 **KNN Classification Implementation**

* **Principle**: Classify based on the majority class of the nearest neighbors.
* **Steps**:
  1. Choose K, the number of neighbors.
  2. Compute distances (e.g., Euclidean) between the test point and all training points.
  3. Identify the K nearest neighbors.
  4. Assign the class by majority vote among the neighbors.

 **Naive Bayesian Classification Implementation**

* **Principle**: Use Bayes’ theorem assuming feature independence.
* **Steps**:
  1. Calculate prior probabilities for each class.
  2. Calculate likelihoods of features given each class.
  3. Compute posterior probabilities using Bayes' theorem.
  4. Assign the class with the highest posterior probability.

 **Decision Tree Classification Implementation**

* **Principle**: Classify by splitting data based on feature values into a tree structure.
* **Steps**:
  1. Select the best feature to split using criteria like Gini impurity or Information Gain.
  2. Split the dataset based on the selected feature.
  3. Recursively apply to subsets until stopping criteria are met (e.g., max depth).
  4. Classify by traversing from root to leaf nodes.

 **Linear Regression**

* **Principle**: Predict continuous values by fitting a linear relationship between features and target.
* **Steps**:
  1. Formulate the linear model
  2. 
  3. Estimate coefficients (β) using least squares.
  4. Minimize the sum of squared residuals.
  5. Predict new values using the linear model.

 **Hierarchical Agglomerative Clustering**

* **Principle**: Build a hierarchy by iteratively merging clusters.
* **Steps**:
  1. Treat each data point as a separate cluster.
  2. Compute distances (e.g., single-linkage, complete-linkage) between clusters.
  3. Merge the closest pair of clusters.
  4. Repeat until one cluster or desired number of clusters is achieved.

 **DBSCAN**

* **Principle**: Density-based clustering to find clusters of arbitrary shape.
* **Steps**:
  1. Define ϵ\epsilonϵ (radius) and MinPts (minimum points).
  2. Classify points as core, border, or noise.
  3. Expand clusters from core points by including points within ϵ\epsilonϵ.
  4. Merge clusters if they share core points.

 **K Means Clustering**

* **Principle**: Partition data into k clusters minimizing within-cluster variance.
* **Steps**:
  1. Initialize k centroids randomly.
  2. Assign points to the nearest centroid.
  3. Recompute centroids as the mean of assigned points.
  4. Repeat until centroids stabilize.

 **Apriori Algorithm**

* **Principle**: Identify frequent itemsets and derive association rules.
* **Steps**:
  1. Generate candidate itemsets.
  2. Calculate the support for each candidate.
  3. Prune candidates below minimum support threshold.
  4. Generate association rules from frequent itemsets.

 **Image Classification using KNN, SVC, ANN**

* **KNN**:
  + **Principle**: Classify based on the majority class of the nearest image neighbors.
  + **Steps**: Convert images to feature vectors, compute distances, and classify using majority vote of K nearest neighbors.
* **SVC (Support Vector Classifier)**:
  + **Principle**: Find the optimal hyperplane separating classes in feature space.
  + **Steps**: Convert images to feature vectors, train SVC to find hyperplane, classify based on hyperplane side.
* **ANN (Artificial Neural Network)**:
  + **Principle**: Use neural networks to learn and classify image features.
  + **Steps**: Convert images to feature vectors, train the network using forward propagation and backpropagation, classify based on network output.





