**Report on Key Machine Learning Techniques**

**1. Support Vector Machine (SVM)**

**Overview**

Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification tasks, although it can also be adapted for regression. The main idea behind SVM is to find a hyperplane that best separates the data points of different classes in a high-dimensional space.

And also to maximise separation from hyperplane as widely as possible so that in future it classisfies properly all sports pf datra

Main advantages-

1)robust to outliers

2)can also handle non linear data

3)can be implemented to both classification and regression problems

**Key Concepts**

* **Hyperplane**: In an n-dimensional space, a hyperplane is a flat affine subspace of dimension n-1. In SVM, the goal is to find the optimal hyperplane that maximizes the margin between the two classes.
* **Support Vectors**: These are the data points that are closest to the hyperplane. They are critical in defining the position and orientation of the hyperplane. The SVM algorithm focuses on these points to create the decision boundary.
* **Margin**: The distance between the hyperplane and the nearest data point from either class. SVM aims to maximize this margin, which helps improve the model's generalization.

**Mathematics Involved**

Here is the correctly formatted and structured version of your text:

**Support Vector Machine (SVM)**

**Overview**

Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification tasks, though it can also be adapted for regression. The main idea behind SVM is to find a **hyperplane** that best separates data points of different classes in a high-dimensional space.

**Key Concepts**

* **Hyperplane**: A decision boundary separating different classes in feature space, represented by the equation**wx + b = 0** in linear classification.
* **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
* **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
* **Kernel**: A function that maps data to a higher-dimensional space, enabling SVM to handle non-linearly separable data.
* **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
* **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
* **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value enforces a stricter penalty for misclassifications.
* **Hinge Loss**: A loss function penalizing misclassified points or margin violations, combined with regularization in SVM.
* **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.
* The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (**support vectors**) on each side.
* 
* *Multiple hyperplanes separate the data from two classes*
* The best hyperplane, also known as the **“hard margin,”** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So, from the above figure, we choose L2 as hard margin.

**Mathematics of SVM**

**The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.**

**A soft margin allows for some misclassifications or violations of the margin to improve generalization. The SVM optimizes the following equation to balance margin maximization and penalty minimization:**

Objective Function=(1/margin)+*λ*∑penalty

The penalty used for violations is often **hinge loss**, which has the following behavior:

* If a data point is correctly classified and within the margin, there is no penalty (loss = 0).
* If a point is incorrectly classified or violates the margin, the hinge loss increases proportionally to the distance of the violation.

Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line).

When data is not linearly separable (i.e., it can’t be divided by a straight line), SVM uses a technique called **kernels** to map the data into a higher-dimensional space where it becomes separable. This transformation helps SVM find a decision boundary even for non-linear data.



*Original 1D dataset for classification*

A **kernel** is a function that maps data points into a higher-dimensional space without explicitly computing the coordinates in that space. This allows SVM to work efficiently with non-linear data by implicitly performing the mapping.

For example, consider data points that are not linearly separable. By applying a kernel function, SVM transforms the data points into a higher-dimensional space where they become linearly separable.

* **Linear Kernel**: For linear separability.
* **Polynomial Kernel**: Maps data into a polynomial space.
* **Radial Basis Function (RBF) Kernel**: Transforms data into a space based on distances between data points.

The equation for the linear hyperplane can be written as:

wTx+b=0*wTx*+*b*=0

Where:

* w*w* is the normal vector to the hyperplane (the direction perpendicular to it).
* b*b* is the offset or bias term, representing the distance of the hyperplane from the origin along the normal vector w*w*.

**Distance from a Data Point to the Hyperplane**

The distance between a data point x\_i and the decision boundary can be calculated as:

di=wTxi+b∣∣w∣∣*di*​=∣∣*w*∣∣*wTxi*​+*b*​

where ||w|| represents the Euclidean norm of the weight vector w. Euclidean norm of the normal vector W

**Linear SVM Classifier**

Distance from a Data Point to the Hyperplane:

y^={1: wTx+b≥00:  wTx+b <0*y*^​={10​: *wTx*+*b*≥0:  *wTx*+*b* <0​

Where y^*y*^​ is the predicted label of a data point.

**Optimization Problem for SVM**

For a linearly separable dataset, the goal is to find the hyperplane that maximizes the margin between the two classes while ensuring that all data points are correctly classified. This leads to the following optimization problem:

minimizew,b12∥w∥2*w*,*b*minimize​21​∥*w*∥2

Subject to the constraint:

yi(wTxi+b)≥1fori=1,2,3,⋯,m*yi*​(*wTxi*​+*b*)≥1*fori*=1,2,3,⋯,*m*

Where:

* yi*yi*​​ is the class label (+1 or -1) for each training instance.
* xi*xi*​​ is the feature vector for the i*i*-th training instance.
* m*m* is the total number of training instances.

The condition yi(wTxi+b)≥1*yi*​(*wTxi*​+*b*)≥1 ensures that each data point is correctly classified and lies outside the margin.

**Soft Margin Linear SVM Classifier**

In the presence of outliers or non-separable data, the SVM allows some misclassification by introducing slack variables ζi*ζi*​​. The optimization problem is modified as:

minimize w,b12∥w∥2+C∑i=1mζi*w*,*b*minimize ​21​∥*w*∥2+*C*∑*i*=1*m*​*ζi*​

Subject to the constraints:

yi(wTxi+b)≥1–ζiandζi≥0for i=1,2,…,m*yi*​(*wTxi*​+*b*)≥1–*ζi*​and*ζi*​≥0for *i*=1,2,…,*m*

Where:

* C*C* is a regularization parameter that controls the trade-off between margin maximization and penalty for misclassifications.
* ζi*ζi*​​ are slack variables that represent the degree of violation of the margin by each data point.

**Dual Problem for SVM**

The dual problem involves maximizing the Lagrange multipliers associated with the support vectors. This transformation allows solving the SVM optimization using kernel functions for non-linear classification.

The dual objective function is given by:

maximize α12∑i=1m∑j=1mαiαjtitjK(xi,xj)–∑i=1mαi*α*maximize ​21​∑*i*=1*m*​∑*j*=1*m*​*αi*​*αj*​*ti*​*tj*​*K*(*xi*​,*xj*​)–∑*i*=1*m*​*αi*​

Where:

* αi*αi*​​ are the Lagrange multipliers associated with the i*i*-th training sample.
* ti*ti*​​ is the class label for the iii-th training sample (+1+1+1 or −1-1−1).
* K(xi,xj)*K*(*xi*​,*xj*​) is the kernel function that computes the similarity between data points xi*xi*​​ and xj*xj*​​. The kernel allows SVM to handle non-linear classification problems by mapping data into a higher-dimensional space.

The dual formulation optimizes the Lagrange multipliers αi*αi*​​, and the support vectors are those training samples where αi>0*αi*​>0.

**SVM Decision Boundary**

Once the dual problem is solved, the decision boundary is given by:

w=∑i=1mαitiK(xi,x)+b*w*=∑*i*=1*m*​*αi*​*ti*​*K*(*xi*​,*x*)+*b*

Where w*w* is the weight vector, x*x* is the test data point, and b*b* is the bias term.

Finally, the bias term b*b* is determined by the support vectors, which satisfy:

ti(wTxi–b)=1⇒b=wTxi–ti*ti*​(*wTxi*​–*b*)=1⇒*b*=*wTxi*​–*ti*​

Where xi*xi*​​ is any support vector.

This completes the mathematical framework of the Support Vector Machine algorithm, which allows for both linear and non-linear classification using the dual problem and kernel trick.

**Types of Support Vector Machine**

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

* **Linear SVM:**Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.
* **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

**Advantages of Support Vector Machine (SVM)**

1. **High-Dimensional Performance**: SVM excels in high-dimensional spaces, making it suitable for **image classification** and **gene expression analysis**.
2. **Nonlinear Capability**: Utilizing **kernel functions** like **RBF** and **polynomial**, SVM effectively handles **nonlinear relationships**.
3. **Outlier Resilience**: The **soft margin** feature allows SVM to ignore outliers, enhancing robustness in **spam detection** and **anomaly detection**.
4. **Binary and Multiclass Support**: SVM is effective for both **binary classification** and **multiclass classification**, suitable for applications in **text classification**.
5. **Memory Efficiency**: SVM focuses on **support vectors**, making it memory efficient compared to other algorithms.

**Disadvantages of Support Vector Machine (SVM)**

1. **Slow Training**: SVM can be slow for large datasets, affecting performance in **SVM in data mining** tasks.
2. **Parameter Tuning Difficulty**: Selecting the right **kernel** and adjusting parameters like **C** requires careful tuning, impacting **SVM algorithms**.
3. **Noise Sensitivity**: SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.
4. **Limited Interpretability**: The complexity of the **hyperplane** in higher dimensions makes SVM less interpretable than other models.
5. **Feature Scaling Sensitivity**: Proper **feature scaling** is essential; otherwise, SVM models may perform poorly.

**Applications of SVM**

• **Text classification** (e.g., spam detection)

• **Face recognition**

• **Bioinformatics** (e.g., cancer detection from gene expression data)

• **Handwriting recognition**

**Conclusion**

**2. Principal Component Analysis (PCA)**

**Overview**

Principal Component Analysis (PCA) is an unsupervised dimensionality reduction technique used to reduce the number of features in a dataset while preserving as much variance as possible. It transforms the original features into a new set of orthogonal features called principal components.

**Key Concepts**

* **Variance**: PCA seeks to maximize the variance captured by the principal components. The first principal component captures the most variance, the second captures the second most, and so on.
* **Orthogonality**: The principal components are orthogonal to each other, meaning they are uncorrelated.

**Mathematics Involved**

**tep 1: Standardize the Data**

**Make sure all features (e.g., height, weight, age) are on the same scale. Why? A feature like “salary” (ranging 0–100,000) could dominate “age” (0–100) otherwise.**

[**Standardizing**](https://www.geeksforgeeks.org/normalization-vs-standardization/)**our dataset to ensures that each variable has a mean of 0 and a standard deviation of 1.**

**Z=X−μσ*Z*=*σX*−*μ*​**

**Here,**

* **μ*μ* is the mean of independent features  μ={μ1,μ2,⋯,μm}*μ*={*μ*1​,*μ*2​,⋯,*μm*​}**
* **σ*σ* is the**[**standard deviation**](https://www.geeksforgeeks.org/mathematics-mean-variance-and-standard-deviation/)**of independent features  σ={σ1,σ2,⋯,σm}*σ*={*σ*1​,*σ*2​,⋯,*σm*​}**

**Step 2: Find Relationships**

**Calculate how features move together using a *covariance matrix*.**[**Covariance**](https://www.geeksforgeeks.org/mathematics-covariance-and-correlation/)**measures the strength of joint variability between two or more variables, indicating how much they change in relation to each other. To find the covariance we can use the formula:**

**cov(x1,x2)=∑i=1n(x1i−x1ˉ)(x2i−x2ˉ)n−1*cov*(*x*1,*x*2)=*n*−1∑*i*=1*n*​(*x*1*i*​−*x*1ˉ)(*x*2*i*​−*x*2ˉ)​**

**The value of covariance can be positive, negative, or zeros.**

* **Positive: As the x1 increases x2 also increases.**
* **Negative: As the x1 increases x2 also decreases.**
* **Zeros: No direct relation.**

**Step 3: Find the “Magic Directions” (Principal Components)**

* **PCA identifies new axes (like rotating a camera) where the data spreads out the most:**
  + **1st Principal Component (PC1): The direction of maximum variance (most spread).**
  + **2nd Principal Component (PC2): The next best direction, *perpendicular to PC1*, and so on.**
* **These directions are calculated using**[**Eigenvalues and Eigenvectors**](https://www.geeksforgeeks.org/applications-of-eigenvalues-and-eigenvectors/#:~:text=Eigenvalues%20and%20eigenvectors%20are%20mathematical,scaled%20by%20its%20corresponding%20eigenvalue.)**where: eigenvectors (math tools that find these axes), and their importance is ranked by eigenvalues (how much variance each captures).**

**For a square matrix A, an eigenvector X (a non-zero vector) and its corresponding eigenvalue λ (a scalar) satisfy:**

**AX=λX*AX*=*λX***

**This means:**

* **When *A* acts on X, it only stretches or shrinks X by the scalar λ.**
* **The direction of X remains unchanged (hence, eigenvectors define “stable directions” of A).**

**It can also be written as :**

**AX−λX=0(A−λI)X=0*AX*−*λX*(*A*−*λI*)*X*​=0=0​**

**where I is the identity matrix of the same shape as matrix A. And the above conditions will be true only if (A–λI)(*A*–*λI*) will be non-invertible (i.e. singular matrix). That means,**

**∣A–λI∣=0∣*A*–*λI*∣=0**

**This determinant equation is called the characteristic equation.**

* **Solving it gives the eigenvalues \lambda,**
* **and therefore corresponding eigenvector can be found using the equation AX=λX*AX*=*λX*.**

***How This Connects to PCA?***

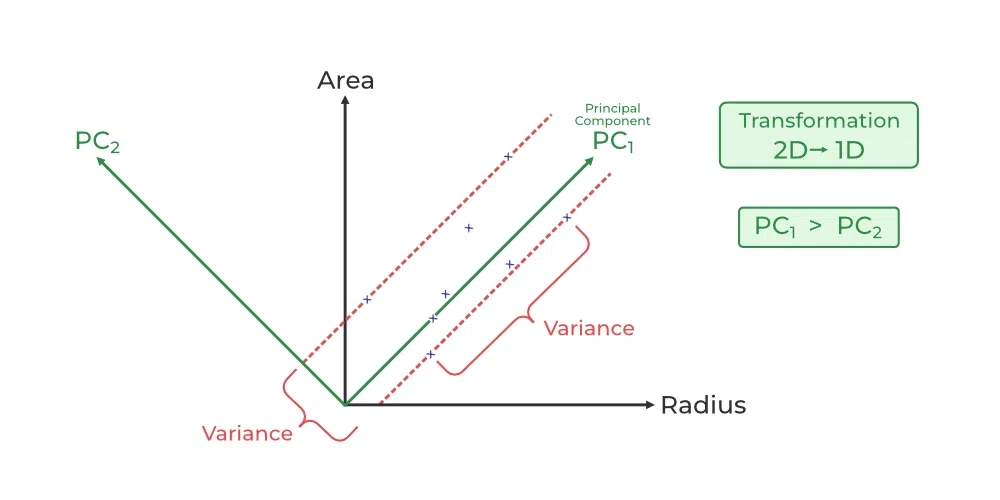
* ***In PCA, the covariance matrix C (from Step 2) acts as matrix A.***
* ***Eigenvectors of C are the principal components (PCs).***
* ***Eigenvalues represent the variance captured by each PC.***

**Step 4: Pick the Top Directions & Transform Data**

* **Keep only the top 2–3 directions (or enough to capture ~95% of the variance).**
* **Project the data onto these directions to get a simplified, lower-dimensional version.**

**PCA is an**[**unsupervised learning**](https://www.geeksforgeeks.org/supervised-unsupervised-learning/)**algorithm, meaning it doesn’t require prior knowledge of target variables. It’s commonly used in exploratory data analysis and machine learning to simplify datasets without losing critical information.**

***We know everything sound complicated, let’s understand again with help of visual image where, x-axis (Radius) and y-axis (Area) represent two original features in the dataset.***

****

***Transform this 2D dataset into a 1D representation while preserving as much variance as possible.***

**Principal Components (PCs):**

* **PC₁ (First Principal Component): The direction along which the data has the maximum variance. It captures the most important information.**
* **PC₂ (Second Principal Component): The direction orthogonal (perpendicular) to PC₁. It captures the remaining variance but is less significant.**

**Now, The red dashed lines indicate the spread (variance) of data along different directions . The variance along PC₁ is greater than PC₂, which means that PC₁ carries more useful information about the dataset.**

* **The data points (blue dots) are projected onto PC₁, effectively reducing the dataset from two dimensions (Radius, Area) to one dimension (PC₁).**
* **This transformation simplifies the dataset while retaining most of the original variability.**
* **Advantages of PCA**
* 1. **Dimensionality Reduction** – PCA reduces the number of features while retaining maximum variance, making computations faster and models more efficient.
* 2. **Removes Multicollinearity** – Since PCA transforms correlated variables into uncorrelated principal components, it helps in handling multicollinearity issues in datasets.
* 3. **Improves Model Performance** – By reducing noise and redundant features, PCA can enhance the performance of machine learning models, especially for algorithms sensitive to feature relationships.
* 4. **Visualization of High-Dimensional Data** – PCA allows high-dimensional data to be visualized in 2D or 3D, making it useful for exploratory data analysis.
* **Disadvantages of PCA**
* 1. **Loss of Interpretability** – The transformed principal components do not have meaningful interpretations like original features, making it harder to understand model behavior.
* 2. **Variance-Based Selection Might Ignore Important Features** – PCA focuses only on variance and may remove features that are important for classification or regression tasks if they contribute little to overall variance.
* 3. **Assumption of Linearity** – PCA assumes that relationships between variables are linear, which may not hold true for all datasets, leading to suboptimal results.
* 4. **Sensitive to Scaling** – PCA is affected by differences in feature scales, requiring proper standardization or normalization before applying the technique.

**Applications**

PCA is widely used in exploratory data analysis, noise reduction, and visualization of high-dimensional data.

**3. Decision Trees**

**Overview**

Decision Trees are a popular supervised learning algorithm used for both classification and regression tasks. They work by recursively splitting the data into subsets based on feature values, creating a tree-like model of decisions.

**Key Concepts**

* **Nodes**: Each internal node represents a feature, each branch represents a decision rule, and each leaf node represents an outcome.
* **Root Node** is the starting point that represents the entire dataset.
* **Branches**: These are the lines that connect nodes. It shows the flow from one decision to another.
* **Internal Nodes**are Points where decisions are made based on the input features.
* **Leaf Nodes**: These are the terminal nodes at the end of branches that represent final outcomes or predictions

**Classification of Decision Tree**

We have mainly two types of decision tree based on the nature of the target variable: **classification trees** and **regression trees**.

* **Classification trees:** They are designed to predict categorical outcomes means they classify data into different classes. They can determine whether an email is “spam” or “not spam” based on various features of the email.
* **Regression trees** : These are used when the target variable is continuous It predict numerical values rather than categories. For example a regression tree can estimate the price of a house based on its size, location, and other features.
* **Splitting Criteria**: Common criteria for splitting include:
  + **Gini Impurity**: Measures the impurity of a node. Lower values indicate better splits.
  + **Entropy**: Measures the amount of disorder or uncertainty. The goal is to minimize entropy after a split.

**Mathematics Involved**

**We have two popular attribute selection measures used:**

1. **1. Information Gain**
2. **2. Gini Index**

**1. Information Gain:**

**Information Gain tells us how useful a question (or feature) is for splitting data into groups. It measures how much the uncertainty decreases after the split. A good question will create clearer groups, and the feature with the highest Information Gain is chosen to make the decision.**

**For example, if we split a dataset of people into “Young” and “Old” based on age, and all young people bought the product while all old people did not, the Information Gain would be high because the split perfectly separates the two groups with no uncertainty left**

* **Suppose S is a set of instances, A is an attribute, Sv is the subset of S *, v* represents an individual value that the attribute *A* can take and Values (A) is the set of all possible values of A, then  
   Gain(S,A)=Entropy(S)–∑vA∣Sv∣∣S∣.Entropy(Sv)*Gain*(*S*,*A*)=*Entropy*(*S*)–∑*vA*​∣*S*∣∣*Sv*​∣​.*Entropy*(*Sv*​)**

**Entropy: is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.**

**For example, if a dataset has an equal number of “Yes” and “No” outcomes (like 3 people who bought a product and 3 who didn’t), the entropy is high because it’s uncertain which outcome to predict. But if all the outcomes are the same (all “Yes” or all “No”), the entropy is 0, meaning there is no uncertainty left in predicting the outcome**

**Suppose S is a set of instances, A is an attribute, Sv is the subset of S with A = v, and Values (A) is the set of all possible values of A, then**

**Gain(S,A)=Entropy(S)–∑vϵValues(A)∣Sv∣∣S∣.Entropy(Sv)  *Gain*(*S*,*A*)=*Entropy*(*S*)–∑*vϵValues*(*A*)​∣*S*∣∣*Sv*​∣​.*Entropy*(*Sv*​)**

1. **Gini Impurity**: [ Gini(D) = 1 - \sum\_{i=1}^{C} p\_i^2 ] where ( p\_i ) is the proportion of class ( i ) in dataset ( D ).
2. **Entropy** (continued): [ Entropy(D) = -\sum\_{i=1}^{C} p\_i \log\_2(p\_i) ] where ( p\_i ) is the proportion of class ( i ) in dataset ( D ) and ( C ) is the number of classes.
3. **Information Gain**: This is the reduction in entropy after a dataset is split on an attribute. It is calculated as: [ IG(D, A) = Entropy(D) - \sum\_{v \in Values(A)} \frac{|D\_v|}{|D|} Entropy(D\_v) ] where ( D\_v ) is the subset of ( D ) for which attribute ( A ) has value ( v ).
4. **Advantages of Decision Trees**

**Simplicity and Interpretability**: Decision trees are straightforward and easy to understand. You can visualize them like a flowchart which makes it simple to see how decisions are made.

**Versatility**: It means they can be used for different types of tasks can work well for both **classification** and **regression**

**No Need for Feature Scaling**: They don’t require you to normalize or scale your data.

**Handles Non-linear Relationships**: It is capable of capturing non-linear relationships between features and target variables.

**Disadvantages of Decision Trees**

**Overfitting**: Overfitting occurs when a decision tree captures noise and details in the training data and it perform poorly on new data.

**Instability**: instability means that the model can be unreliable slight variations in input can lead to significant differences in predictions.

**Bias towards Features with More Levels**: Decision trees can become biased towards features with many categories focusing too much on them during decision-making. This can cause the model to miss out other important features led to less accurate predictions .

1. **Applications of Decision Trees**

**Loan Approval in Banking**: A bank needs to decide whether to approve a loan application based on customer profiles.

Input features include income, credit score, employment status, and loan history.

The decision tree predicts loan approval or rejection, helping the bank make quick and reliable decisions.

**Medical Diagnosis:** A healthcare provider wants to predict whether a patient has diabetes based on clinical test results.

Features like glucose levels, BMI, and blood pressure are used to make a decision tree.

Tree classifies patients into diabetic or non-diabetic, assisting doctors in diagnosis.

**Predicting Exam Results in Education : S**chool wants to predict whether a student will pass or fail based on study habits.

Data includes attendance, time spent studying, and previous grades.

The decision tree identifies at-risk students, allowing teachers to provide additional support.

**Random Forests**

Random Forests are an ensemble learning method that builds multiple decision trees and merges them together to get a more accurate and stable prediction.

**Key Concepts**

* **Bootstrap Aggregating (Bagging)**: Random Forests use bagging to create multiple subsets of the training data by sampling with replacement.basically resampling snd column sampling is done . Each tree is trained on a different subset.
* **Feature Randomness**: When splitting nodes, Random Forests consider a random subset of features, which helps in reducing overfitting and improving generalization.

**Advantages**

* Robust to overfitting compared to individual decision trees.
* Handles large datasets with higher dimensionality well.

MAIN ADVANTAGE of random forest is to achive high bias and low variance

Because even when u include new test data or replace old test data by new ones ,error is minimal because of so many decision trees considering and hence least variance

**Bagging**

Bagging, or Bootstrap Aggregating, is a technique used to improve the stability and accuracy of machine learning algorithms. It reduces variance and helps to avoid overfitting.

**Key Concepts**

* **Bootstrap Samples**: Random samples of the dataset are created with replacement.
* **Model Training**: Each model is trained independently on its bootstrap sample.
* **Aggregation**: The final prediction is made by averaging the predictions (for regression) or by majority voting (for classification).

**Boosting**

Boosting is another ensemble technique that combines multiple weak learners to create a strong learner. Unlike bagging, boosting focuses on correcting the errors made by previous models.

**Key Concepts**

* **Sequential Learning**: Models are trained sequentially, with each new model focusing on the errors made by the previous ones.
* **Weight Adjustment**: The weights of misclassified instances are increased so that subsequent models pay more attention to them.

**Popular Boosting Algorithms**

* **AdaBoost**: Adjusts the weights of misclassified instances and combines weak classifiers.
* **Gradient Boosting**: Builds models in a stage-wise fashion and optimizes a loss function.
* **Mathematics Behind Bagging and Boosting**

**Bagging (Bootstrap Aggregating)**

* 1. **Bootstrap Sampling**: Given a dataset with samples, multiple subsets are created using random sampling with replacement. Each subset is of size .
* 2. **Model Training**: An independent model is trained on each subset.
* 3. **Aggregation**: The final prediction is obtained by averaging (for regression) or majority voting (for classification).
* • **Regression**:
* • **Classification**:
* where is the number of models, and is the indicator function.

**Boosting**

* 1. **Weighting Samples**: Each sample has an associated weight , initially set as .
* 2. **Model Training**: A weak learner is trained at iteration , focusing more on previously misclassified points.
* 3. **Error Computation**: The weighted error of the weak learner is:
* 4. **Alpha Calculation**: The weight of each weak learner is:
* 5. **Updating Weights**: The sample weights are updated as:
* ensuring misclassified points receive higher weight.
* 6. **Final Prediction**: A weighted combination of weak learners forms the final model:
* Bagging reduces variance by averaging multiple models, while Boosting reduces bias by sequentially improving weak learners.

**4. Clustering**

**K-Means Clustering**

**Overview**

K-Means is a popular unsupervised learning algorithm used for clustering. It partitions the dataset into ( k ) distinct clusters based on feature similarity.

**Key Concepts**

* **Centroids**: Each cluster is represented by a centroid, which is the mean of all points in that cluster.
* **Distance Metric**: The most common distance metric used is Euclidean distance.

**Mathematics Behind K-Means Clustering**

**K-Means is an unsupervised clustering algorithm that partitions a dataset into clusters. The goal is to minimize intra-cluster variance (sum of squared distances between points and their cluster centers).**

**1. Initialization**

**• Choose cluster centroids randomly from the data points.**

**2. Assignment Step (E-Step)**

**• Each data point is assigned to the nearest centroid based on the Euclidean distance:**

**where is the cluster assigned to , and is the centroid of cluster .**

**3. Update Step (M-Step)**

**• Update each centroid by computing the mean of all points assigned to that cluster:**

**where is the set of points in cluster , and is the number of points in that cluster.**

**4. Objective Function (Cost Function)**

**• K-Means minimizes the within-cluster sum of squares (WCSS), also called the distortion function:**

**The algorithm iterates between steps (2) and (3) until the centroids converge (i.e., no significant change in positions).**

**5. Convergence**

**• The algorithm stops when centroids do not change significantly or a predefined number of iterations is reached.**

**• It may converge to local minima, so multiple runs with different initializations (e.g., K-Means++) help improve results.**

**Complexity**

**• Assignment step: per iteration.**

**• Update step: .**

**• Overall Complexity: , where = number of data points, = dimensions, = clusters, and = iterations.**

**K-Means is simple yet powerful for clustering, but it requires choosing and is sensitive to initialization.**

Process-

1)randomly select 3 data points ,which become our initial 3 clusters

2)then we calculate Euclidean distance of every point to these initial clusters and group the point into the cluster which is nearest from it

3)we do the same for all the data points

4)calculate the mean of all the clusters ,and the mean becomes the new initial cluster and we repeat the same process

5)k means algorithm considers all the possibilities by trying to start with different initial points and performing kmeans clustering ,considering the points with least variance between the clusters

Each time we increase k value ,the total variance within each cluster is smaller than before

Ways to find k-

1)elbow method-

In elbow method ,the reduction in variation in clusters drastically till a point and then the change is smaller ,such that an elbow is formed ,the point of this change (the elbow) is considered as the k value

A diagram of a hand and a line

Description automatically generated

DB SCAN- DBSCAN is a **density-based clustering algorithm** that**groups data points that are closely packed together and marks outliers as noise** based on their density in the feature space. It  identifies clusters as dense regions in the data space, separated by areas of lower density.

Unlike K-Means or hierarchical clustering, which assume clusters are **compact and spherical**, DBSCAN excels in handling real-world data irregularities such as:

* **Arbitrary-Shaped Clusters**: Clusters can take any shape, not just circular or convex.
* **Noise and Outliers**: It effectively identifies and handles noise points without assigning them to any cluster.

**Key Parameters in DBSCAN**

* **1. eps**: This defines the radius of the neighborhood around a data point.

If the distance between two points is less than or equal to **eps**, they are considered neighbors. Choosing the right **eps** is crucial:

* If **eps** is too small, most points will be classified as noise.
* If **eps** is too large, clusters may merge, and the algorithm may fail to distinguish between them.

A common method to determine **eps** is by analyzing the **k-distance graph**.

* **2. MinPts**: This is the minimum number of points required within the **eps** radius to form a dense region.

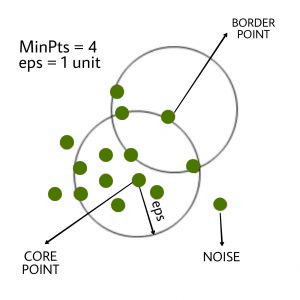
A general rule of thumb is to set MinPts >= D+1, where **D** is the number of dimensions in the dataset. For most cases, a minimum value of **MinPts = 3** is recommended.

**How Does DBSCAN Work?**

DBSCAN **works by categorizing data points into three types**:

1. core points, which have a sufficient number of neighbors within a specified radius (eplison)
2. border points, which are near core points but lack enough neighbors to be core points themselves
3. noise points, which do not belong to any cluster.

By iteratively expanding clusters from core points and connecting density-reachable points, DBSCAN forms clusters without relying on rigid assumptions about their shape or size.



**Steps in the DBSCAN Algorithm**

1. **Identify Core Points**: For each point in the dataset, count the number of points within its **eps** neighborhood. If the count meets or exceeds **MinPts**, mark the point as a **core point**.
2. **Form Clusters**: For each core point that is not already assigned to a cluster, create a new cluster. Recursively find all **density-connected points** (points within the **eps** radius of the core point) and add them to the cluster.

**Non core point which are close to a core point in a cluster is added to that cluster**

**Cluster is not further spreaded using non core points ,hence points outside final clusters are the noise points**

1. **Density Connectivity**: Two points, **a** and **b**, are **density-connected** if there exists a chain of points where each point is within the **eps** radius of the next, and at least one point in the chain is a core point. This chaining process ensures that all points in a cluster are connected through a series of dense regions.
2. **Label Noise Points**: After processing all points, any point that does not belong to a cluster is labeled as **noise**.

.