1)cross validation-

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model’s performance.

In general we consider 10 folds if not mentioned

2)Basically used to prevent overfitting ie to make the gernalisation of the data so gud that it works better with test data also

3)types-

1)holdout validation-

50% training ,50% test of the data set

Simple and easy ,only disadvantage is the

50% test data may contain important information

2)LOOCV(Leave one out cross validation)-

In this method, we perform training on the whole dataset but leaves only one data-point of the available dataset and then iterates for each data-point ie training with  n-1 samples and tested on the one omitted sample

Adv-we make use of all data points and hence it is low bias.

Disadv-

1)takes a lot of time as iterative

2)high variance as we are considering all the data ,and testing it against one data point ,if that point is an outlier then it may cause a lot of variance

3)stratified cross validation

1. The dataset is divided into k folds while maintaining the proportion of classes in each fold.
2. During each iteration, one-fold is used for testing, and the remaining folds are used for training.
3. The process is repeated k times, with each fold serving as the test set exactly once.

4)K fold validation-

we split the dataset into k number of subsets (known as folds) then we perform training on the all the subsets but leave one(k-1) subset for the evaluation of the trained model. In this method, we iterate k times with a different subset reserved for testing purpose each time

Both **K-Fold Cross-Validation** and **Stratified K-Fold Cross-Validation** are techniques used to evaluate machine learning models, but they differ in how they split the data.

• **K-Fold CV**: Randomly splits the dataset into k equal parts, ensuring each fold is used as a test set once.may lead high bias when extremely unbalanced data

• **Stratified K-Fold CV**: Ensures **each fold maintains the same class distribution** as the original dataset, making it better suited for **imbalanced classification problems**.

Ensuring low bias

Bias and variance are two types of errors in machine learning models that impact their generalization ability.

**Bias**

• Bias measures how far the predicted values are from the actual values.

• A model with **high bias** makes strong assumptions about the data, leading to underfitting.

• A model with **low bias** can capture patterns in the data effectively.

**Example:**

A linear regression model trying to fit a complex non-linear dataset would have high bias because it oversimplifies the data and fails to capture relationships.

**Variance**

• Variance measures the fluctuations of predictions for different training datasets.

• A model with **high variance** is too complex and learns even the noise in the training data, leading to overfitting.

• A model with **low variance** generalizes well to unseen data.

**Example:**

A deep decision tree may memorize the training data and fail to generalize to new data points.

**Bias-Variance Tradeoff**

| **Scenario** | **Bias** | **Variance** | **Performance** |
| --- | --- | --- | --- |
| Underfitting | High | Low | Poor (fails to capture patterns) |
| Good Fit | Low | Low | Best (generalizes well) |
| Overfitting | Low | High | Poor (memorizes training data) |

**Solutions:**

• **Reduce bias**: Use more complex models (e.g., neural networks, polynomial regression).

• **Reduce variance**: Use simpler models, regularization (L1/L2), or ensemble methods like bagging and boosting.

**Precision, Recall, Accuracy & F1 Score**

These are classification metrics used to evaluate model performance.

**Confusion Matrix**

A **confusion matrix** helps visualize model predictions:

|  | **Actual Positive** | **Actual Negative** |
| --- | --- | --- |
| **Predicted Positive** | True Positive (TP) | False Positive (FP) |
| **Predicted Negative** | False Negative (FN) | True Negative (TN) |

**Accuracy**

Measures how many total predictions were correct.

**Limitations**:

Accuracy is not a good metric for imbalanced datasets. If 95% of the data belongs to one class, a model predicting everything as that class will have high accuracy but poor performance.

**Precision (Positive Predictive Value)**

Measures how many of the predicted positives were actually correct.

• **High precision** means fewer false positives.

• Useful when false positives are costly (e.g., spam detection).

**Recall (Sensitivity or True Positive Rate)**

Measures how many actual positives were correctly identified.

• **High recall** means fewer false negatives.

• Important when missing actual positives is costly (e.g., detecting diseases).

**F1 Score**

Harmonic mean of precision and recall, balancing both.

• Useful when there is an **imbalance** between classes.

• Higher F1 score means a better balance between precision and recall.

**Specificity (True Negative Rate)**

Measures how many actual negatives were correctly identified.

• High specificity is important when **false positives** must be minimized (e.g., legal or medical applications).

**ROC Curve (Receiver Operating Characteristic)**

• Plots **True Positive Rate (Recall)** vs. **False Positive Rate**.

• The **Area Under Curve (AUC-ROC)** measures the ability to distinguish between classes.

• **Higher AUC** means better classification performance.

**PR Curve (Precision-Recall Curve)**

• Plots **Precision** vs. **Recall**.

• More useful when working with highly **imbalanced datasets**.

**Comparison of Metrics**

| **Metric** | **Measures** | **When to Use** |
| --- | --- | --- |
| Accuracy | Overall correctness | When classes are balanced |
| Precision | Correct positive predictions | When false positives are costly |
| Recall | Correctly identified positives | When false negatives are costly |
| F1 Score | Balance between precision & recall | When classes are imbalanced |
| Specificity | Correctly identified negatives | When false positives must be minimized |
| AUC-ROC | Model’s ability to distinguish classes | When ranking predictions is important |

**Summary**

• **Bias-variance tradeoff** is crucial for balancing model performance.

• **Accuracy** is not always the best metric, especially for imbalanced data.

• **Precision vs. Recall** depends on whether false positives or false negatives are more costly.

• **F1 Score** balances precision and recall for imbalanced datasets.

• **AUC-ROC** and **PR Curve** help evaluate model performance in classification tasks.

Let me know if you need further details.

Confusion matrix and correlation-

The size of the confusion matrix depends on the number of features we want to predict

Rows correspond to predicted values

Columns correspond to known truths

Sensitivity-tells us what% of the patients with heart disease were correctly identified

Sensitivity=tp/tp+fn

Specificity- tells us what% of the patients without heart disease were correctly identified

Sensitivity=tn/tn+fp

Sensitivity model predicts 0.81 ,means 81% of the heart patients were correctly identified to have heart disease by the whichever model used for prediction

We can use both these parameters to decide which model is better for predicting better output ,hogher the result better the model

The basic formula remains the same ,each time u have to identify tn fn tp fp and calculate things ,higher the dimension more complicated the calculation

Tp-predicted and actually same

tn-predicted no actual no

fp-predicted yes actual no

fn-predicted no actual yes

K MEANS-

The algorithm works by first randomly picking some central points called**centroids**and each data point is then assigned to the closest centroid forming a cluster. After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster. This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group.

\*\* ‘K’ in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

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

2) trial and error method

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.

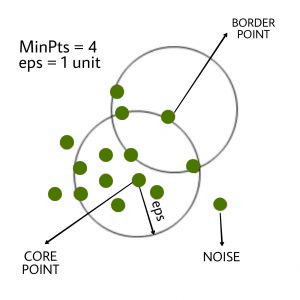
A general rule of thumb is to set MinPts >= D+1, where **D** is the number of dimensions in the dataset. For most cases, a minimum value of **MinPts = 3** is recommended.

**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**.

KNN-

K NEARES NEIGHBOURS-

In the **k-Nearest Neighbours (k-NN)** algorithm **k** is just a number that tells the algorithm how many nearby points (neighbours) to look at when it makes a decision.

**If the dataset has significant outliers or noise a higher k can help smooth out the predictions and reduce the influence of noisy data. However choosing very high value can lead to underfitting where the model becomes too simplistic.**

* **Cross-Validation**: A robust method for selecting the best k is to perform k-fold [cross-validation](https://www.geeksforgeeks.org/cross-validation-machine-learning/). This involves splitting the data into k subsets training the model on some subsets and testing it on the remaining ones and repeating this for each subset. The value of k that results in the highest average validation accuracy is usually the best choice.
* **Elbow Method**: In the [**elbow method**](https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/) we plot the model’s error rate or accuracy for different values of k. As we increase k the error usually decreases initially. However after a certain point the error rate starts to decrease more slowly. This point where the curve forms an “elbow” that point is considered as best k.
* **Odd Values for k**: It’s also recommended to choose an odd value for k especially in classification tasks to avoid ties when deciding the majority class.

**Step 1: Selecting the optimal value of K**

* K represents the number of nearest neighbors that needs to be considered while making prediction.

**Step 2: Calculating distance**

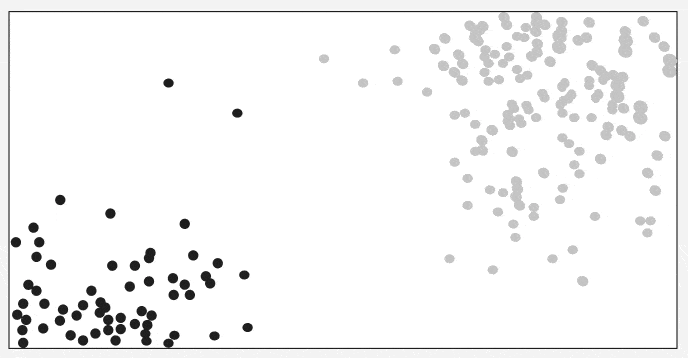
* To measure the similarity between target and training data points Euclidean distance is used. Distance is calculated between data points in the dataset and target point.

**Step 3: Finding Nearest Neighbors**

* The k data points with the smallest distances to the target point are nearest neighbors.

**Step 4: Voting for Classification or Taking Average for Regression**

* When you want to classify a data point into a category (like spam or not spam), the K-NN algorithm looks at the **K closest points** in the dataset. These closest points are called neighbors. The algorithm then looks at which category the neighbors belong to and picks the one that appears the most. This is called **majority voting**.
* In regression, the algorithm still looks for the **K closest points**. But instead of voting for a class in classification, it takes the **average** of the values of those K neighbors. This average is the predicted value for the new point for the algorithm.



*Working of KNN Algorithm*

It shows how a test point is classified based on its nearest neighbors. As the test point moves the algorithm identifies the closest ‘k’ data points i.e 5 in this case and assigns test point the majority class label that is grey label class here.

**Decision Trees**

A **Decision Tree** is a supervised learning algorithm used for classification and regression tasks. It mimics human decision-making by splitting data into branches based on feature values.

**Key Components of a Decision Tree**

1. **Root Node**: The first node that represents the entire dataset and splits into branches.

2. **Decision Nodes**: Nodes where the dataset is split based on a feature.

3. **Leaf Nodes**: The final nodes that give the output (class label or regression value).

4. **Splitting Criteria**: The condition used to split a node, based on:

• **Gini Impurity**: Measures how often a randomly chosen element would be incorrectly classified.

• **Entropy (Information Gain)**: Measures the amount of uncertainty in the dataset and aims to reduce it.

• **Variance Reduction** (for regression tasks).

**How Decision Trees Work**

1. Start from the root node and choose the best feature to split using a criterion (like Gini or Entropy).

2. Repeat the process for child nodes recursively until:

• A stopping condition is met (e.g., all samples in a node belong to the same class).

• The maximum tree depth is reached.

3. Assign class labels (classification) or return an average value (regression) at the leaf nodes.

**Advantages of Decision Trees**

Easy to understand and interpret.

Works well with both numerical and categorical data.

Requires little data preprocessing (no need for feature scaling).

**Disadvantages of Decision Trees**

Prone to **overfitting** if deep trees are created.

Unstable

Less effective on complex datasets compared to ensemble methods like **Random Forest**.

**Ways to Improve Decision Trees**

• **Pruning**: Removing unnecessary branches to reduce overfitting.

• **Setting a max depth**: Limiting the depth of the tree.

• **Minimum samples per split**: Prevents small branches from forming.

**Random Forest**

A **Random Forest** is an ensemble learning method that builds multiple decision trees and combines their results to improve accuracy and robustness.

**How Random Forest Works**

1. **Bootstrapping (Bagging)**: Multiple decision trees are trained on randomly sampled subsets of data with replacement.

2. **Feature Randomness**: At each split, only a random subset of features is considered.

3. **Voting Mechanism**:

• **Classification**: Each tree votes, and the majority class is chosen.

• **Regression**: The average of all tree outputs is taken.

**Advantages of Random Forest**

**Reduces Overfitting**: The aggregation of multiple trees reduces variance.

**Handles Missing Values**: Missing values are handled by averaging over multiple trees.

**Feature Importance**: It provides rankings of feature importance.

**Robust to Noisy Data**: Works well with imbalanced or noisy datasets.

**Disadvantages of Random Forest**

**Computationally Expensive**: Training multiple trees requires more resources.

**Harder to Interpret**: Unlike a single decision tree, random forests are more of a “black box” model.

**Hyperparameters in Random Forest**

• **n\_estimators**: Number of trees in the forest.

• **max\_depth**: Maximum depth of each tree.

• **min\_samples\_split**: Minimum samples required to split a node.

• **max\_features**: Number of features considered for splitting.

**Regularization**

Regularization is a technique used in machine learning to prevent **overfitting** by adding a penalty to complex models.

**Types of Regularization**

1. **L1 Regularization (Lasso Regression)**

• Adds the sum of absolute values of coefficients to the loss function.

• Shrinks some feature coefficients to **exactly zero**, effectively selecting important features.

• **Formula**:

• **Best for**: Feature selection.

2. **L2 Regularization (Ridge Regression)**

• Adds the sum of squared values of coefficients to the loss function.

• Shrinks coefficients towards zero but does not eliminate them.

• **Formula**:

• **Best for**: Handling multicollinearity (highly correlated features).

3. **Elastic Net (Combination of L1 and L2)**

• Uses a mix of L1 and L2 regularization.

• Helps in both feature selection and coefficient shrinking.

• **Formula**:

**Why is Regularization Needed?**

• **Overfitting occurs** when a model learns patterns specific to training data but fails on unseen data.

• **Regularization forces the model** to generalize better by penalizing complex patterns.

**Summary Table**

**When to Use Each?**

• Use **Decision Trees** when interpretability is important, and you have structured data with clear patterns.

• Use **Random Forest** when you need **high accuracy** and robustness to noise.

• Use **Regularization** when dealing with overfitting in **linear models** (like Logistic or Linear Regression).

RANSAC is an iterative algorithm used to estimate the parameters of a mathematical model from a dataset that contains outliers. It is widely used in fields such as computer vision, robotics, and data analysis, where the data is often noisy.

**Working of RANSAC**

1. **Random Sampling:**

• Select a small subset of data points randomly.

• The number of selected points should be enough to define the desired model (e.g., two points for fitting a line).

2. **Model Estimation:**

• Use the selected subset to fit a model.

3. **Consensus Measurement:**

• Evaluate how well the model fits the rest of the data.

• Count the number of points that lie within a predefined threshold distance from the model (inliers).

4. **Iteration:**

• Repeat the above steps for a fixed number of iterations.

• Keep track of the model with the highest number of inliers.

5. **Final Model Estimation:**

• Use all identified inliers to compute the final model parameters.

**Advantages**

• Robust to outliers, making it suitable for noisy data.

• Works well with datasets containing a high percentage of incorrect observations.

• Can be applied to various problems, including computer vision, 3D reconstruction, and object recognition.

**Disadvantages**

• Computationally expensive for large datasets.

• The final model depends on the number of iterations and selected threshold values.

• Does not guarantee the best possible model, as results depend on random sampling.

**Applications**

• **Computer Vision:** Image stitching, feature matching, and motion estimation.

• **Robotics:** Detecting objects and planes in 3D data from LiDAR sensors.

• **Geometric Model Estimation:** Identifying lines, circles, and planes from noisy point clouds.

**Support Vector Machine (SVM)**

**Overview**

SVM is a supervised learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that separates different classes in a dataset.

**Working of SVM**

1. **Hyperplane Selection:**

• The hyperplane is a decision boundary that maximizes the separation between different classes.

• The best hyperplane is the one that has the maximum margin from the closest data points of each class.

2. **Support Vectors:**

• The data points that are closest to the hyperplane and influence its position are called support vectors.

3. **Maximizing the Margin:**

• The margin is the distance between the hyperplane and the closest data points from either class.

• SVM aims to maximize this margin to improve generalization.

4. **Kernel Trick (For Non-Linear Data):**

• If data is not linearly separable, SVM applies the **kernel trick** to transform it into a higher-dimensional space where a hyperplane can be used for separation.

• Common kernel functions include:

• **Linear Kernel:** Used for linearly separable data.

• **Polynomial Kernel:** Captures non-linear relationships.

• **Radial Basis Function (RBF) Kernel:** Maps data to an infinite-dimensional space.

• **Sigmoid Kernel:** Similar to neural networks.

**Mathematical Formulation**

The SVM model aims to find a hyperplane defined as:

where:

• **w** is the weight vector.

• **b** is the bias.

• **x** is the input feature vector.

The optimization problem:

subject to the constraint:

where **y** represents class labels.

**Advantages**

• Effective in high-dimensional spaces and suitable for datasets with many features.

• Robust to overfitting, especially with a properly chosen kernel.

• Works well even with small datasets.

**Disadvantages**

• Slow for large datasets, especially with complex kernels.

• Sensitive to noise and overlapping classes.

• Requires careful tuning of hyperparameters like the kernel type, regularization parameter (C), and gamma.

**Applications**

• **Text Classification:** Spam filtering, sentiment analysis.

• **Image Classification:** Handwriting recognition, face detection.

• **Medical Diagnosis:** Cancer detection using MRI scans.

• **Financial Applications:** Fraud detection in banking systems.

**Conclusion**

• **RANSAC** is best for model fitting in noisy data with outliers.

• **SVM** is a powerful classification algorithm that finds the optimal decision boundary between different classes.

• **SVM works well with structured data**, while **RANSAC is useful for geometric model estimation** in unstructured and noisy datasets.