CS4403 - MACHINE LEARNING

COURSE OBJECTIVES:

- To understand the basic concepts of machine learning.
- To understand and build supervised learning models.
- To understand and build unsupervised learning models.
- To learn the basics of deep learning using neural networks.
- To learn the basics of optimization searching and reinforcement learning methods.

UNIT I - INTRODUCTION (11 Periods)

Introduction to Machine Learning: Introduction. Different types of learning, Hypothesis space and inductive bias, Evaluation. Training and test sets, cross validation, Concept of over fitting, under fitting, Bias and Variance. Linear Regression: Introduction, Linear regression, Simple and Multiple Linear regression, Polynomial regression, evaluating regression fit.

UNIT II - SUPERVISED LEARNING (9 Periods)

Bayesian linear regression, gradient descent, Linear Classification Models: Discriminant function – Perceptron algorithm, Probabilistic discriminative model - Logistic regression, Probabilistic generative model - Naïve Bayes, Maximum margin classifier – Support vector machine, Decision Tree, Random Forests, Instance Based Learning- KNN.

UNIT III - NEURAL NETWORKS (9 Periods)

Artificial Neural Networks: Introduction, Biological motivation, ANN representation, appropriate problem for ANN learning, Perceptron, multilayer networks and the back propagation algorithm, Popular CNN Architectures, RNNs, LSTM, BERT, GANS and Generative Models.

UNIT IV - UNSUPERVISED LEARNING AND OPTIMISATION (8 Periods)

Unsupervised learning: Expectation maximization - Gaussian mixture models -K-means / K medoid - hierarchical clustering-top-down, bottom-up –single linkage-multiple linkage. Dimensionality Reduction- Linear Discriminant Analysis, Principal Components Analysis, Factor Analysis, Independent Component Analysis. Optimization- Going Downhill, Least-Squares optimization, Conjugate Gradients - Exploitation and Exploration.

UNIT V - APPLICATION PROBLEMS (9 Periods)

The case studies- churn analysis and prediction using Cox-proportional models, and churn prediction techniques. Credit card fraud analysis with a focus on handling imbalanced data and neural networks. Sentiment analysis- Sentiment mining from the New York Times are addressed using similarity measures like cosine similarity, chi-square, and N-grams. Part-of-speech tagging, stemming, chunking - sales funnel analysis, A/B testing, and campaign effectiveness. Web page layout effectiveness - recommendation systems with collaborative filtering - customer segmentation strategies and lifetime value- portfolio risk conformance and optimization, and Uber alternative routing with graph construction and route optimization.

UNIT I - INTRODUCTION (11 Periods)

Introduction to Machine Learning: Introduction. Different types of learning, Hypothesis space and inductive bias, Evaluation. Training and test sets, cross validation, Concept of over fitting, under fitting,

Bias and Variance. Linear Regression: Introduction, Linear regression, Simple and Multiple Linear regression, Polynomial regression, evaluating regression fit.

Introduction to Machine Learning

Machine learning (ML) allows computers to learn and make decisions without being explicitly programmed. It involves feeding data into algorithms to identify patterns and make predictions on new data. It is used in various applications like image recognition, speech processing, language translation, recommender systems, etc.

Why do we need Machine Learning?

Traditional programming requires exact instructions and doesn't handle complex tasks like understanding images or language well. It can't efficiently process large amounts of data. Machine Learning solves these problems by learning from examples and making predictions without fixed rules. Let's see various reasons why it is important:

1. Solving Complex Business Problems

Traditional programming struggles with tasks like language understanding and medical diagnosis. ML learns from data and predicts outcomes easily.

Examples:

- Image and speech recognition in healthcare.
- Language translation and sentiment analysis.

2. Handling Large Volumes of Data

The internet generates huge amounts of data every day. Machine Learning processes and analyzes this data quickly by providing valuable insights and real-time predictions.

Examples:

- Fraud detection in financial transactions.
- Personalized feed recommendations on Facebook and Instagram from billions of interactions.

3. Automate Repetitive Tasks

ML automates time-consuming, repetitive tasks with high accuracy hence reducing manual work and errors.

Examples:

- Gmail filtering spam emails automatically.
- Chatbots handling order tracking and password resets.
- Automating large-scale invoice analysis for key insights.

4. Personalized User Experience

ML enhances user experience by tailoring recommendations to individual preferences. It analyze user behavior to deliver highly relevant content.

Examples:

- Netflix suggesting movies and TV shows based on our viewing history.
- E-commerce sites recommending products we're likely to buy.

5. Self-Improvement in Performance

ML models evolve and improve with more data helps in making them smarter over time. They adapt to user behavior and increase their performance.

Examples:

- Voice assistants like Siri and Alexa learning our preferences and accents.
- Search engines refining results based on user interaction.
- Self-driving cars improving decisions using millions of miles of driving data.

What Makes a Machine "Learn"?

A machine "learns" by identifying patterns in data and improving its ability to perform specific tasks without being explicitly programmed for every scenario. This learning process helps machines to make accurate predictions or decisions based on the information they receive. Unlike traditional programming where instructions are fixed, ML allows models to adapt and improve through experience.

Here is how the learning process works:

- **Data Input**: Machine needs data like text, images or numbers to analyze. Good quality and enough quantity of data are important for effective learning.
- **Algorithms**: Algorithms are mathematical methods that help the machine find patterns in data. Different algorithms help different tasks such as classification or regression.
- Model Training: During training, the machine adjusts its internal settings to better predict outcomes. It learns by reducing the difference between its predictions and actual results.
- Feedback Loop: Machine compares its predictions with true outcomes and uses this feedback to correct errors. Techniques like gradient descent help it update and improve.
- Experience and Iteration: Machine repeats training many times with data helps in refining its predictions with each pass, more data and iterations improve accuracy.
- Evaluation and Generalization: Model is tested on unseen data to ensure it performs well on real-world tasks.

Machines "learn" by continuously increasing their understanding through data-driven iterations like how humans learn from experience.

Importance of Data in Machine Learning

- Data is the foundation of machine learning (ML) without quality data ML models cannot learn, perform or make accurate predictions.

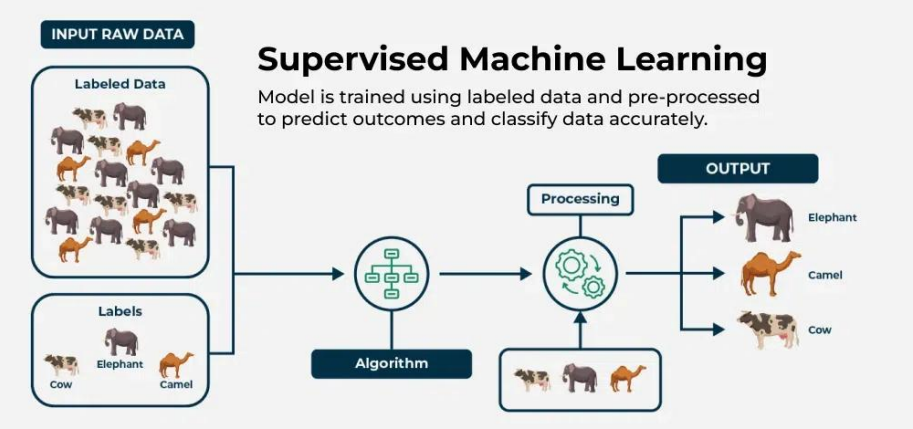
- Data provides the examples from which models learn patterns and relationships.
- High-quality and diverse data improves how well models perform and generalize to new situations.
- It helps models to understand real-world scenarios and adapt to practical uses.
- Features extracted from data are important for effective training.
- Separate datasets for validation and testing measure how well the model works on unseen data.
- Data drives continuous improvements in models through feedback loops.

Types of Machine Learning

There are three main types of machine learning which are as follows:

1. Supervised learning

Supervised learning is a type of machine learning where a model learns from labelled data—meaning every input has a corresponding correct output. The model makes predictions and compares them with the true outputs, adjusting itself to reduce errors and improve accuracy over time. The goal is to make accurate predictions on new, unseen data. For example, a model trained on images of handwritten digits can recognise new digits it has never seen before.

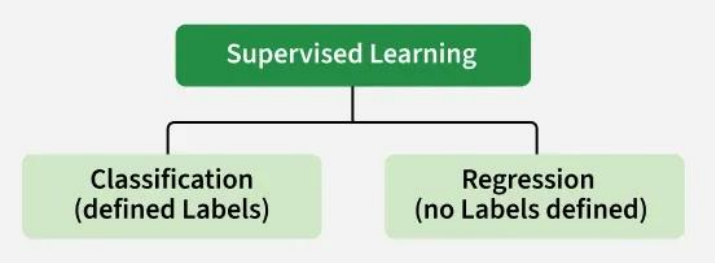


Supervised Machine Learning

Types of Supervised Learning in Machine Learning

Now, Supervised learning can be applied to two main types of problems:

- **Classification:** Where the output is a categorical variable (e.g., spam vs. non-spam emails, yes vs. no).
- **Regression:** Where the output is a continuous variable (e.g., predicting house prices, stock prices).



Types of Supervised Learning

While training the model, data is usually split in the ratio of 80:20 i.e. 80% as training data and the rest as testing data. In training data, we feed input as well as output for 80% of data. The model learns from training data only. We use different supervised learning algorithms (which we will discuss in detail in the next section) to build our model. Let's first understand the classification and regression data through the table below:

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

Sample dataset

Both the above figures have labelled data set as follows:

Figure A: It is a dataset of a shopping store that is useful in predicting whether a customer will purchase a particular product under consideration or not based on his/her gender, age and salary.

- **Input**: Gender, Age, Salary
- **Output**: Purchased i.e. 0 or 1; 1 means yes the customer will purchase and 0 means that the customer won't purchase it.

Figure B: It is a Meteorological dataset that serves the purpose of predicting wind speed based on different parameters.

- **Input**: Dew Point, Temperature, Pressure, Relative Humidity, Wind Direction
- **Output**: Wind Speed

Working of Supervised Machine Learning

The working of supervised machine learning follows these key steps:

1. **Collect Labeled Data**
   - Gather a dataset where each input has a known correct output (label).
   - **Example**: Images of handwritten digits with their actual numbers as labels.
2. **Split the Dataset**
   - Divide the data into training data (about 80%) and testing data (about 20%).
   - The model will learn from the training data and be evaluated on the testing data.
3. **Train the Model**
   - Feed the training data (inputs and their labels) to a suitable supervised learning algorithm (like Decision Trees, SVM or Linear Regression).
   - The model tries to find patterns that map inputs to correct outputs.
4. **Validate and Test the Model**
   - Evaluate the model using testing data it has never seen before.
   - The model predicts outputs and these predictions are compared with the actual labels to calculate accuracy or error.
5. **Deploy and Predict on New Data**
   - Once the model performs well, it can be used to predict outputs for completely new, unseen data.

Supervised Machine Learning Algorithms

Supervised learning can be further divided into several different types, each with its own unique characteristics and applications. Here are some of the most common types of supervised learning algorithms:

- **Linear Regression:** Linear regression is a type of supervised learning regression algorithm that is used to predict a continuous output value. It is one of the simplest and most widely used algorithms in supervised learning.
- **Logistic Regression**: Logistic regression is a type of supervised learning classification algorithm that is used to predict a binary output variable.
- **Decision Trees** : Decision tree is a tree-like structure that is used to model decisions and their possible consequences. Each internal node in the tree represents a decision, while each leaf node represents a possible outcome.
- **Random Forests**: Random forests again are made up of multiple decision trees that work together to make predictions. Each tree in the forest is trained on a different subset of the input features and data. The final prediction is made by aggregating the predictions of all the trees in the forest.
- **Support Vector Machine(SVM):** The SVM algorithm creates a hyperplane to segregate n-dimensional space into classes and identify the correct category of new data points. The extreme cases that help create the hyperplane are called support vectors, hence the name Support Vector Machine.

- **K-Nearest Neighbors:** KNN works by finding k training examples closest to a given input and then predicts the class or value based on the majority class or average value of these neighbors. The performance of KNN can be influenced by the choice of k and the distance metric used to measure proximity.
- **Gradient Boosting:** Gradient Boosting combines weak learners, like decision trees, to create a strong model. It iteratively builds new models that correct errors made by previous ones.
- **Naive Bayes Algorithm:** The Naive Bayes algorithm is a supervised machine learning algorithm based on applying Bayes' Theorem with the “naive” assumption that features are independent of each other given the class label.

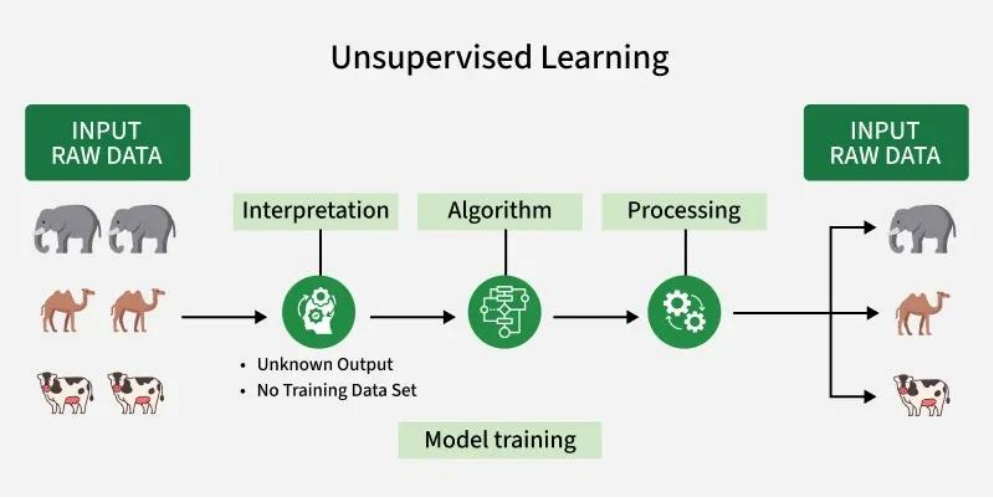
Practical Examples of Supervised learning

Few practical examples of supervised machine learning across various industries:

- **Fraud Detection in Banking**
- **Parkinson Disease Prediction**
- **Customer Churn Prediction**
- **Cancer cell classification**
- **Stock Price Prediction**

2. Unsupervised learning:

Unsupervised learning is a type of machine learning that analyzes and models data without labelled responses or predefined categories. Unlike supervised learning, where the algorithm learns from input-output pairs, unsupervised learning algorithms work solely with input data and aim to discover hidden patterns, structures or relationships within the dataset independently, without any human intervention or prior knowledge of the data's meaning.



Unsupervised Learning

The image shows set of animals like elephants, camels and cows that represents raw data that the unsupervised learning algorithm will process.

- The "Interpretation" stage signifies that the algorithm doesn't have predefined labels or categories for the data. It needs to figure out how to group or organize the data based on inherent patterns.
- An algorithm represents unsupervised learning process which can be clustering, dimensionality reduction or anomaly detection to identify patterns in the data.
- The processing stage shows the algorithm working on the data.

The output shows the results of the unsupervised learning process. In this case, the algorithm might have grouped the animals into clusters based on their species (elephants, camels, cows).

Working of Unsupervised Learning

The working of unsupervised machine learning can be explained in these steps:

1. Collect Unlabeled Data

- Gather a dataset without predefined labels or categories.
- **Example**: Images of various animals without any tags.

2. Select an Algorithm

- Choose a suitable unsupervised algorithm such as clustering like K-Means, association rule learning like Apriori or dimensionality reduction like PCA based on the goal.

3. Train the Model on Raw Data

- Feed the entire unlabeled dataset to the algorithm.
- The algorithm looks for similarities, relationships or hidden structures within the data.

4. Group or Transform Data

- The algorithm organizes data into groups (clusters), rules or lower-dimensional forms without human input.
- Example: It may group similar animals together or extract key patterns from large datasets.

5. Interpret and Use Results

- Analyze the discovered groups, rules or features to gain insights or use them for further tasks like visualization, anomaly detection or as input for other models.

Unsupervised Learning Algorithms

There are mainly 3 types of Unsupervised Algorithms that are used:

1. Clustering Algorithms

Clustering is an unsupervised machine learning technique that groups unlabeled data into clusters based on similarity. Its goal is to discover patterns or relationships within the data without any prior knowledge of categories or labels.

- Groups data points that share similar features or characteristics.
- Helps find natural groupings in raw, unclassified data.
- Commonly used for customer segmentation, anomaly detection and data organization.
- Works purely from the input data without any output labels.
- Enables understanding of data structure for further analysis or decision-making.

Some common clustering algorithms:

- ***K-means Clustering:*** *Groups data into K clusters based on how close the points are to each other.*
- ***Hierarchical Clustering:*** *Creates clusters by building a tree step-by-step, either merging or splitting groups.*
- ***Density-Based Clustering (DBSCAN):*** *Finds clusters in dense areas and treats scattered points as noise.*
- ***Mean-Shift Clustering:*** *Discovers clusters by moving points toward the most crowded areas.*
- ***Spectral Clustering:*** *Groups data by analyzing connections between points using graphs.*

2. Association Rule Learning

Association rule learning is a rule-based unsupervised learning technique used to discover interesting relationships between variables in large datasets. It identifies patterns in the form of "if-then" rules, showing how the presence of some items in the data implies the presence of others.

- Finds frequent item combinations and the rules connecting them.
- Commonly used in market basket analysis to understand product purchase relationships.
- Helps retailers design promotions and cross-selling strategies.

Some common Association Rule Learning algorithms:

- ***Apriori Algorithm:*** *Finds patterns by exploring frequent item combinations step-by-step.*
- ***FP-Growth Algorithm:*** *An Efficient Alternative to Apriori. It quickly identifies frequent patterns without generating candidate sets.*
- ***Eclat Algorithm:*** *Uses intersections of itemsets to efficiently find frequent patterns.*
- ***Efficient Tree-based Algorithms:*** *Scales to handle large datasets by organizing data in tree structures.*

3. Dimensionality Reduction

Dimensionality reduction is the process of decreasing the number of features or variables in a dataset while retaining as much of the original information as possible. This technique helps simplify complex data making it easier to analyze and visualize. It also improves the efficiency and performance of machine learning algorithms by reducing noise and computational cost.

- It reduces the dataset's feature space from many dimensions to fewer, more meaningful ones.
- Helps focus on the most important traits or patterns in the data.
- Commonly used to improve model speed and reduce overfitting.

Here are some popular Dimensionality Reduction algorithms:

- ***Principal Component Analysis (PCA):*** *Reduces dimensions by transforming data into uncorrelated principal components.*
- ***Linear Discriminant Analysis (LDA):*** *Reduces dimensions while maximizing class separability for classification tasks.*
- ***Non-negative Matrix Factorization (NMF):*** *Breaks data into non-negative parts to simplify representation.*
- ***Locally Linear Embedding (LLE)****: Reduces dimensions while preserving the relationships between nearby points.*
- ***Isomap****: Captures global data structure by preserving distances along a manifold.*

Applications of Unsupervised learning

Unsupervised learning has diverse applications across industries and domains. Key applications include:

- **Customer Segmentation**
- **Anomaly Detection**
- **Recommendation Systems**
- **Image and Text Clustering**
- **Social Network Analysis**

3. Reinforcement Learning

Reinforcement Learning (RL) trains an agent to make decisions by interacting with an environment. Instead of being told the correct answers, agent learns by trial and error method and gets rewards for good actions and penalties for bad ones. Over time it develops a strategy to maximize rewards and achieve goals. This approach is good for problems having sequential decision making such as robotics, gaming and autonomous systems.

Example: While Identifying a Fruit, system receives an input for example an apple and initially makes an incorrect prediction like "It's a mango". Feedback is provided to correct the error "Wrong! It's an apple" and the system updates its model based on this feedback.
Over time it learns to respond correctly that "It's an apple" when getting similar inputs and also improves accuracy.

Reinforcement Learning

Reinforcement Learning revolves around the idea that an agent (the learner or decision-maker) interacts with an environment to achieve a goal. The agent performs actions and receives feedback to optimize its decision-making over time.

- **Agent**: The decision-maker that performs actions.
- **Environment**: The world or system in which the agent operates.
- **State**: The situation or condition the agent is currently in.
- **Action**: The possible moves or decisions the agent can make.
- **Reward**: The feedback or result from the environment based on the agent's action.

Core Components

Let's see the core components of Reinforcement Learning

1. Policy

- Defines the agent's behavior i.e maps states for actions.
- Can be simple rules or complex computations.
- **Example**: An autonomous car maps pedestrian detection to make necessary stops.

2. Reward Signal

- Represents the goal of the RL problem.
- Guides the agent by providing feedback (positive/negative rewards).
- **Example**: For self-driving cars rewards can be fewer collisions, shorter travel time, lane discipline.

3. Value Function

- Evaluates long-term benefits, not just immediate rewards.
- Measures desirability of a state considering future outcomes.
- **Example**: A vehicle may avoid reckless maneuvers (short-term gain) to maximize overall safety and efficiency.

4. Model

- Simulates the environment to predict outcomes of actions.
- Enables planning and foresight.
- **Example**: Predicting other vehicles' movements to plan safer routes.

Working of Reinforcement Learning

The agent interacts iteratively with its environment in a feedback loop:

- The agent observes the current state of the environment.
- It chooses and performs an action based on its policy.
- The environment responds by transitioning to a new state and providing a reward (or penalty).
- The agent updates its knowledge (policy, value function) based on the reward received and the new state.
- This cycle repeats with the agent balancing exploration (trying new actions) and exploitation (using known good actions) to maximize the cumulative reward over time.

This process is mathematically framed as a Markov Decision Process (MDP) where future states depend only on the current state and action, not on the prior sequence of events.

Application of Reinforcement Learning

- Robotics
- Games
- Industrial Control
- Personalized Training Systems

Benefits of Machine Learning

- Enhanced Efficiency and Automation: ML automates repetitive tasks, freeing up human resources for more complex work. This leads to faster, smoother processes and higher productivity.
- Data-Driven Insights: It can analyze large amounts of data to identify patterns and trends that might be missed by people and help businesses make better decisions.
- Improved Personalization: It customizes user experiences by tailoring recommendations and ads based on individual preferences.
- Advanced Automation and Robotics: It helps robots and machines to perform complex tasks with greater accuracy and adaptability. This is transforming industries like manufacturing and logistics.

Challenges of Machine Learning

- Data Bias and Fairness: ML models learn from training data and if the data is biased, model's decisions can be unfair so it's important to select and monitor data carefully.
- Security and Privacy Concerns: Since it depends on large amounts of data, there is a risk of sensitive information being exposed so protecting privacy is important.
- Interpretability and Explainability: Complex ML models can be difficult to understand which makes it difficult to explain why they make certain decisions. This can affect trust and accountability.
- Job Displacement and Automation: Automation may replace some jobs so retraining and helping workers learn new skills is important to adapt to these changes.
- Applications of Machine Learning
- Machine Learning is used in many industries to solve problems and improve services. Here are some common real-world applications:
- Healthcare: It helps doctors to diagnose diseases from medical images like X-rays and MRIs. It also predicts patient outcomes and personalizes treatments which improves healthcare quality.
- Finance: In finance it detects fraudulent transactions in real time and supports algorithmic trading. It also helps to assess credit risk helps in making lending safer and faster.
- Retail and E-Commerce: It helps in personalized product recommendations and forecasts demand to optimize inventory and also analyzes customer sentiment to improve shopping experiences.
- Transportation and Automotive: Self-driving cars rely on ML to navigate and make decisions. It optimizes delivery routes and predicts vehicle maintenance needs which reduces downtime.
- Social Media and Entertainment: Platforms like Netflix and YouTube use ML to recommend content we'll enjoy. It enables image and speech recognition for better user interaction.

- Manufacturing: It improves quality control by detecting defects in products automatically and predicts machine failures in advance and helps in production processes.

Hypothesis in Machine Learning

The concept of a hypothesis is fundamental in Machine Learning and data science endeavours. In the realm of machine learning, a hypothesis serves as an initial assumption made by data scientists and ML professionals when attempting to address a problem. Machine learning involves conducting experiments based on past experiences, and these hypotheses are crucial in formulating potential solutions.

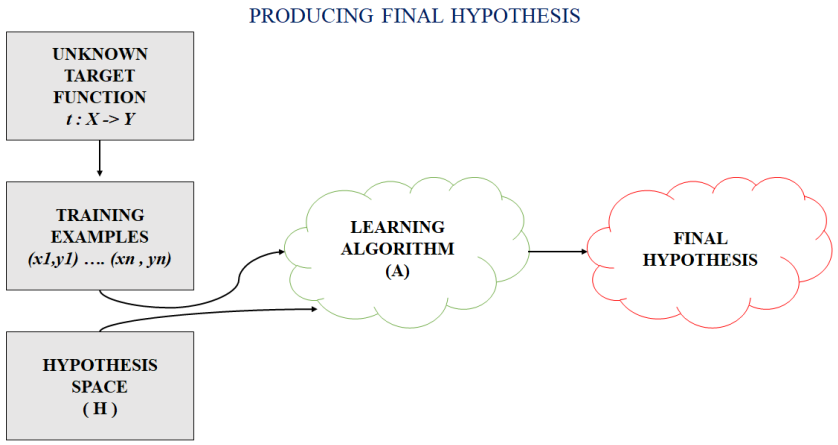
It's important to note that in machine learning discussions, the terms "hypothesis" and "model" are sometimes used interchangeably. However, a hypothesis represents an assumption, while a model is a mathematical representation employed to test that hypothesis. This section on "Hypothesis in Machine Learning" explores key aspects related to hypotheses in machine learning and their significance.

Hypothesis in Machine Learning

A hypothesis in machine learning is the model's presumption regarding the connection between the input features and the result. It is an illustration of the mapping function that the algorithm is attempting to discover using the training set. To minimize the discrepancy between the expected and actual outputs, the learning process involves modifying the weights that parameterize the hypothesis. The objective is to optimize the model's parameters to achieve the best predictive performance on new, unseen data, and a cost function is used to assess the hypothesis' accuracy.

How does a Hypothesis work?

In most supervised machine learning algorithms, our main goal is to find a possible hypothesis from the hypothesis space that could map out the inputs to the proper outputs. The following figure shows the common method to find out the possible hypothesis from the Hypothesis space:



Hypothesis Space (H)

Hypothesis space is the set of all the possible legal hypothesis. This is the set from which the machine learning algorithm would determine the best possible (only one) which would best describe the target function or the outputs.

Hypothesis (h)

A hypothesis is a function that best describes the target in supervised machine learning. The hypothesis that an algorithm would come up depends upon the data and also depends upon the restrictions and bias that we have imposed on the data.

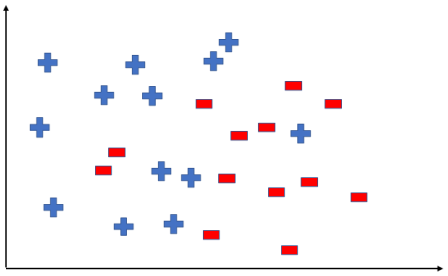
The Hypothesis can be calculated as:

$$y = mx + b$$

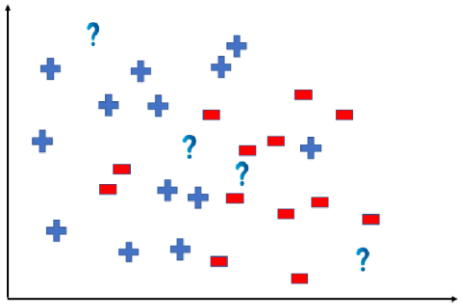
Where,

- y = range
- m = slope of the lines
- x = domain
- b = intercept

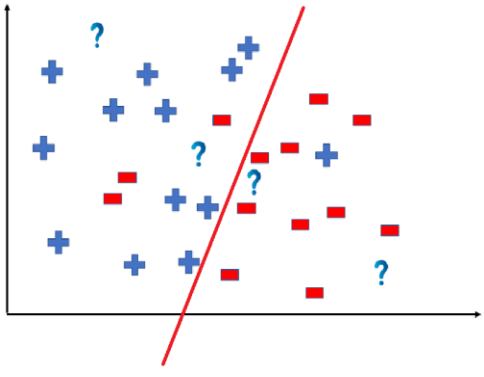
To better understand the Hypothesis Space and Hypothesis consider the following coordinate that shows the distribution of some data:



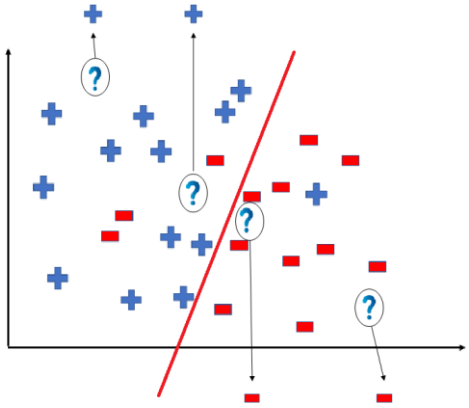
Say suppose we have test data for which we have to determine the outputs or results. The test data is as shown below:



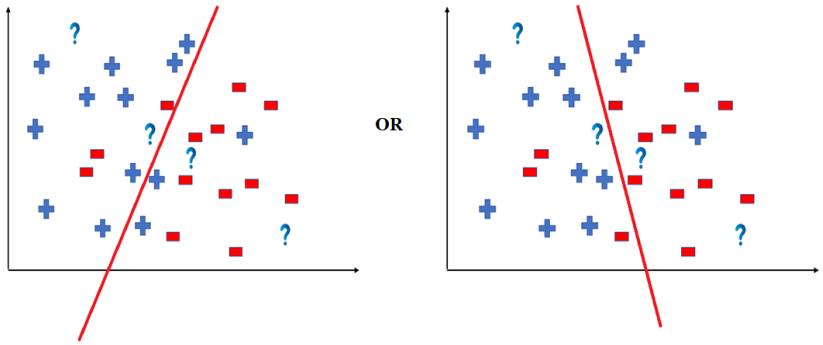
We can predict the outcomes by dividing the coordinate as shown below:



So the test data would yield the following result:



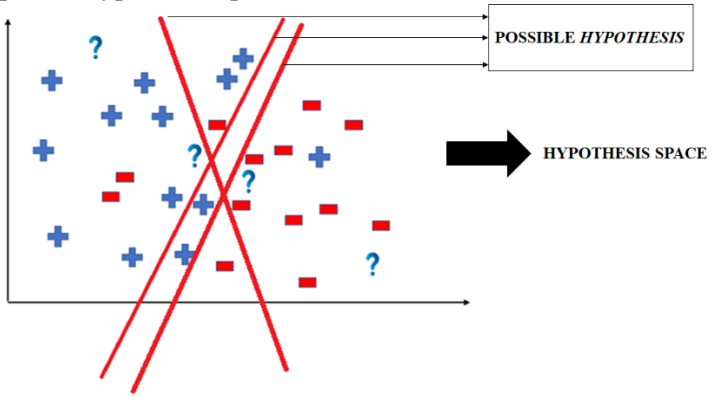
But note here that we could have divided the coordinate plane as:



The way in which the coordinate would be divided depends on the data, algorithm and constraints.

- All these legal possible ways in which we can divide the coordinate plane to predict the outcome of the test data composes of the Hypothesis Space.
- Each individual possible way is known as the hypothesis.

Hence, in this example the hypothesis space would be like:



Hypothesis Space and Representation in Machine Learning

The hypothesis space comprises all possible legal hypotheses that a machine learning algorithm can consider. Hypotheses are formulated based on various algorithms and techniques, including linear regression, decision trees, and neural networks. These hypotheses capture the mapping function transforming input data into predictions.

Hypothesis Formulation and Representation in Machine Learning

Hypotheses in machine learning are formulated based on various algorithms and techniques, each with its representation. For example:

- **Linear Regression:** $h(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots + \theta_n X_n$
- **Decision Trees:** $h(X) = \text{Tree}(X)$
- **Neural Networks:** $h(X) = \text{NN}(X)$

In the case of complex models like neural networks, the hypothesis may involve multiple layers of interconnected nodes, each performing a specific computation.

Hypothesis Evaluation:

The process of machine learning involves not only formulating hypotheses but also evaluating their performance. This evaluation is typically done using a loss function or an evaluation metric that quantifies the disparity between predicted outputs and ground truth labels. Common evaluation metrics include mean squared error (MSE), accuracy, precision, recall, F1-score, and others. By comparing the predictions of the hypothesis with the actual outcomes on a validation or test dataset, one can assess the effectiveness of the model.

Hypothesis Testing and Generalization:

Once a hypothesis is formulated and evaluated, the next step is to test its generalization capabilities. Generalization refers to the ability of a model to make accurate predictions on unseen data. A hypothesis that performs well on the training dataset but fails to generalize to new instances is said to suffer from overfitting. Conversely, a hypothesis that generalizes well to unseen data is deemed robust and reliable.

The process of hypothesis formulation, evaluation, testing, and generalization is often iterative in nature. It involves refining the hypothesis based on insights gained from model performance, feature importance, and domain knowledge. Techniques such as hyperparameter tuning, feature engineering, and model selection play a crucial role in this iterative refinement process.

Hypothesis in Statistics

In statistics, a hypothesis refers to a statement or assumption about a population parameter. It is a proposition or educated guess that helps guide statistical analyses. There are two types of hypotheses: the null hypothesis (H0) and the alternative hypothesis (H1 or Ha).

- **Null Hypothesis(H0):** This hypothesis suggests that there is no significant difference or effect, and any observed results are due to chance. It often represents the status quo or a baseline assumption.
- **Aternative Hypothesis(H1 or Ha):** This hypothesis contradicts the null hypothesis, proposing that there is a significant difference or effect in the population. It is what researchers aim to support with evidence.

What is Inductive Bias in Machine Learning?

In the realm of machine learning, the concept of inductive bias plays a pivotal role in shaping how algorithms learn from data and make predictions. It serves as a guiding principle that helps algorithms generalize from the training data to unseen data, ultimately influencing their performance and decision-making processes. In this article, we delve into the intricacies of inductive bias, its significance in machine learning, and its implications for model development and interpretation.

What is Inductive Bias?

Inductive bias can be defined as the set of assumptions or biases that a learning algorithm employs to make predictions on unseen data based on its training data. These assumptions are inherent in the algorithm's design and serve as a foundation for learning and generalization.

The inductive bias of an algorithm influences how it selects a hypothesis (a possible explanation or model) from the hypothesis space (the set of all possible hypotheses) that best fits the training data. It helps the algorithm navigate the trade-off between fitting the training data perfectly (overfitting) and generalizing well to unseen data (underfitting).

Types of Inductive Bias

- Inductive bias can manifest in various forms, depending on the algorithm and its underlying assumptions. Some common types of inductive bias include:
- Bias towards simpler explanations: Many machine learning algorithms, such as decision trees and linear models, have a bias towards simpler hypotheses. They prefer explanations that are more parsimonious and less complex, as these are often more likely to generalize well to unseen data.
- Bias towards smoother functions: Algorithms like kernel methods or Gaussian processes have a bias towards smoother functions. They assume that neighboring points in the input space should have similar outputs, leading to smooth decision boundaries.
- Bias towards specific types of functions: Neural networks, for example, have a bias towards learning complex, nonlinear functions. This bias allows them to capture intricate patterns in the data but can also lead to overfitting if not regularized properly.
- Bias towards sparsity: Some algorithms, like Lasso regression, have a bias towards sparsity. They prefer solutions where only a few features are relevant, which can improve interpretability and generalization.

Importance of Inductive Bias

Inductive bias is crucial in machine learning as it helps algorithms generalize from limited training data to unseen data. Without a well-defined inductive bias, algorithms may struggle to make accurate predictions or may overfit the training data, leading to poor performance on new data.

Understanding the inductive bias of an algorithm is essential for model selection, as different biases may be more suitable for different types of data or tasks. It also provides insights into how the algorithm is learning and what assumptions it is making about the data, which can aid in interpreting its predictions and results.

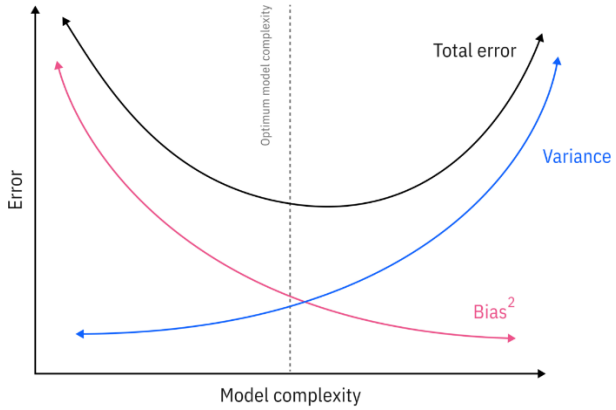
Challenges and Considerations

While inductive bias is essential for learning, it can also introduce limitations and challenges. Biases that are too strong or inappropriate for the data can lead to poor generalization or biased predictions. Balancing bias with variance (the variability of predictions) is a key challenge in machine learning, requiring careful tuning and model selection.

Additionally, the choice of inductive bias can impact the interpretability of the model. Simpler biases may lead to more interpretable models, while more complex biases may sacrifice interpretability for improved performance.

Overfitting vs. underfitting

When data scientists and engineers train machine learning (ML) models, they risk using an algorithm that is too simple to capture the underlying patterns in the data, leading to underfitting, or one that is too complex, leading to overfitting. Managing overfitting and underfitting is a core challenge in data science workflows and developing reliable artificial intelligence (AI) systems.



Bias and variance in machine learning

Bias and variance explain the balance engineers need to strike to help ensure a good fit in their machine learning models. As such, the bias-variance tradeoff is central to addressing underfitting and overfitting.

A biased model makes strong assumptions about the training data to simplify the learning process, ignoring subtleties or complexities it cannot account for. Variance refers to the model's sensitivity to learning fluctuations in the training data.

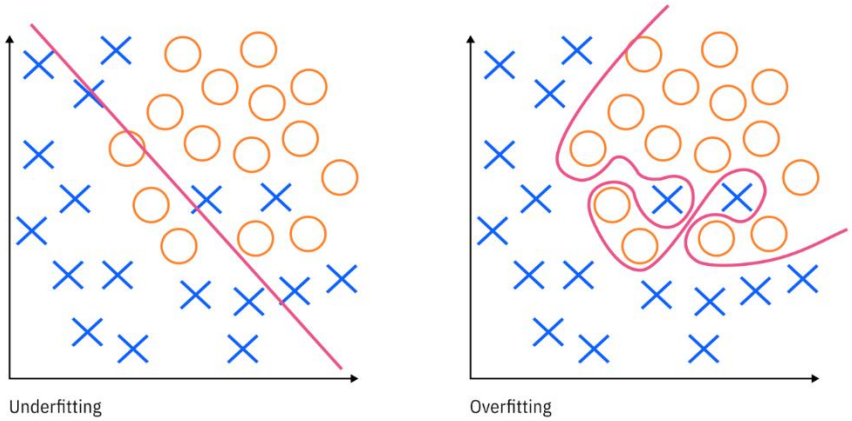
Examples of high-bias models include linear regression algorithms or shallow decision trees, which assume simple linear or binary relationships even when the data patterns are more complex.

Using a linear regression model for data with a quadratic relationship will result in underfitting because the linear model cannot capture the inherent curvature. As a result, the model performs poorly on the training set and unseen test data because it cannot generalize well to new data.

Generalization is the model's ability to understand and apply learned patterns to unseen data. Models with low variance also tend to underfit as they are too simple to capture complex patterns. However, low-bias models might overfit if they are too flexible.

High variance indicates that the model might capture noise, idiosyncrasies and random details within the training data. High-variance models are overly flexible, resulting in low training error, but when tested on new data, the learned patterns fail to generalize, leading to high test error.

Imagine memorizing answers for a test instead of understanding the concepts needed to get the answers yourself. If the test differs from what was studied, you will struggle to answer the questions. Striking the balance between variance and bias is key to achieving optimal performance in machine learning models.

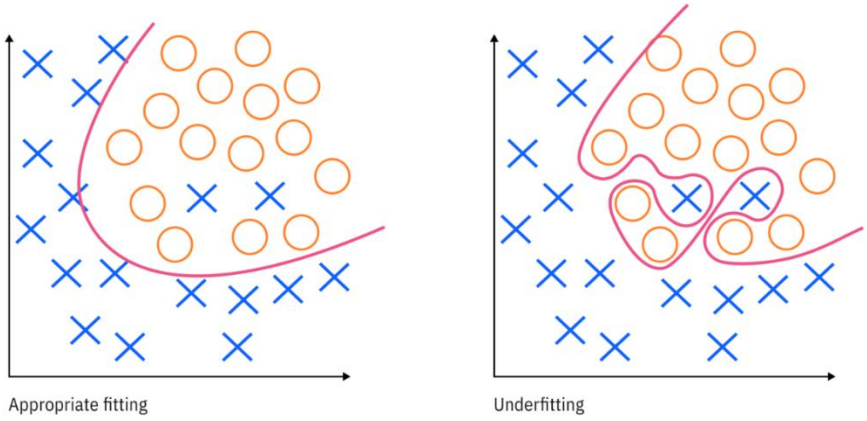


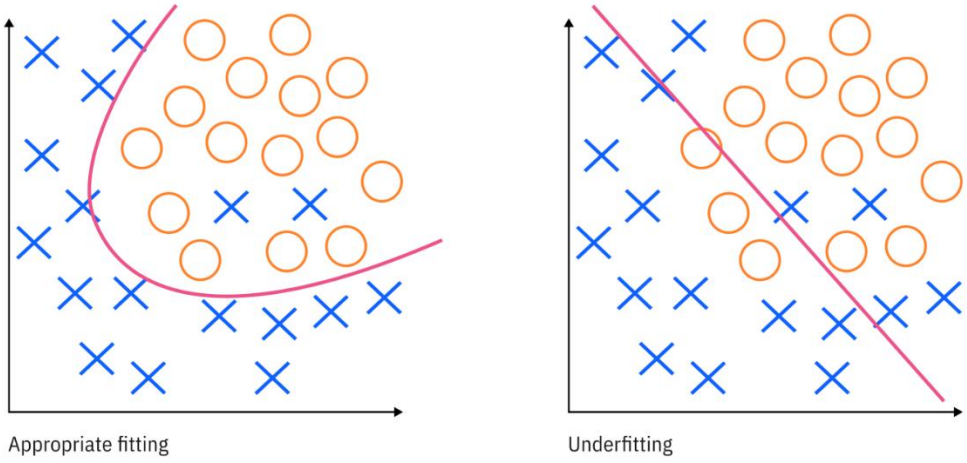
How to recognize overfitting and underfitting

The rules

- **Overfitting**: Training error is low, but testing error is significantly higher.
- **Underfitting**: Errors are consistently high across training and testing data sets.

- An **overfit model** can result in high model accuracy on training data but low accuracy on new data due to memorization instead of generalization. Overfitting happens when engineers use a machine learning model with too many parameters or layers, such as a deep learning neural network, making it highly adaptable to the training data.
- When trained on a small or noisy data set, the model risks memorizing specific data points and noise rather than learning the general patterns. If the data contains errors or inconsistencies, the model might incorrectly learn these as meaningful patterns.
- Engineers look for a performance gap between training and testing, but they can also detect overfitting in learning curves, where training loss decreases toward zero while validation loss increases, indicating poor generalization.
- Another sign of an overfit model is its decision boundaries, the model's learned rules for classifying data points. The decision boundary becomes overly complex and erratic in overfit models, as it adapts to noise in the training set rather than capturing true underlying structures, further indicating overfitting.



- In addition, high-dimensional data sets can lead to overfitting due to the "curse of dimensionality." As the number of features increases, data points become sparse, making it harder for models to find meaningful patterns, increasing variance and the risk of overfitting.
- An **underfit model** performs poorly on training data and testing data because it fails to capture the dominant patterns in the data set. Engineers typically identify underfitting through consistently poor performance across both data sets.
- Underfit models also tend to show high errors in learning curves, return suboptimal evaluation metrics and exhibit systematic residual patterns, all of which indicate an inability to learn the underlying relationships in the data effectively.
- Underfitting in machine learning often occurs due to simplistic models, poor feature engineering or excessive regularization that overly restricts the model's flexibility. Similarly, poor feature selection—such as omitting interaction terms or polynomial features—can prevent the model from understanding hidden relationships in the data. Inadequate preprocessing, insufficient training time or a lack of sufficient data to train the model can also contribute to underfitting.
- 

Evaluation: How Good is Our Model?

Once our algorithm finds a hypothesis (h*), we need to know if it's actually useful. This is **Evaluation**.

1. The Problem: Overfitting

We want a model that performs well on **new, unseen data**, not just the data it was trained on.

- **Overfitting** occurs when a model learns the training data *too well*, including the noise and random fluctuations. It's like a student who memorizes the answers to a test but doesn't understand the concept.
- **Result:** High accuracy on the training data, but **poor accuracy** on new, real-world data.

2. Splitting the Data

To check for overfitting and evaluate the true performance, we **split our original dataset** into three parts:

Dataset	Percentage	Purpose
Training Set	70% - 80%	Used to **train** the model (find h*).
Validation Set	10% - 15%	Used to **tune** the model's parameters (hyperparameters).
Test Set	10% - 15%	Used to provide a **final, unbiased** evaluation of the chosen model. **The model never sees this data during training or tuning.**

3. Key Evaluation Metrics

The most common ways to measure performance depend on the type of problem:

- **For Classification:**
  - **Accuracy:** The percentage of correct predictions out of all predictions.
  - **Precision and Recall:** Measures of how relevant the results are and how many relevant results are found.
- **For Regression:**
  - **Mean Squared Error (MSE):** Measures the average squared difference between the model's prediction and the actual value. Lower MSE is better.

Training, validation, and test data sets

In machine learning, a common task is the study and construction of algorithms that can learn from and make predictions on data. Such algorithms function by making data-driven predictions or decisions, through building a mathematical model from input data. These input data used to build the model are usually divided into multiple data sets. In particular, three data sets are commonly used in different stages of the creation of the model: training, validation, and testing sets.

The model is initially fit on a **training data set**, which is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a naive Bayes classifier) is trained on the training data set using a supervised learning method, for example using optimization methods such as gradient descent or stochastic gradient descent. In practice, the training data set often consists of pairs of an input vector (or scalar) and the corresponding output vector (or scalar), where the answer key is commonly denoted as the *target* (or *label*). The current model is run with the training data set and produces a result, which is then compared with the *target*, for each input vector in the training data set. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Successively, the fitted model is used to predict the responses for the observations in a second data set called the **validation data set**. The validation data set provides an unbiased evaluation of a model fit on the training data set while tuning the model's hyperparameters (e.g. the number of hidden units—layers and layer widths—in a neural network). Validation data sets can be used for regularization by early stopping (stopping training when the error on the validation data set increases, as this is a sign of over-fitting to the training data set). This simple procedure is complicated in practice by the fact that the validation data set's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when over-fitting has truly begun.

Finally, the **test data set** is a data set used to provide an unbiased evaluation of a *final* model fit on the training data set. If the data in the test data set has never been used in training (for example in cross-validation), the test data set is also called a **holdout data set**. The term "validation set" is sometimes used instead of "test set" in some literature (e.g., if the original data set was partitioned into only two subsets, the test set might be referred to as the validation set).

Deciding the sizes and strategies for data set division in training, test and validation sets is very dependent on the problem and data available.

Training data set:

A training data set is a data set of examples used during the learning process and is used to fit the parameters (e.g., weights) of, for example, a classifier.

For classification tasks, a supervised learning algorithm looks at the training data set to determine, or learn, the optimal combinations of variables that will generate a good predictive model. The goal is to produce a trained (fitted) model that generalizes well to new, unknown data. The fitted model is evaluated using “new” examples from the held-out data sets (validation and test data sets) to estimate the model’s accuracy in classifying new data. To reduce the risk of issues such as over-fitting, the examples in the validation and test data sets should not be used to train the model.

Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

When a training set is continuously expanded with new data, then this is incremental learning.

Validation data set:

A validation data set is a data set of examples used to tune the hyperparameters (i.e. the architecture) of a model. It is sometimes also called the development set or the "dev set". An example of a hyperparameter for artificial neural networks includes the number of hidden units in each layer. It, as well as the testing set (as mentioned below), should follow the same probability distribution as the training data set.

In order to avoid overfitting, when any classification parameter needs to be adjusted, it is necessary to have a validation data set in addition to the training and test data sets. For example, if the most suitable classifier for the problem is sought, the training data set is used to train the different candidate classifiers, the validation data set is used to compare their performances and decide which one to take and, finally, the test data set is used to obtain the performance characteristics such as accuracy, sensitivity, specificity, F-measure, and so on. The validation data set functions as a hybrid: it is training data used for testing, but neither as part of the low-level training nor as part of the final testing.

The basic process of using a validation data set for model selection (as part of training data set, validation data set, and test data set) is:

Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected. This approach is called the hold out method. Since this

procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a test set.

An application of this process is in early stopping, where the candidate models are successive iterations of the same network, and training stops when the error on the validation set grows, choosing the previous model (the one with minimum error).

Test data set:

A test data set is a data set that is independent of the training data set, but that follows the same probability distribution as the training data set. If a model fit to the training data set also fits the test data set well, minimal overfitting has taken place (see figure below). A better fitting of the training data set as opposed to the test data set usually points to over-fitting.

A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier. To do this, the final model is used to predict classifications of examples in the test set. Those predictions are compared to the examples' true classifications to assess the model's accuracy.

In a scenario where both validation and test data sets are used, the test data set is typically used to assess the final model that is selected during the validation process. In the case where the original data set is partitioned into two subsets (training and test data sets), the test data set might assess the model only once (e.g., in the holdout method). Note that some sources advise against such a method. However, when using a method such as cross-validation, two partitions can be sufficient and effective since results are averaged after repeated rounds of model training and testing to help reduce bias and variability.

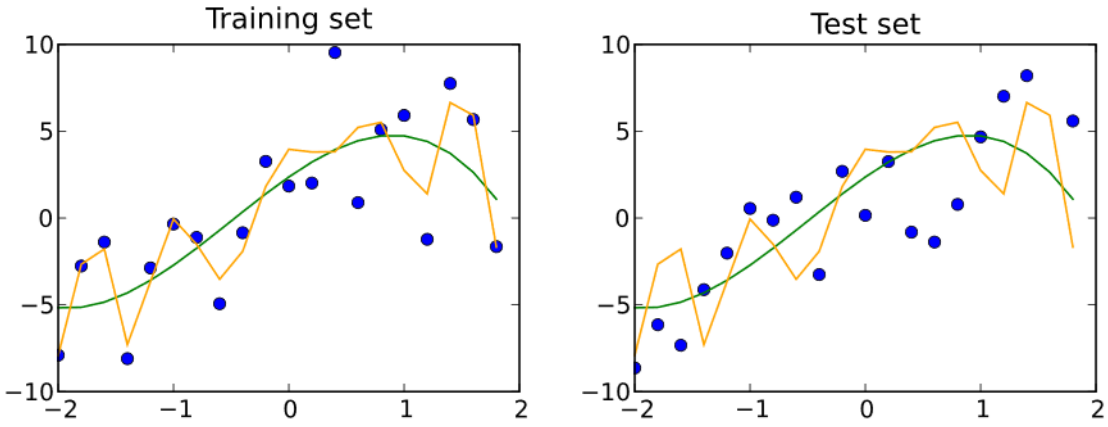


Figure : A training set (left) and a test set (right) from the same statistical population are shown as blue points. Two predictive models are fit to the training data. Both fitted models are plotted with both the training and test sets. In the training set, the MSE of the fit shown in orange is 4 whereas the MSE for the fit shown in green is 9. In the test set, the MSE for the fit shown in orange is 15 and the MSE for the fit shown in green is 13. The orange curve severely overfits the training data, since its MSE increases by almost a factor of four when comparing the test set to the training set. The green curve overfits the training data much less, as its MSE increases by less than a factor of 2.

Linear regression

Linear regression is a type of **supervised machine-learning algorithm** that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.

For example we want to predict a student's exam score based on how many hours they studied. We observe that as students study more hours, their scores go up. In the example of predicting exam scores based on hours studied. Here

- **Independent variable (input):** Hours studied because it's the factor we control or observe.
- **Dependent variable (output):** Exam score because it depends on how many hours were studied.

We use the independent variable to predict the dependent variable.

Why Linear Regression is Important?

Here's why linear regression is important:

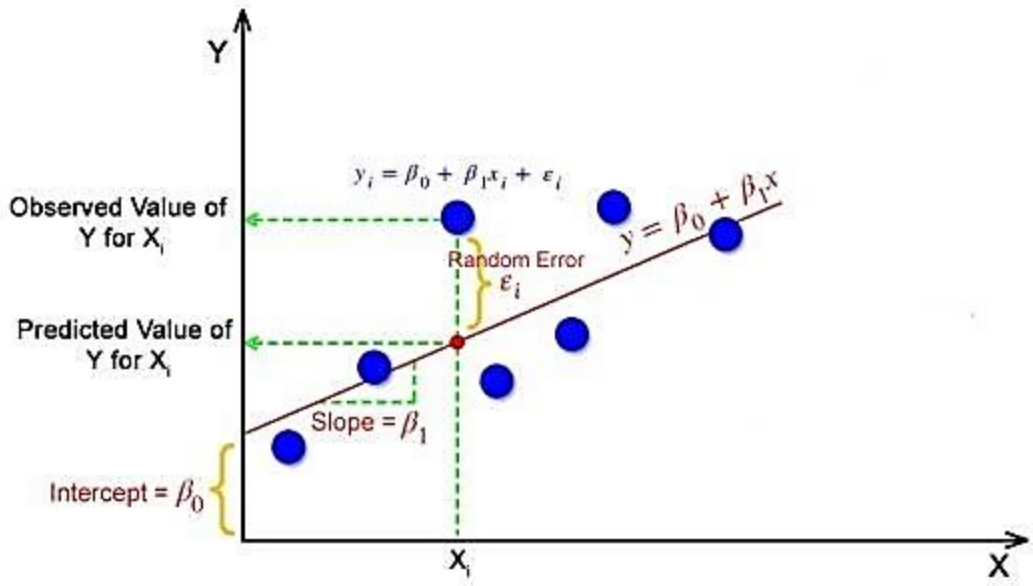
- **Simplicity and Interpretability:** It's easy to understand and interpret, making it a starting point for learning about machine learning.
- **Predictive Ability**: Helps predict future outcomes based on past data, making it useful in various fields like finance, healthcare and marketing.
- **Basis for Other Models:** Many advanced algorithms, like logistic regression or neural networks, build on the concepts of linear regression.
- **Efficiency:** It's computationally efficient and works well for problems with a linear relationship.
- **Widely Used:** It's one of the most widely used techniques in both statistics and machine learning for regression tasks.
- **Analysis:** It provides insights into relationships between variables (e.g., how much one variable influences another).

Best Fit Line in Linear Regression

In linear regression, the best-fit line is the straight line that most accurately represents the relationship between the independent variable (input) and the dependent variable (output). It is the line that minimizes the difference between the actual data points and the predicted values from the model.

1. Goal of the Best-Fit Line

The goal of linear regression is to find a straight line that minimizes the error (the difference) between the observed data points and the predicted values. This line helps us predict the dependent variable for new, unseen data.



Linear Regression

Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y. There are many types of functions or modules that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

2. Equation of the Best-Fit Line

For simple linear regression (with one independent variable), the best-fit line is represented by the equation

$$y = mx + b$$

Where:

- **y** is the predicted value (dependent variable)
- **x** is the input (independent variable)
- **m** is the slope of the line (how much y changes when x changes)
- **b** is the intercept (the value of y when x = 0)

The best-fit line will be the one that optimizes the values of m (slope) and b (intercept) so that the predicted y values are as close as possible to the actual data points.

3. Minimizing the Error: The Least Squares Method

To find the best-fit line, we use a method called **Least Squares**. The idea behind this method is to minimize the sum of squared differences between the actual values (data points) and the predicted values from the line. These differences are called residuals.

The formula for residuals is:

$$Residual = y_i - \hat{y}_i$$

Where:

- y_i is the actual observed value
- $\hat{y}_i$ is the predicted value from the line for that x_i

The least squares method minimizes the sum of the squared residuals:

$$Sum of squared errors (SSE) = \Sigma(y_i - \hat{y}_i)^2$$

This method ensures that the line best represents the data where the sum of the squared differences between the predicted values and actual values is as small as possible.

4. Interpretation of the Best-Fit Line

- **Slope (m):** The slope of the best-fit line indicates how much the dependent variable (y) changes with each unit change in the independent variable (x). For example if the slope is 5, it means that for every 1-unit increase in x, the value of y increases by 5 units.
- **Intercept (b):** The intercept represents the predicted value of y when x = 0. It's the point where the line crosses the y-axis.

In linear regression some hypothesis are made to ensure reliability of the model's results.

Limitations

- ***Assumes Linearity:*** *The method assumes the relationship between the variables is linear. If the relationship is non-linear, linear regression might not work well.*
- ***Sensitivity to Outliers:*** *Outliers can significantly affect the slope and intercept, skewing the best-fit line.*

Hypothesis function in Linear Regression

In linear regression, the hypothesis function is the equation used to make predictions about the dependent variable based on the independent variables. It represents the relationship between the input features and the target output.

For a simple case with one independent variable, the hypothesis function is:

$$h(x) = \beta_0 + \beta_1 x$$

Where:

- $h(x) (or \hat{y})$ is the predicted value of the dependent variable (y).
- x $x$is the independent variable.
- β_0 is the intercept, representing the value of y when x is 0.
- β_1 is the slope, indicating how much y changes for each unit change in x.

For **Multiple linear regression** (with more than one independent variable), the hypothesis function expands to:

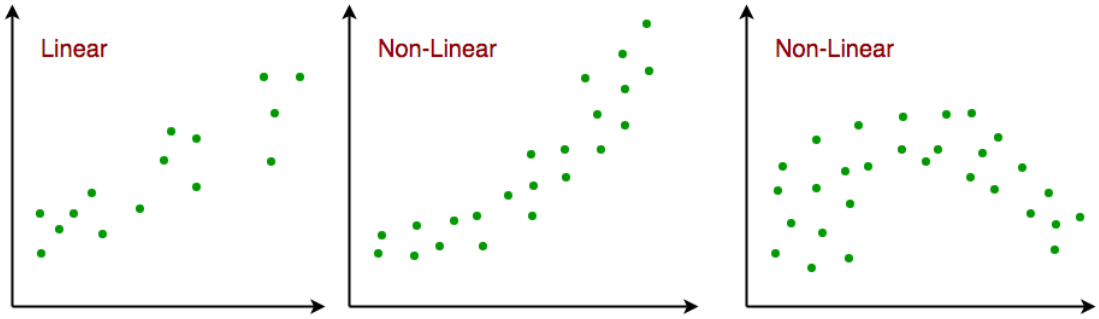
$$h(x_1, x_2, \ldots, x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Where:

- $x_1, x_2, \ldots, x_k$ are the independent variables.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients, representing the influence of each respective independent variable on the predicted output.

Assumptions of the Linear Regression

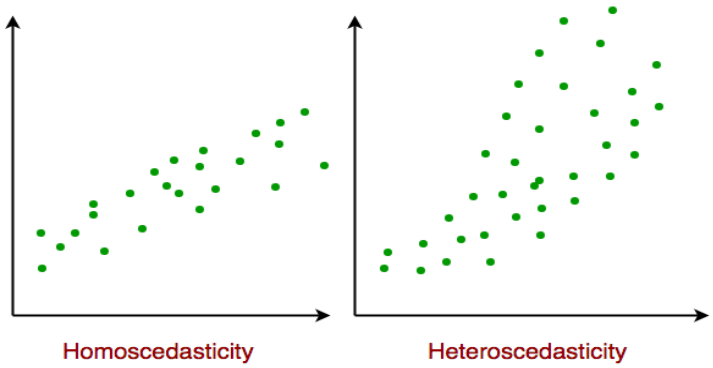
1. Linearity: The relationship between inputs (X) and the output (Y) is a straight line.



Linearity

2. Independence of Errors: The errors in predictions should not affect each other.

3. Constant Variance (Homoscedasticity): The errors should have equal spread across all values of the input. If the spread changes (like fans out or shrinks), it's called heteroscedasticity and it's a problem for the model.



4. Normality of Errors: The errors should follow a normal (bell-shaped) distribution.

5. No Multicollinearity *(*for multiple regression): Input variables shouldn't be too closely related to each other.

6. No Autocorrelation: Errors shouldn't show repeating patterns, especially in time-based data.

7. Additivity: The total effect on Y is just the sum of effects from each X, no mixing or interaction between them.'

.

Use Case of Multiple Linear Regression

Multiple linear regression allows us to analyze relationship between multiple independent variables and a single dependent variable. Here are some use cases:

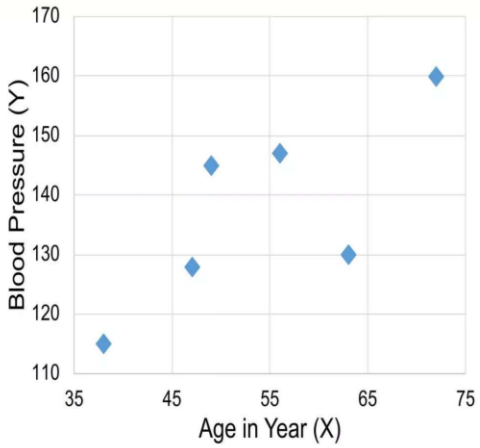
- **Real Estate Pricing:** In real estate MLR is used to predict property prices based on multiple factors such as location, size, number of bedrooms, etc. This helps buyers and sellers understand market trends and set competitive prices.
- **Financial Forecasting:** Financial analysts use MLR to predict stock prices or economic indicators based on multiple influencing factors such as interest rates, inflation rates and market trends. This enables better investment strategies and risk management24.
- **Agricultural Yield Prediction:** Farmers can use MLR to estimate crop yields based on several variables like rainfall, temperature, soil quality and fertilizer usage. This information helps in planning agricultural practices for optimal productivity
- **E-commerce Sales Analysis:** An e-commerce company can utilize MLR to assess how various factors such as product price, marketing promotions and seasonal trends impact sales.

Now that we have understood about linear regression, its assumption and its type now we will learn how to make a linear regression model.

Regression –Problem Formulation

Let you have given with a data:

Age in Years (X)	Blood Pressure (Y)
56	147
49	145
72	160
38	115
63	130
47	128



❖ For given example the Linear Regression is modeled as:

$$BloodPressure(y) = w_0 + w_1 AgeinYear(X)$$

OR

$$y = w_0 + w_1 X$$ – Equation of line

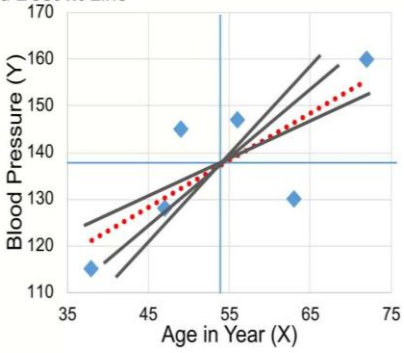
with w_0 *is intercept on* Y_axis *and* w_1 *is slope of line*

Blood Pressure - **Dependent Variable**

Age in Year - **Independent Variable**

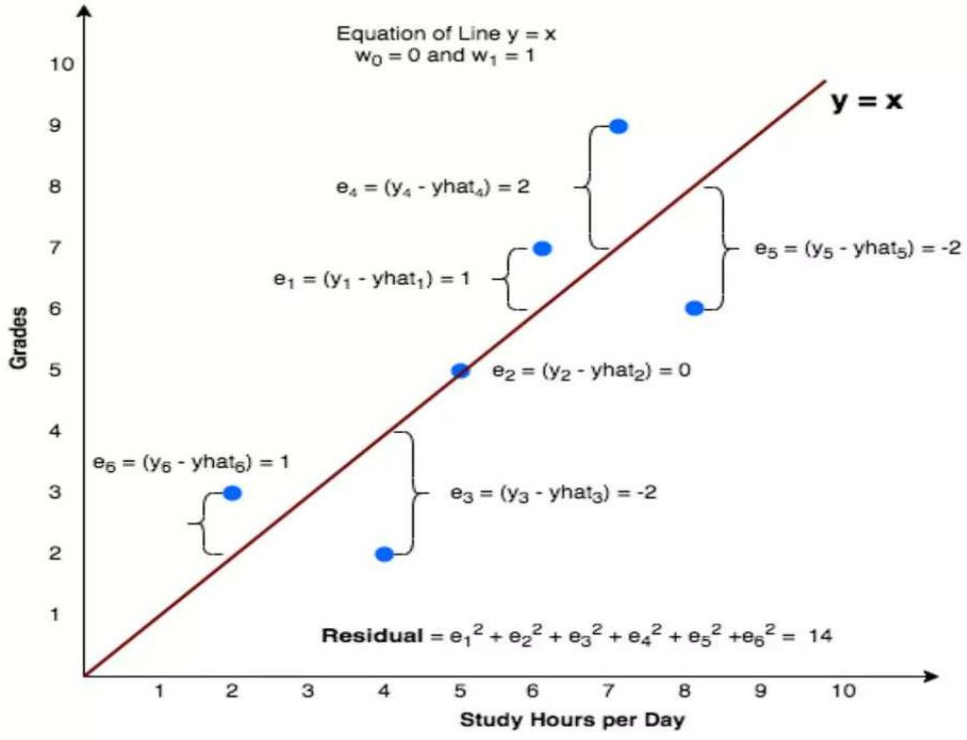
Linear Regression- Best Fit Line

- Regression uses line to show the trend of distribution.
- There can be many lines that try to fit the data points in scatter diagram
- The aim is to find **Best fit** Line



What is Best Fit Line

- Best fit line tries to explain the variance in given data. (minimize the total residual/error)



Polynomial Regression

Polynomial Regression is a form of linear regression where the relationship between the independent variable (x) and the dependent variable (y) is modelled as an n^{th} degree polynomial. It is useful when the data exhibits a non-linear relationship allowing the model to fit a curve to the data.

Need for Polynomial Regression

- **Non-linear Relationships:** Polynomial regression is used when the relationship between the independent variable (input) and dependent variable (output) is non-linear. Unlike linear regression which fits a straight line, it fits a polynomial equation to capture the curve in the data.
- **Better Fit for Curved Data:** When a researcher hypothesizes a curvilinear relationship, polynomial terms are added to the model. A linear model often results in residuals with noticeable patterns which shows a poor fit. It can capture these non-linear patterns effectively.
- **Flexibility and Complexity:** It does not assume all independent variables are independent. By introducing higher-degree terms, it allows for more flexibility and can model more complex, curvilinear relationships between variables.

How does a Polynomial Regression work?

Polynomial regression is an extension of linear regression where higher-degree terms are added to model non-linear relationships. The general form of the equation for a polynomial regression of degree n is:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon$$

where:

- y is the dependent variable.
- x is the independent variable.
- $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients of the polynomial terms.
- $n$is the degree of the polynomial.
- ϵ represents the error term.

The goal of regression analysis is to model the expected value of a dependent variable y in terms of an independent variable x. In simple linear regression, this relationship is modeled as:

$$y = a + bx + e$$

Here

- y is a dependent variable
- a is the y-intercept, b is the slope
- e is the error term

However in cases where the relationship between the variables is nonlinear such as modelling chemical synthesis based on temperature, a linear model may not be sufficient. Instead, we use polynomial regression which introduces higher-degree terms such as x^2 to better capture the relationship.
For example, a quadratic model can be written as:

$$y = a + b_1 x + b_2 x^2 + e$$

Here:

- y is the dependent variable on x
- $a$is the y-intercept and $e$is the error rate.

In general, **polynomial regression** can be extended to the nth degree:

$$\boldsymbol{y = a + b_1 x + b_2 x^2 + \ldots + b_n x^n}$$

While the regression function is linear in terms of the unknown coefficients $b_0, b_1, \ldots, b_n$, the model itself captures non-linear patterns in the data. The coefficients are estimated using techniques like Least Square technique to minimize the error between predicted and actual values.
Choosing the right polynomial degree n is important: a higher degree may fit the data more closely but it can lead to overfitting. The degree should be selected based on the complexity of the data. Once the model is trained, it can be used to make predictions on new data, capturing non-linear relationships and providing a more accurate model for real-world applications.

Real-Life Example for Polynomial Regression

Let's consider an example in the field of finance where we analyze the relationship between an employee's years of experience and their corresponding salary. If we check that the relationship might not be linear, polynomial regression can be used to model it more accurately.

Years of Experience	Salary (in dollars)
1	50,000
2	55,000
3	65,000
4	80,000
5	110,000
6	150,000
7	200,000

Example Data:

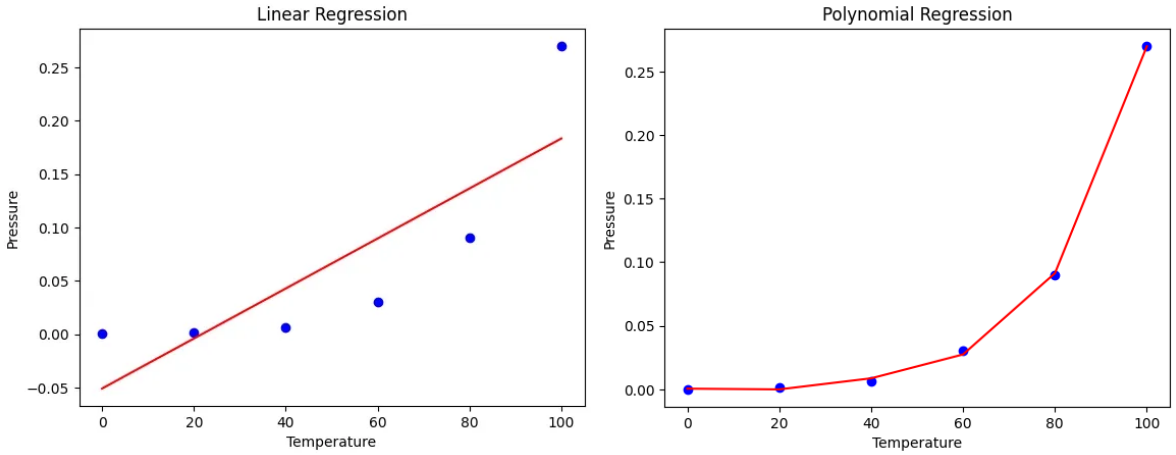
Now, let's apply polynomial regression to model the relationship between years of experience and salary. We'll use a quadratic polynomial (degree 2) which includes both linear and quadratic terms for better fit. The quadratic polynomial regression equation is:

$$\boldsymbol{Salary = \beta_0 + \beta_1 \times Experience + \beta_2 \times Experience^2 + \epsilon}$$

To find the coefficients $\beta_0, \beta_1, \beta_2$ that minimize the difference between the predicted and actual salaries, we can use the **Least Squares method.** The objective is to minimize the sum of squared differences between the predicted salaries and the actual data points which allows us to fit a model that captures the non-linear progression of salary with respect to experience.

Example 2:

	sno	Temperature	Pressure
0	1	0	0.0002
1	2	20	0.0012
2	3	40	0.0060
3	4	60	0.0300
4	5	80	0.0900
5	6	100	0.2700



Application of Polynomial Regression

1. **Modeling Growth Ra**tes: Polynomial regression is used to model non-linear growth rates such as the growth of tissues over time.
2. **Disease Epidemic Progression:** It helps track and predict the progression of disease outbreaks, capturing the non-linear nature of epidemic curves.
3. **Environmental Studies:** It is applied in studies like the distribution of carbon isotopes in lake sediments where relationships are non-linear.
4. **Economics and Finance:** It is used to analyze non-linear relationships in financial markets, predicting trends and fluctuations over time.