LIST OF EXPERIMENTS

Exp. No	Title of the Experiment
1.	Implementation of Linear Regression Algorithm.
2.	Implementation of Support Vector Machine.
3.	Implementation of K-Means Clustering.
4.	Implementation of K Nearest Neighbour Algorithm.
5.	Implementation of ID3 Algorithm.
6.	Implementation of Backpropagation Algorithm-ANN
7.	Implementation of A* Algorithm
8.	Implementation of Candidate Elimination Algorithm

AIML Lab Manual

1. Regression

About the Algorithm:

Regression is a statistical and machine learning method to model and analyze relationships between dependent (target) and independent (predictor) variables. It predicts continuous numerical outcomes.

Types:

1. **Linear Regression**: Predicts using a straight-line relationship.
2. **Logistic Regression**: For binary classification; uses a sigmoid function.
3. **Polynomial Regression**: Fits non-linear data using polynomial terms.
4. **Ridge Regression**: Adds an L2L2-norm penalty to regularize.
5. **Lasso Regression**: Uses L1L1-norm for feature selection.

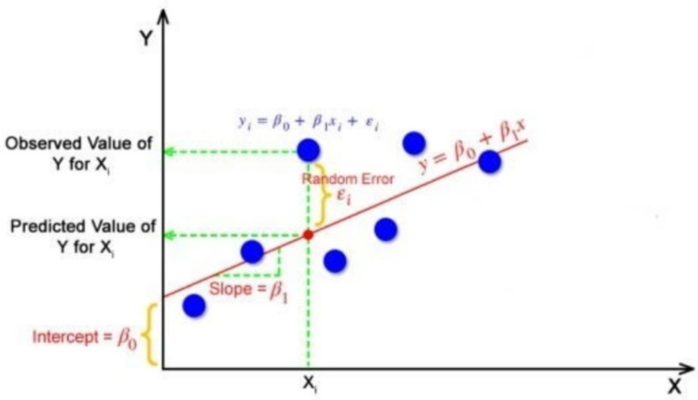


Fig: Simple linear regression

Logic or Maths:

- Linear regression predicts as:

$$y = \beta_0 + \beta_1 x + \epsilon$$

  where:

  - y: Target value
  - x: Input feature
  - β_0, β_1: Coefficients
  - ϵ: Error term

- The cost function minimizes the sum of squared errors (SSE):

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Applications:

- Predicting sales, stock prices, or energy consumption.
- Analyzing trends and relationships in business and healthcare.

Limitations:

- Assumes linearity.
- Sensitive to outliers.
- Prone to multicollinearity when predictors are correlated.

How to Overcome:

- Use regularization techniques (e.g., Ridge or Lasso).
- Normalize data and remove outliers.
- Use polynomial regression for non-linear relationships.

2.Support Vector Machine (SVM)

About the Algorithm:

SVM is a supervised learning method for classification and regression tasks. It finds the hyperplane that separates data into distinct classes with maximum margin.

Types:

1. **Linear SVM**: Used for linearly separable data.
2. **Non-linear SVM**: Applies kernel tricks (e.g., polynomial, RBF) for non-linear data.

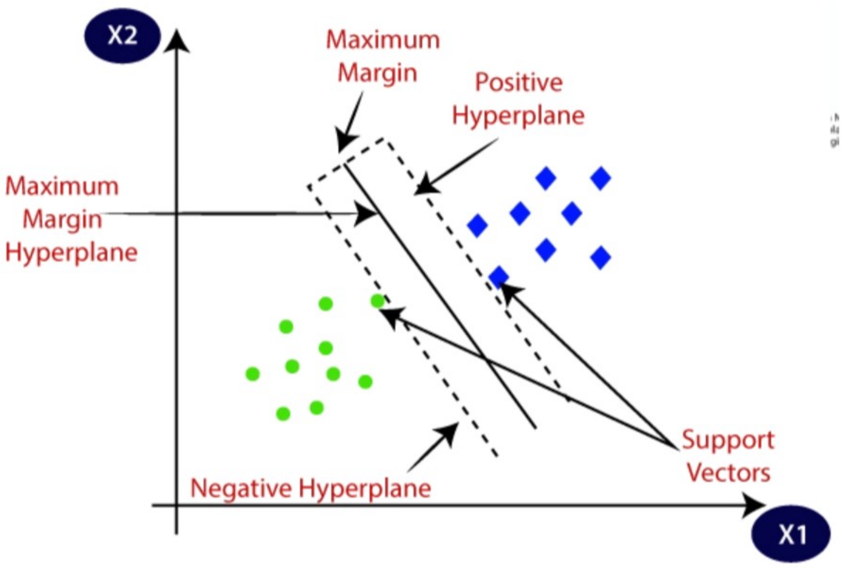


Fig: Support Vector Classifier

Logic or Maths:

- Maximizing the margin between classes:

$$\text{Maximize} \quad M = \frac{2}{||w||}$$

- The constraint:

$$y_i(w \cdot x_i + b) \geq 1$$

- Kernel trick: Maps input to a higher dimension for non-linear problems.

Applications:

- Handwriting recognition.
- Bioinformatics (protein classification).
- Email spam detection.

Limitations:

- Slow for large datasets.
- Requires careful kernel selection.

How to Overcome:

- Use approximate SVM solvers like SVM-Light.
- Perform hyperparameter tuning.
- Employ dimensionality reduction techniques.

---

3. K-Means Clustering

About the Algorithm:

An unsupervised learning method for grouping data points into K clusters by minimizing intra-cluster variance.

Types:

1. **K-means++**: Improves centroid initialization to speed up convergence.
2. **Mini-Batch K-means**: Processes data in smaller batches for efficiency.

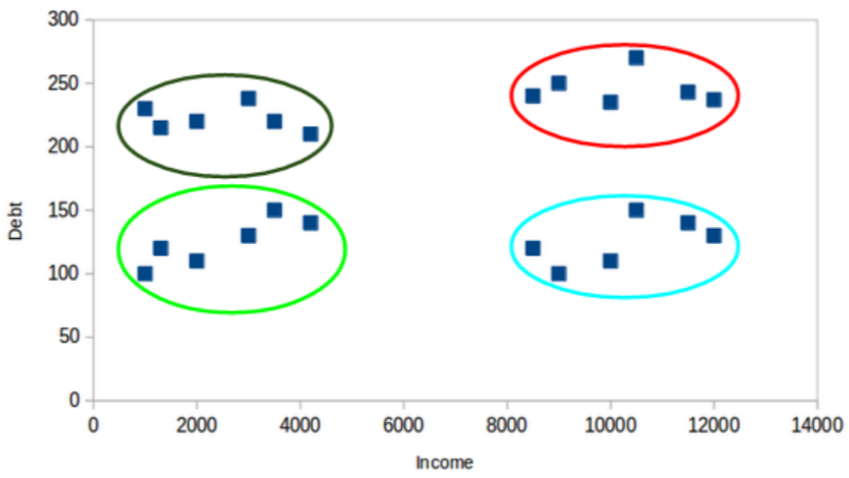


Fig: Grouping based on Income Vs Debt using KNN

Logic or Maths:

- Objective function:

$$J = \sum_{i=1}^{K} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

where μ_i is the centroid of cluster i.

Applications:

- Customer segmentation.
- Image compression.
- Topic modelling in text analysis.

Limitations:

- Sensitive to initialization.
- Assumes clusters are spherical and equally sized.

How to Overcome:

- Use K-means++ for better initialization.
- Use silhouette analysis to determine the optimal K.
- Switch to density-based clustering for complex shapes.

4. K-Nearest Neighbours (KNN)

About the Algorithm:

KNN is a simple, instance-based learning algorithm. It classifies data points based on the majority class among their K nearest neighbors.

Types:

1. **Weighted KNN**: Assigns higher weight to closer neighbors.
2. **Distance-based KNN**: Uses distances like Euclidean or Manhattan.

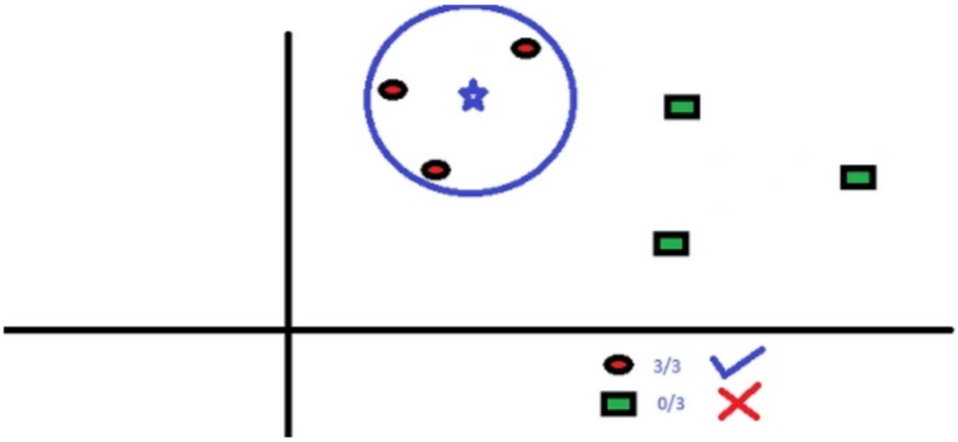


Fig: Classifying new data point blue star based on KNN

Logic or Maths:

Logic or Maths:

- Distance computation (Euclidean):

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Majority voting determines the class.

Applications:

- Recommender systems.
- Disease classification (e.g., diabetes prediction).
- Fraud detection.

Limitations:

- High computational cost for large datasets.
- Sensitive to irrelevant features.

How to Overcome:

- Use dimensionality reduction techniques like PCA.
- Leverage KD-trees or Ball trees for faster neighbor search.

---

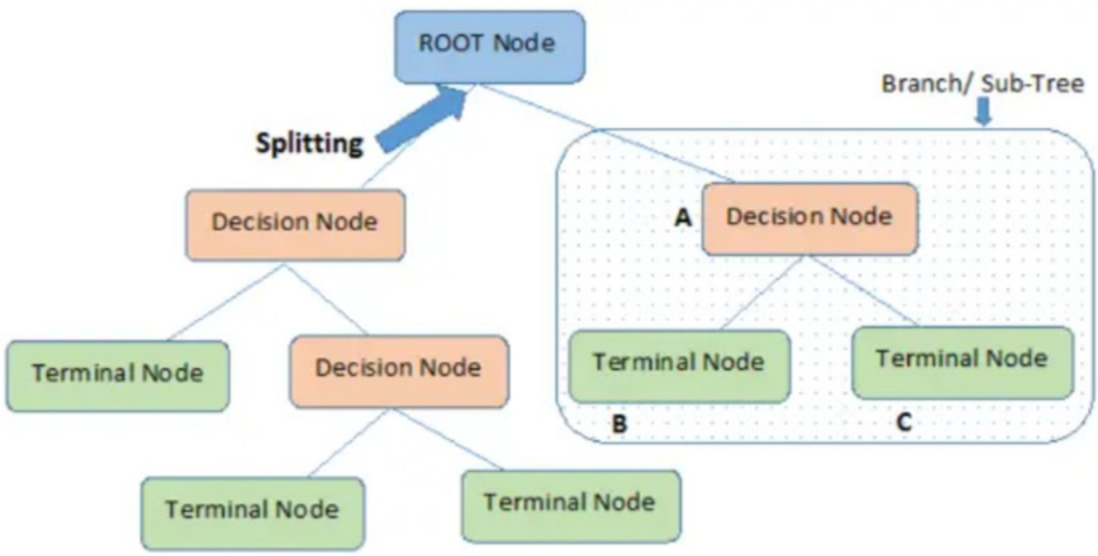
5. ID3 Decision Tree

About the Algorithm:

ID3 constructs decision trees by selecting features with the highest information gain at each node.

Types:

1. **C4.5**: An improvement of ID3, handles continuous data.
2. **CART**: Uses Gini impurity for splitting.



Logic or Maths:

- Entropy calculation:

$$H = -\sum p(x) \log_2 p(x)$$

- Information gain:

$$IG = H(\text{parent}) - \sum \left(\frac{|C_i|}{|C|} H(C_i) \right)$$

Applications:

- Customer segmentation.
- Medical diagnosis.
- Fraud detection.

Limitations:

- Prone to overfitting.
- Biased towards features with more levels.

How to Overcome:

- Use tree pruning techniques.
- Combine with ensemble methods like Random Forest.

---

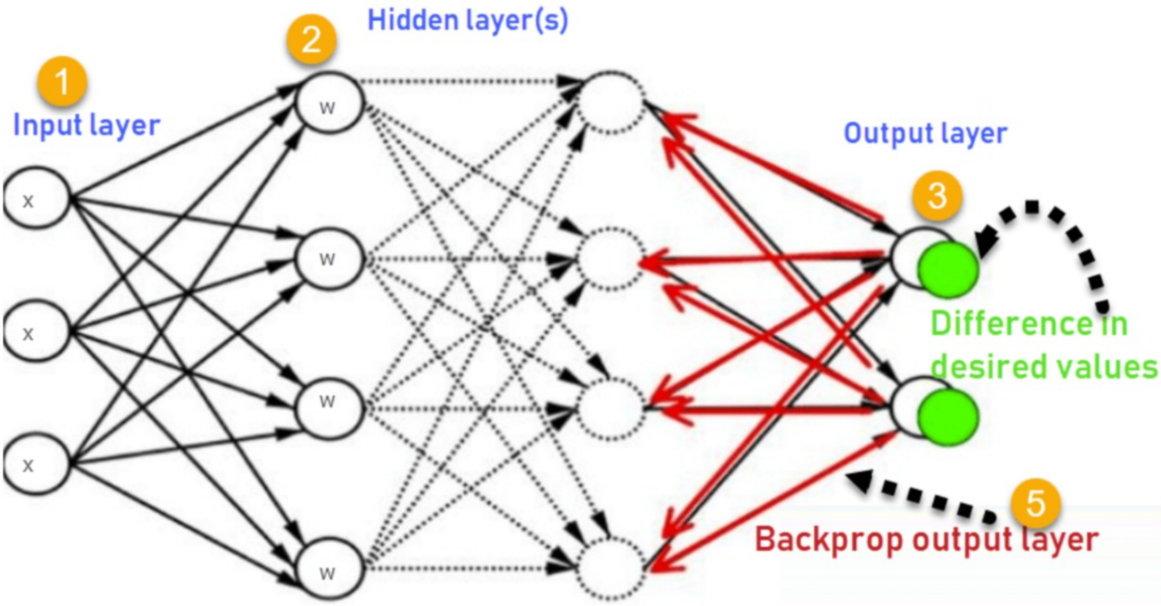
6. Backpropagation Algorithm

About the Algorithm:

Backpropagation optimizes the weights of neural networks by propagating errors backward and using gradient descent.

Types:

1. **Stochastic Gradient Descent**: Updates weights after each example.
2. **Batch Gradient Descent**: Updates weights after all examples.
3. **Mini-batch Gradient Descent**: Updates weights after a batch of examples.



How Backpropagation Algorithm Works

Logic or Maths:

- Weight update:

$$w = w - \eta \cdot \frac{\partial L}{\partial w}$$

where:

- w: Weight
- η: Learning rate
- L: Loss function

Applications:

- Speech recognition.
- Image classification.
- Natural Language Processing.

Limitations:

- Can get stuck in local minima.
- Suffers from vanishing or exploding gradients.
- Requires large datasets.

How to Overcome:

- Use advanced optimizers like Adam or RMSProp.
- Use ReLU activations to combat vanishing gradients.
- Apply batch normalization to stabilize training.

7. A* Algorithm

About the Algorithm:

A* (A-star) is a pathfinding and graph traversal algorithm widely used in computer science and artificial intelligence. It finds the shortest path between nodes in a weighted graph by considering both the actual cost from the start node and the estimated cost to the goal.

Logic or Maths Involved:

- The algorithm uses the evaluation function:
  $f(n) = g(n) + h(n)$
    - **$g(n)$:** Cost from the start node to node n.
    - **$h(n)$:** Heuristic estimate of the cost to reach the goal from node n.
- It prioritizes nodes with the lowest $f(n)$ value.

Applications:

- Game development (e.g., NPC movement in games).
- Robotics (path planning).
- Network routing.
- Mapping and navigation systems.

Limitations:

- May consume significant memory and processing power.
- Accuracy depends on the heuristic; a poorly chosen heuristic can make it inefficient.

How to Overcome:

- Use optimized heuristics like admissible heuristics (e.g., Manhattan or Euclidean distance).
- Consider alternatives like Dijkstra's algorithm for non-heuristic problems.

---

8. Candidate Elimination Algorithm

About the Algorithm:

The Candidate Elimination algorithm is a supervised learning algorithm used in concept learning. It maintains two boundaries, a general boundary (G) and a specific boundary (S), to find hypotheses consistent with the training data.

Logic or Maths Involved:

- It iteratively refines the boundaries:
  - **S:** Starts as the most specific hypothesis.
  - **G:** Starts as the most general hypothesis.
- Updates occur when a new training example is added.

Applications:

- Concept learning in machine learning.
- Knowledge representation in AI.

Limitations:

- Sensitive to noise in training data.
- Struggles with complex concepts or large hypothesis spaces.

How to Overcome:

- Use noise-tolerant extensions like k-nearest neighbors or decision trees.
- Preprocess data to reduce noise.

Blue Print to build ML models:

1. Importing the libraries
2. Loading the dataset using pandas
3. Display the number of columns and rows in the dataset
4. Knowing the statistical data for each column
5. Display how many null values each column has?
6. Methods to replace null values
7. Encoding using one hot encoder or label encoder
8. Scaling, normalization of the data
9. Importing the model
10. Defining train test split
11. Defining X and Y variables
12. Prediction for given values
13. Performance evaluation
14. Analysis of the results.