clustering.

Semi-supervised Learning: This method combines both labeled and unlabeled data to improve learning accuracy. It's useful when acquiring a fully labeled dataset is costly.

Reinforcement Learning: This involves training models to make a sequence of decisions by rewarding desired behaviors and punishing undesired ones. It's often used in game playing and robotics.

Deep Learning: A subset of machine learning that uses neural networks with many layers (hence "deep") to model complex patterns in data. It's highly effective for image and speech recognition.

Applications in Data Science

1. **Predictive Analytics**: Using historical data to predict future outcomes. For example, predicting stock prices or customer churn.
2. **Natural Language Processing (NLP)**: Enabling machines to understand and interpret human language. Applications include sentiment analysis, language translation, and chatbots.
3. **Computer Vision**: Teaching machines to interpret and make decisions based on visual data. Examples include facial recognition, object detection, and autonomous vehicles.
4. **Anomaly Detection**: Identifying unusual patterns that do not conform to expected behavior. This is crucial in fraud detection, network security, and quality control.
5. **Recommendation Systems**: Suggesting products or content to users based on their preferences and past behavior. Think of Netflix or Amazon recommendations.

Steps in a Machine Learning Project

1. **Data Collection**: Gathering data from various sources.
2. **Data Preparation**: Cleaning and preprocessing the data to make it suitable for analysis.
3. **Model Selection**: Choosing the appropriate machine learning algorithm.
4. **Training**: Feeding the prepared data into the algorithm to train the model.
5. **Evaluation**: Assessing the model's performance using metrics like accuracy, precision, and recall.
6. **Deployment**: Integrating the model into a production environment to make real-time predictions.
7. **Monitoring**: Continuously tracking the model's performance and making necessary adjustments.

1. Healthcare

- **Disease Prediction and Diagnosis**: Machine learning models can analyze medical data to predict the likelihood of diseases such as diabetes, heart disease, and cancer. For instance, IBM Watson Health uses machine learning to assist doctors in diagnosing diseases and recommending treatments.
- **Medical Imaging**: Deep learning techniques are used to analyze medical images (like X-rays and MRIs) to detect abnormalities and assist radiologists in making accurate diagnoses.

2. Finance

- **Fraud Detection**: Banks and financial institutions use machine learning to detect fraudulent transactions. For example, algorithms analyze patterns in transaction data to identify anomalies that could indicate fraud.
- **Algorithmic Trading**: Machine learning models analyze market data to make trading decisions. These models can execute trades at high speeds, optimizing profits based on market trends and historical data.

3. Retail

- **Recommendation Systems**: Online retailers like Amazon and Netflix use machine learning to recommend products or content to users based on their past behavior. These systems analyze user preferences and browsing history to provide personalized recommendations.
- **Inventory Management**: Retailers use machine learning to predict demand for products, optimize inventory levels, and reduce stockouts. This helps in efficient supply chain management and reduces costs.

Regression

It is a fundamental concept in data science, and it's all about modeling the relationship between a dependent variable (usually called the target or outcome) and one or more independent variables (predictors or features). Here’s a quick rundown of key regression techniques:

1. **Linear Regression**: This is the simplest form of regression. It assumes a linear relationship between the target and the predictors. The goal is to find the best-fitting straight line through the data points.
2. **Multiple Linear Regression**: An extension of linear regression that uses two or more independent variables to predict a single dependent variable.
3. **Polynomial Regression**: This is used when the relationship between the variables is not linear. It fits a polynomial equation to the data.

Hours Studied	Exam Score
1	50
2	55
3	65
4	70
5	80

Solution:

1. **Fit a simple linear regression model:**
   - Dependent Variable (y): Exam Score
   - Independent Variable (x): Hours Studied

   Using the least squares method, the regression line equation is:

Step 1: Calculate the Mean

First, let's calculate the mean of the hours studied ($\bar{x}$) and the mean of the exam scores ($\bar{y}$):

$$\bar{x} = \frac{1+2+3+4+5}{5} = \frac{15}{5} = 3$$

$$\bar{y} = \frac{50+55+65+70+80}{5} = \frac{320}{5} = 64$$

Step 2: Calculate the Slope (β_1)

The formula to calculate the slope (β_1) is:

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Let's calculate the terms in the numerator and denominator:

$$\sum (x_i - \bar{x})(y_i - \bar{y}) = (1-3)(50-64) + (2-3)(55-64) + (3-3)(65-64) + (4-3)(70 -$$

$$= (-2)(-14) + (-1)(-9) + (0)(1) + (1)(6) + (2)(16)$$

$$= 28 + 9 + 0 + 6 + 32 = 75$$

$$\sum (x_i - \bar{x})^2 = (1-3)^2 + (2-3)^2 + (3-3)^2 + (4-3)^2 + (5-3)^2$$

$$= (-2)^2 + (-1)^2 + (0)^2 + (1)^2 + (2)^2$$

Now, calculate the slope:

$$\beta_1 = \frac{75}{10} = 7.5$$

Step 3: Calculate the Intercept (β_0)

The formula to calculate the intercept (β_0) is:

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 64 - 7.5 \times 3 = 64 - 22.5 = 41.5$$

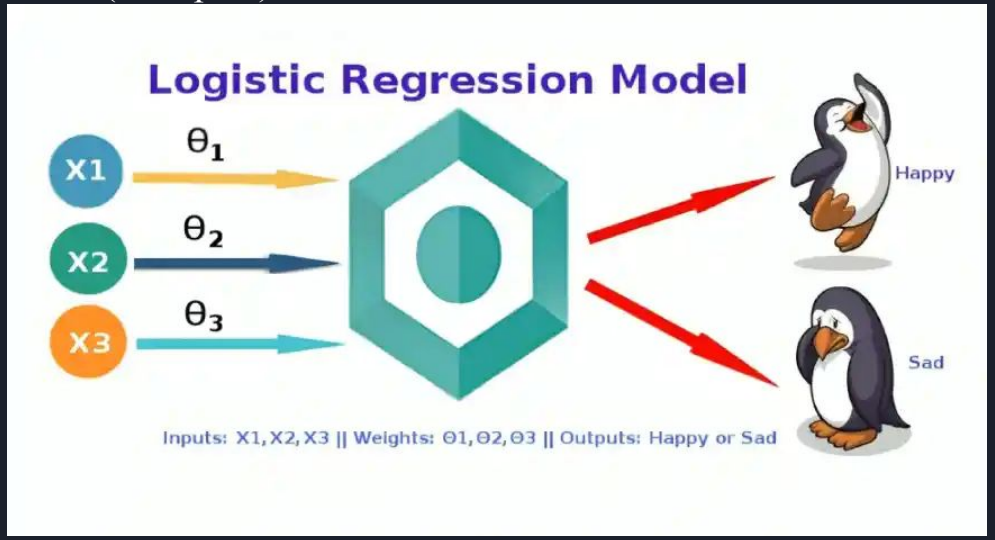
Step 4: Write the Regression Equation

The regression equation is:

$$\hat{y} = \beta_0 + \beta_1 x = 41.5 + 7.5x$$

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis. It is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

for example, A person will survive this accident or not, The student will pass this exam or not. The outcome can either be yes or no (2 outputs).



types of logistic regression.

Binary Logistic Regression

Binary logistic regression is used to predict the probability of a binary outcome, such as yes or no, true or false, or 0 or 1. For example, it could be used to predict whether a customer will churn or not, whether a patient has a disease or not, or whether a loan will be repaid or not.

Multinomial Logistic Regression

Multinomial logistic regression is used to predict the probability of one of three or more possible outcomes, such as the type of product a customer will buy, the rating a customer will give a product, or the political party a person will vote for.

Ordinal Logistic Regression

It is used to predict the probability of an outcome that falls into a predetermined order, such as the level of customer satisfaction, the severity of a disease, or the stage of cancer.

Predicting a class label using naïve Bayesian classification. We wish to predict the class label of a tuple using naïve Bayesian classification, given the same training data as in Example 8.3 for decision tree induction. The training data were shown earlier in Table 8.1. The data tuples are described by the attributes *age*, *income*, *student*, and *credit_rating*. The class label attribute, *buys_computer*, has two distinct values (namely, {*yes*, *no*}). Let C_1 correspond to the class *buys_computer* = *yes* and C_2 correspond to *buys_computer* = *no*. The tuple we wish to classify is

$$\boldsymbol{X} = (age = youth,\ income = medium,\ student = yes,\ credit\_rating = fair)$$

We need to maximize $P(\boldsymbol{X}|C_i)P(C_i)$, for $i = 1, 2$. $P(C_i)$, the prior probability of each class, can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

To compute $P(\boldsymbol{X}|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$
$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$
$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$
$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$
$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$

$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$$
$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$$

Using these probabilities, we obtain

$$\begin{aligned} P(\boldsymbol{X} \mid buys\_computer = yes) &= P(age = youth \mid buys\_computer = yes) \\ &\quad \times P(income = medium \mid buys\_computer = yes) \\ &\quad \times P(student = yes \mid buys\_computer = yes) \\ &\quad \times P(credit\_rating = fair \mid buys\_computer = yes) \\ &= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044. \end{aligned}$$

Similarly,

$$P(\boldsymbol{X} \mid buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, C_i, that maximizes $P(\boldsymbol{X} \mid C_i)P(C_i)$, we compute

$$P(\boldsymbol{X} \mid buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(\boldsymbol{X} \mid buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts $buys\_computer = yes$ for tuple $\boldsymbol{X}$.

Support Vector Machine (SVM) is a machine learning algorithm that aims to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The key features of SVM include:

1. **Hyperplane**: SVM finds the optimal hyperplane that maximizes the margin between different classes.
2. **Support Vectors**: These are the data points that are closest to the hyperplane and help determine its position.
3. **Margin**: The distance between the hyperplane and the nearest data point from either class. SVM aims to maximize this margin.

Hyperplane example

1. Spam Email Detection

- **Hyperplane**: Separates spam and non-spam emails based on features like word frequency, special characters, etc.
- **Example**: Emails with words like "sale," "win," and multiple exclamation marks might be on one side (spam), while emails with a formal tone and recognized senders are on the other (non-spam).

2. Handwriting Recognition

- **Hyperplane**: Differentiates between different handwritten characters.
- **Example**: In digit recognition (0-9), SVM can separate handwritten 8s from 3s based on pixel intensity and position.

3. Medical Diagnosis

- **Support Vectors**: Patients whose symptoms and test results place them on the borderline between two diagnoses, such as borderline blood sugar levels in diabetes diagnosis. These patients help define the criteria for classification.

4. Image Classification

- **Support Vectors**: Images that have features of multiple classes, such as a picture of a dog that has some features resembling a cat (e.g., similar fur pattern). These images help refine the classifier.

Hyperplane and Margin

- **Hyperplane**: This is the line (or plane) that separates the spam emails from the non-spam emails.
- **Support Vectors**: These are the emails that lie closest to the hyperplane. They are crucial in defining the decision boundary.
- **Margin**: This is the distance between the hyperplane and the nearest email from either class (spam or non-spam).

How it Works

1. **Linear SVM**:
    - For linearly separable data, SVM tries to find the best hyperplane (a line in 2D, a plane in 3D) that separates the data into different classes.
2. **Non-Linear SVM**:
    - For non-linearly separable data, SVM uses a technique called the **kernel trick** to transform the data into a higher-dimensional space where it becomes linearly separable. Common kernels include:
        - Linear Kernel
        - Polynomial Kernel
        - Radial Basis Function (RBF) Kernel
        - Sigmoid Kernel

Why Use SVM?

- **Effective in high-dimensional spaces**: SVM is effective in cases where the number of features is greater than the number of samples.
- **Memory-efficient**: It uses a subset of training points (support vectors) in the decision function.
- **Versatile**: Different kernel functions can be specified for the decision function.

Applications

- Text Categorization
- Image Classification
- Bioinformatics (e.g., protein classification, cancer classification)
- Handwriting recognition

- **Logistic Regression:** Logistic regression is a technique used to predict whether an input belongs to one of two categories, making it ideal for situations with a simple binary outcome (yes/no, true/false, or 0/1). Specific to binary classification, logistic regression models a dependent variable with only two possible outcomes using a sigmoid function that transforms predicted values into probabilities between 0 and 1. The algorithm creates a linear boundary that divides the two categories, helping to determine to which class an input belongs.
- **Applications :** Logistic regression is widely used across various domains:

    1. Spam Detection: Classifying emails as spam (1) or not spam (0).
    2. Medical Diagnosis: Predicting the presence (1) or absence (0) of a disease.
    3. Customer Conversion: Determining whether a customer will make a purchase (1) or not (0).

-

Softmax Regression:

Softmax regression, akin to multinomial logistic regression, broadens traditional logistic regression for tasks involving more than two classes. It excels in classification scenarios where the response variable belongs to multiple categories (three or more). For multi-class classification, softmax regression shines. It deploys the softmax function, transforming raw class scores (logits) into probabilities that always add up to one. To differentiate among multiple classes, softmax regression creates unique decision boundaries for each class, enhancing its ability to discern between different categories

Applications: Softmax regression finds its use in various multi-class classification problems:

1. Image Recognition: Classifying images into multiple categories, such as distinguishing between different types of animals (cats, dogs, rabbits).

2. Document Classification: Sorting documents into predefined categories like news articles into sports, politics, and technology.

3. Handwritten Digit Recognition: Identifying digits (0-9) from handwritten samples.

When to Use Logistic Regression:

Use logistic regression for binary classification problems, where you predict the probability of two outcomes (e.g., "Yes" or "No," "Approved" or "Rejected").

When to Use Softmax Regression:

Opt for softmax regression when dealing with multi-class classification problems, where you need to predict one of multiple possible outcomes (e.g., classifying fruit types like "Apple," "Orange," and "Banana" based on their characteristics).

Classification with K-Nearest Neighbors (KNN) is an intuitive and widely used machine learning technique. KNN is a non-parametric and lazy learning algorithm, which makes decisions based on the data points closest to the one being classified.

Here's a quick overview of how KNN works:

1. **Training Phase**:
    - KNN doesn’t have a traditional training phase. Instead, it simply stores the labeled training data.
2. **Classification Phase**:
    - When a new data point (query point) needs to be classified, the algorithm calculates the distance between this point and all the points in the training data. The most common distance metric is the Euclidean distance, but others (like Manhattan or Hamming distance) can also be used.
    - It identifies the k nearest neighbors to the query point based on these distances.
    - The class of the query point is determined by a majority vote of its k nearest neighbors. That is, the query point is assigned to the class that is most common among its neighbors.

Key Considerations:

- **Choosing** k: The value of k is critical. A small value of k can lead to noise in the classification, while a large value might make the classification too generalized. Cross-validation can help in selecting the optimal k.
- **Distance Metric**: The choice of distance metric can influence the algorithm’s performance.
- **Feature Scaling**: Since KNN relies on distance calculations, features should be scaled (e.g., using normalization or standardization) so that one feature doesn't dominate others due to its range.

Advantages:

- Simple and easy to implement.
- No need for a training phase, which saves time.

Disadvantages:

- Computationally expensive, especially with large datasets, as distances need to be calculated for all data points.
- Sensitive to irrelevant or redundant features and to the scale of the features.

Problem: Classify a Point Using KNN

You have the following data points:

Data Point	Feature 1 (X1)	Feature 2 (X2)	Class
A	2	4	0
B	4	2	0
C	4	4	1
D	6	4	1

You want to classify a new point, $P(5, 3)$, using KNN with $k = 3$. The distance metric is **Euclidean distance**.

The formula for Euclidean distance is:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Now, compute the distance from point $P(5, 3)$ to each data point:

1. Distance to $A(2, 4)$:

$$\sqrt{(5-2)^2 + (3-4)^2} = \sqrt{3^2 + (-1)^2} = \sqrt{9+1} = \sqrt{10} \approx 3.16$$

2. Distance to $B(4, 2)$:

$$\sqrt{(5-4)^2 + (3-2)^2} = \sqrt{1^2 + 1^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$$

○

3. Distance to $C(4,4)$:

$$\sqrt{(5-4)^2+(3-4)^2}=\sqrt{1^2+(-1)^2}=\sqrt{1+1}=\sqrt{2}\approx 1.41$$

4. Distance to $D(6,4)$:

$$\sqrt{(5-6)^2+(3-4)^2}=\sqrt{(-1)^2+(-1)^2}=\sqrt{1+1}=\sqrt{2}\approx 1.41$$

2. **Sort Distances**: Arrange distances in ascending order:
   - $B(4, 2)$: 1.41
   - $C(4, 4)$: 1.41
   - $D(6, 4)$: 1.41
   - $A(2, 4)$: 3.16
3. **Choose** `k` **Neighbors**: Select the 3 closest neighbors ($k = 3$):
   - $B(4, 2)$: Class 0
   - $C(4, 4)$: Class 1
   - $D(6, 4)$: Class 1
4. **Determine the Class**: Perform a majority vote among the selected neighbors:
   - Class 0: 1 vote
   - Class 1: 2 votes

   The majority class is **Class 1**.

Final Classification:

The point P(5,3) is classified as **Class 1**.

Problem: Classify Students Based on Grades

You have a dataset of students with the following features:

- **Grades** (High, Medium, Low)
- **Hours Studied** (High, Low)
- **Class Attendance** (Good, Poor)
- **Final Exam Result** (Pass, Fail)

Here's the dataset:

Student	Grades	Hours Studied	Attendance	Result
1	High	High	Good	Pass
2	High	Low	Good	Pass
3	Medium	High	Good	Pass
4	Low	High	Poor	Fail
5	Low	Low	Poor	Fail
6	Medium	Low	Good	Fail

We will use this data to classify whether a student will "Pass" or "Fail" using a Decision Tree.

Step 1: Calculate Entropy

The formula for entropy is:

$$E = -\sum p_i \cdot \log_2(p_i)$$

- There are **3 Pass** and **3 Fail** in the dataset.
- Total records = 6.

Entropy of the dataset:

$$E = -\left(\frac{3}{6} \cdot \log_2\frac{3}{6} + \frac{3}{6} \cdot \log_2\frac{3}{6}\right) = -\left(0.5 \cdot -1 + 0.5 \cdot -1\right) = 1.0$$

Step 2: Choose the First Split

We will calculate the Information Gain for each feature to determine the best split. Information Gain (IG) is calculated as:

$$\text{IG} = \text{Entropy (Parent)} - \sum \left(\frac{\text{Child size}}{\text{Parent size}} \cdot \text{Entropy (Child)} \right)$$

Feature 1: Grades

- Split by **Grades**:
  - High: $2\ \text{Pass}, 0\ \text{Fail}$: Entropy = 0.
  - Medium: $1\ \text{Pass}, 1\ \text{Fail}$: Entropy = 1.0.
  - Low: $0\ \text{Pass}, 2\ \text{Fail}$: Entropy = 0.

Weighted entropy for Grades:

$$\text{E} = \left(\frac{2}{6} \cdot 0 + \frac{2}{6} \cdot 1.0 + \frac{2}{6} \cdot 0 \right) = 0.33$$

Information Gain for Grades:

$$\text{IG} = 1.0 - 0.33 = 0.67$$

Feature 2: Hours Studied

Similarly, calculate the entropy and information gain for other features. For simplicity, let's assume **Grades** has the highest IG and is chosen for the first split.

Step 3: Build the Tree

1. Start with **Grades** as the root node.
2. Create branches for **High**, **Medium**, and **Low**.
   - For **High**: All records are **Pass** → Leaf node = Pass.
   - For **Medium**: Continue splitting using the next feature (e.g., Hours Studied or Attendance).
   - For **Low**: All records are **Fail** → Leaf node = Fail.

Using this tree, we can classify new students. For instance:

- A student with **High Grades** will be classified as **Pass**.
- A student with **Low Grades** will be classified as **Fail**.

A **random forest classifier** builds multiple decision trees during training, using random subsets of the data and features. It combines the predictions of these trees to classify a new data point using **majority voting**.

How It Works (Step-by-Step):

1. **Bootstrap Sampling:**
    - From the original dataset, random subsets (with replacement) are created for each decision tree.
    - This ensures variability among the trees.
2. **Feature Subset Randomization:**
    - At each split, the algorithm selects a random subset of features to determine the split.
    - This reduces overfitting and ensures that the trees are diverse.

Tree Construction:

- Each decision tree is built independently based on its respective subset of data.
- The tree grows until a stopping condition, such as a maximum depth or minimum number of samples in a leaf node, is met.

Prediction (Voting):

- For classification, each tree predicts a class label for the input data point.
- The final prediction is the class with the majority votes from all the trees.

Advantages of Random Forest Classification

- **High Accuracy:** Aggregation reduces overfitting compared to single decision trees.
- **Robustness:** Works well with noisy data and unbalanced classes.
- **Handles Nonlinearity:** Captures complex patterns in data.
- **Feature Importance:** Can estimate the importance of features in predictions.

1. Healthcare

- Disease diagnosis: Identifying diseases like diabetes, cancer, or heart conditions from medical data.
- Genomics: Analyzing gene expression data to understand genetic markers for diseases.

2. Finance

- Credit scoring: Assessing a customer's creditworthiness based on financial history.
- Fraud detection: Spotting unusual patterns in transaction data to identify fraudulent activities.
- Stock market predictions: Forecasting trends based on historical stock data.

3. Retail and E-commerce

- Recommendation systems: Suggesting products to customers based on purchase history and preferences.
- Inventory optimization: Predicting demand for products to manage stock efficiently.

4. Marketing

- Customer segmentation: Grouping customers for targeted marketing campaigns.
- Churn prediction: Identifying customers likely to leave a service and taking preventive actions.