

MACHINE LEARNING NOTES

- PRANEET

INTRO

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computer systems to learn and make predictions or decisions without being explicitly programmed. It is based on the idea that machines can learn from and adapt to data, improving their performance over time.

The goal of machine learning is to develop algorithms that can automatically analyse and interpret complex patterns and relationships in data, and then use that knowledge to make predictions or take actions. This is done by training the machine learning models on large datasets, which consist of input data along with corresponding output or target values.

The process of machine learning typically involves the following steps:

1. Data collection: Gathering relevant data that is representative of the problem domain and includes the features or attributes that are important for making predictions or decisions.

2. Data preprocessing: Cleaning and transforming the data to handle missing values, outliers, and other inconsistencies. This step also involves feature engineering, where new features are derived or selected to improve the performance of the models.

3. Model selection and training: Choosing an appropriate machine learning algorithm or model architecture that is suitable for the problem at hand. The model is trained on the prepared data by adjusting its internal parameters to minimise the difference between predicted and actual outputs.

4. Model evaluation: Assessing the performance of the trained model using evaluation metrics such as accuracy, precision, recall, or mean squared error, depending on the nature of the problem. This step helps to determine if the model is able to generalise well to unseen data.

5. Model deployment and inference: Once the model has been trained and evaluated, it can be deployed to make predictions or decisions on new, unseen data. The model takes input data and produces output predictions or classifications based on what it has learned during the training phase.

There are various types of machine learning algorithms, including supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training models on labelled data, where the desired outputs are provided. Unsupervised learning deals with unlabeled data, and the models aim to discover patterns or structures in the data. Reinforcement learning involves training models to make decisions or take actions in an environment by receiving feedback or rewards.

Machine learning has applications in various fields, such as image and speech recognition, natural language processing, recommendation systems, fraud detection, autonomous vehicles, and many others. It continues to advance and play a crucial role in enabling intelligent systems to learn, adapt, and make accurate predictions or decisions based on data.

BASIC CONCEPTS

1. Features and Labels: In machine learning, the input data is typically represented as a set of features or attributes. These features capture relevant information about the problem domain. The output or target variable that we want to predict or classify is known as the label.

2. Training Data and Test Data: The available dataset is usually divided into two parts: training data and test data. The training data is used to train the machine learning model by feeding it with input features and their corresponding labels. The test data is used to evaluate the performance of the trained model by providing inputs and comparing the predicted outputs with the actual labels.

3. Supervised Learning: Supervised learning is a type of machine learning where the training data consists of labelled examples, meaning that each input is associated with a known output. The goal is to learn a mapping function that can

generalise from the training data to make accurate predictions or classifications on unseen data.

4. Unsupervised Learning: Unsupervised learning is a type of machine learning where the training data does not have labelled outputs. The objective is to discover patterns, structures, or relationships within the data. Unsupervised learning algorithms can be used for tasks such as clustering, anomaly detection, and dimensionality reduction.

5. Validation Data: In addition to training and test data, it is common to have a validation dataset. The validation data is used to tune the hyperparameters of the machine learning model during the training phase. Hyperparameters are parameters that are not learned from the data but control the behaviour of the model.

6. Model Evaluation Metrics: To measure the performance of a machine learning model, various evaluation metrics are used. The choice of metric depends on the type of problem. For classification tasks, metrics like accuracy, precision, recall, and F1 score are commonly used. For regression tasks, metrics like mean squared error (MSE) or root mean squared error (RMSE) are often employed.

7. Overfitting and Underfitting: Overfitting occurs when a model performs well on the training data but fails to generalise to new, unseen data. It usually happens when the model is too complex and learns noise or irrelevant patterns from the training data. Underfitting, on the other hand, occurs when a model is too simple to capture the underlying patterns in the data and performs poorly both on the training and test data.

8. Bias and Variance Trade-off: The bias-variance trade-off refers to the balance between the model's ability to capture the true underlying patterns in the data (low bias) and its sensitivity to the specific data points in the training set (high variance). Finding the right balance is crucial to building a good machine learning model.

9. Feature Engineering: Feature engineering is the process of selecting, transforming, or creating new features from the existing ones to improve the

performance of the machine learning models. It involves domain knowledge, creativity, and experimentation to extract the most relevant information from the data.

10. Generalisation: Generalisation refers to the ability of a machine learning model to perform well on new, unseen data that it has not been trained on. A model with good generalisation can make accurate predictions or classifications on real-world data, beyond the data it was trained on.

TYPES OF MACHINE LEARNING

- **SUPERVISED ML**

Supervised learning is a type of machine learning where the model is trained on labelled data, meaning that both the input features and their corresponding target or output values are provided during the training process. The goal of supervised learning is to learn a mapping between the input data and the desired output, enabling the model to make predictions or classifications on new, unseen data.

Here's a general overview of the supervised learning process:

- 1. Data Collection:** Gather a dataset that consists of input features and their corresponding labelled output values. The labelled data is essential for the model to learn the patterns and relationships between the features and the target variable.

- 2. Data Preparation:** Preprocess the data by handling missing values, outliers, or inconsistencies. Split the data into training and test sets to assess the model's performance on unseen data. Optionally, you can also perform additional preprocessing steps such as feature scaling or normalisation.

- 3. Model Selection:** Choose an appropriate supervised learning algorithm or model that is well-suited for the problem at hand. The choice of model depends on various factors such as the type of problem (classification or regression), the nature of the data (linear or nonlinear relationships), and the available resources.

4. Feature Engineering: If necessary, perform feature engineering to transform or create new features from the existing ones. This process can involve techniques such as feature selection, dimensionality reduction, or creating interaction terms to enhance the model's ability to learn and capture relevant information from the data.

5. Model Training: Feed the labelled training data into the chosen model and train it to learn the underlying patterns and associations between the input features and the output variable. During training, the model adjusts its internal parameters or weights based on an optimization algorithm that minimises the discrepancy between the predicted output and the true output.

6. Model Evaluation: Once the model is trained, evaluate its performance on the test set, which contains unseen data. Common evaluation metrics depend on the type of problem and can include accuracy, precision, recall, F1 score for classification, or mean squared error, mean absolute error for regression. Evaluation provides insights into the model's ability to generalise and make accurate predictions on new data.

7. Model Deployment: If the model meets the desired performance criteria, it can be deployed in a production environment to make predictions or classifications on new, unseen data. This can involve integrating the model into an application or system to provide real-time predictions.

Throughout the supervised learning process, it's important to monitor and analyse the model's performance, iterate on the feature engineering and model selection process if needed, and consider techniques to address issues such as overfitting or bias in the model.

Common supervised learning algorithms include linear regression, logistic regression, support vector machines (SVM), decision trees, random forests, gradient boosting, and artificial neural networks (including deep learning models).

Supervised learning has a wide range of applications, including sentiment analysis, spam detection, image classification, object recognition, medical diagnosis, and stock market prediction, to name a few.

- **UNSUPERVISED ML**

Unsupervised learning is a type of machine learning where the model is trained on unlabeled data, meaning that there are no predefined output values associated with the input features. The goal of unsupervised learning is to discover patterns, structures, or relationships within the data without any specific guidance or labels.

Here's a general overview of the unsupervised learning process.

- 1. Data Collection:** Gather a dataset that consists of unlabeled data, typically comprising input features without any corresponding output values. Unsupervised learning does not require labelled data, as it focuses on finding inherent structures or relationships in the data.

- 2. Data Preprocessing:** Preprocess the data by handling missing values, outliers, or inconsistencies. Perform any necessary data transformations, scaling, or normalisation to ensure the data is in a suitable format for the chosen unsupervised learning algorithm.

- 3. Model Selection:** Choose an appropriate unsupervised learning algorithm or model that is well-suited for the problem at hand. The choice of model depends on the type of problem and the specific goals of the analysis, such as clustering, dimensionality reduction, or anomaly detection.

- 4. Clustering:** Clustering is a common task in unsupervised learning, where the goal is to group similar data points together based on their inherent similarities or proximity in the feature space. There are various clustering algorithms, such as k-means, hierarchical clustering, or density-based clustering, each with its own assumptions and characteristics.

5. Dimensionality Reduction: Dimensionality reduction techniques aim to reduce the number of features or variables in the data while preserving the most important information. This is useful when dealing with high-dimensional data or when the presence of irrelevant or redundant features hinders analysis. Principal Component Analysis (PCA) and t-SNE (t-Distributed Stochastic Neighbour Embedding) are commonly used dimensionality reduction techniques.

6. Anomaly Detection: Anomaly detection is the identification of data points or patterns that deviate significantly from the norm or expected behaviour. Unsupervised learning can be used to detect anomalies in various domains, such as fraud detection, network intrusion detection, or equipment failure prediction. Outlier detection algorithms, density-based methods, or autoencoders are commonly employed for this task.

7. Pattern Discovery: Unsupervised learning can be used to discover underlying patterns or structures within the data. This can involve techniques such as association rule mining, where relationships or associations between variables are uncovered, or frequent pattern mining, where frequently occurring combinations of features are identified.

8. Evaluation and Interpretation: Evaluating unsupervised learning algorithms can be challenging since there are no predefined output values to compare against. Evaluation often involves assessing the quality and meaningfulness of the discovered patterns, clusters, or reduced feature representations. Visualisations and qualitative analysis can help interpret and understand the results.

Unsupervised learning techniques are widely applied in various domains, including customer segmentation, market basket analysis, anomaly detection, image and text clustering, recommender systems, and data exploration.

It's important to note that unsupervised learning is exploratory in nature and relies on the inherent structure of the data. Understanding the limitations, assumptions, and interpretability of the chosen unsupervised

learning methods is essential to extract meaningful insights from unlabeled data.

- **REINFORCEMENT ML**

Reinforcement learning (RL) is a type of machine learning that focuses on training agents to make a sequence of actions in an environment to maximise cumulative rewards. RL is inspired by how humans and animals learn from interaction and feedback in order to make decisions and take actions in a dynamic environment.

Here's a general overview of the reinforcement learning process

1. Agent: The agent is the entity that interacts with the environment and learns to take actions. It receives observations from the environment and selects actions to perform based on a policy.

2. Environment: The environment is the external system or world in which the agent operates. It provides the agent with state observations and receives actions as input, resulting in state transitions and rewards.

3. State: The state represents the current condition or configuration of the environment at a particular time. It provides information for the agent to make decisions and take actions.

4. Action: The action represents the choice made by the agent at a particular state. It can be a discrete action (such as moving left or right) or a continuous action (such as a specific angle or velocity).

5. Reward: The reward is the feedback or signal that the agent receives from the environment after taking an action. It quantifies the desirability or quality of the agent's actions and guides it towards maximising long-term cumulative rewards.

6. Policy: The policy defines the strategy or decision-making process of the agent. It maps states to actions and determines which action the agent should take in a given state. The policy can be deterministic or stochastic.

7. Value Function: The value function estimates the expected cumulative reward or value of being in a particular state or taking a specific action. It helps the agent evaluate the desirability of different actions or states and guides the learning process.

8. Q-Learning and Policy Gradient: Two commonly used approaches in reinforcement learning are Q-learning and policy gradient. Q-learning is a value-based method that learns the optimal action-value function, known as the Q-function. Policy gradient methods directly optimise the policy parameters by iteratively improving the policy through gradient ascent.

9. Exploration vs. Exploitation: In reinforcement learning, there is a trade-off between exploration and exploitation. Exploration involves trying out different actions to gain more information about the environment, while exploitation involves selecting actions that are estimated to yield higher rewards based on the learned policy.

10. Training and Learning: The agent interacts with the environment over multiple episodes or iterations. It learns from the received rewards and updates its policy or value function using specific algorithms, such as temporal difference learning, Monte Carlo methods, or deep reinforcement learning with neural networks.

Reinforcement learning has been successfully applied to various domains, including robotics, game playing (such as AlphaGo and OpenAI Five), autonomous vehicles, recommendation systems, and resource management.

However, reinforcement learning can be challenging due to the exploration-exploitation trade-off, the need for long-term planning, the curse of dimensionality in large state spaces, and the time required for training. Additionally, RL may require careful reward design and environment modelling to ensure effective learning and avoid unintended behaviours.

Overall, reinforcement learning is a powerful approach for training agents to learn from experience and make decisions in complex and dynamic environments, aiming to maximise cumulative rewards.

SUPERVISED ML ALGORITHMS

- **KNN**

k-Nearest Neighbors (kNN) is a simple and intuitive machine learning algorithm used for classification and regression tasks. It makes predictions based on the "nearest neighbours" of a given data point in the feature space. In other words, the algorithm classifies a data point by looking at the class labels of its k closest neighbours and determining the majority class.

How kNN Works:

1. Training Phase: The algorithm simply stores the training dataset and their corresponding class labels.

2. Prediction Phase (Classification):

- For a given test data point, the algorithm calculates the distances between the test point and all training data points using a distance metric (e.g., Euclidean distance).

- It then selects the k data points with the smallest distances.

- The majority class among the k neighbours is assigned as the predicted class for the test data point.

3. Prediction Phase (Regression):

For regression tasks, kNN calculates the average (or another aggregation function) of the target values of the k nearest neighbours and assigns it as the predicted value for the test data point.

Example: kNN for Iris Flower Classification:

Let's illustrate kNN with the famous Iris flower classification problem. Given the features (sepal length, sepal width, petal length, petal width) of different Iris flowers and their corresponding classes (setosa, versicolor, virginica), we want to predict the class of a new Iris flower based on its features.

Suppose we have a new Iris flower with the following features: (5.1, 3.5, 1.4, 0.2) and we want to use kNN with $k = 3$.

1. Calculate distances to all training data points.
2. Select the 3 nearest neighbours based on distances.
3. Among the 3 neighbours, let's say 2 are setosa and 1 is versicolor.
4. Since the majority class is setosa, the new Iris flower is classified as setosa.

Advantages of kNN:

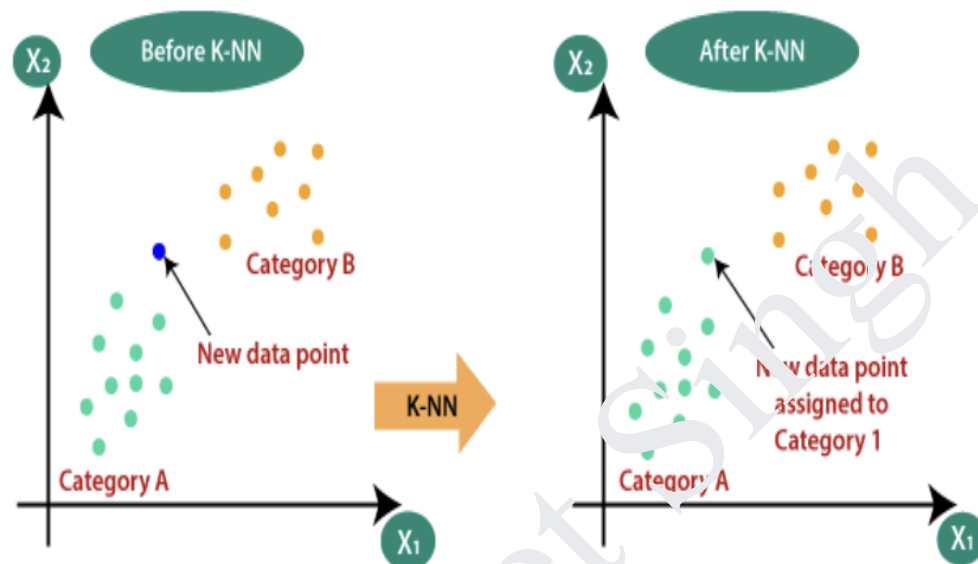
1. Simplicity: kNN is easy to understand and implement making it a good starting point for beginners.
2. No Training Phase: kNN does not require an explicit training phase; it memorises the data.
3. Adapts to Data: kNN can adapt well to changes in data distribution and handle non-linear relationships.
4. Multi-Class Classification: kNN naturally supports multi-class classification.
5. Doesn't Make Assumptions: kNN does not make strong assumptions about the data.

Drawbacks of kNN:

1. Computational Cost: As the dataset grows, the computation of distances becomes expensive, affecting efficiency.
2. Sensitive to Noise: Outliers or noisy data can significantly impact predictions.
3. Choosing k : The choice of k is critical and can affect the algorithm's performance.
4. Curse of Dimensionality: kNN's performance can degrade in high-dimensional spaces due to the "curse of dimensionality."
5. Imbalanced Data: In imbalanced datasets, kNN may favour the majority class.

In summary, k-Nearest Neighbors (kNN) is a simple and intuitive algorithm for classification and regression tasks. It relies on the concept of "closeness" to make predictions. While it offers simplicity and adaptability, it also has drawbacks related to computational cost,

sensitivity to noise, and the need to choose an appropriate value for k . It's important to consider the characteristics of the data and the specific problem when deciding whether k NN is an appropriate choice.



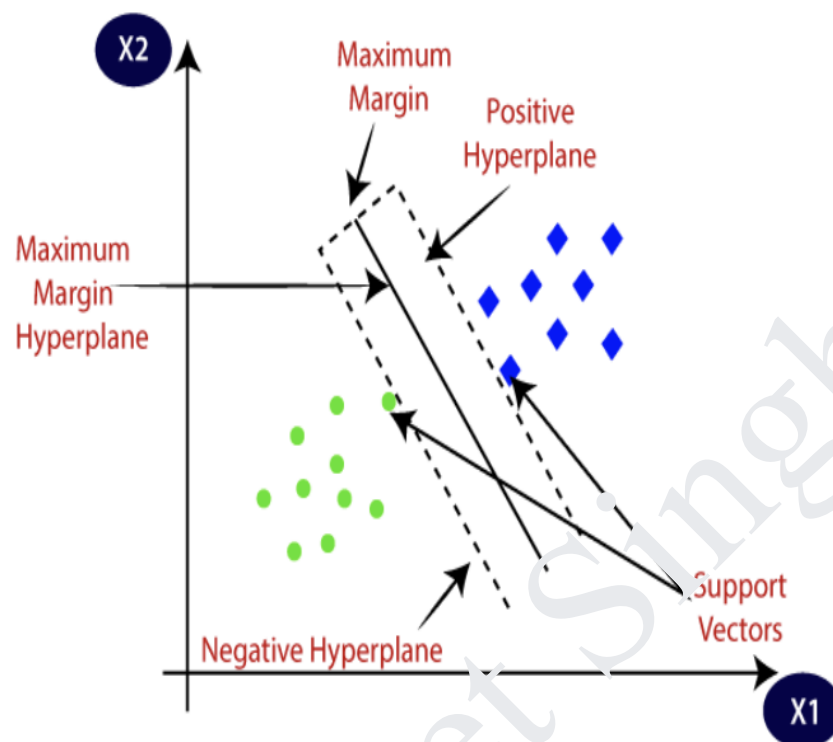
- **SVM**

Support Vector Machines (SVMs) are powerful supervised machine learning algorithms used for classification and regression tasks. SVMs aim to find a hyperplane that best separates different classes in the feature space while maximizing the margin between the classes.

Key Concepts:

1. Hyperplane: In a binary classification problem, a hyperplane is a decision boundary that separates the two classes. For an n -dimensional feature space, the hyperplane is an $(n-1)$ -dimensional plane.

2. Margin: The margin is the distance between the hyperplane and the nearest data points from each class. SVM aims to find the hyperplane with the maximum margin, as this helps improve the generalisation performance of the model.



3. Linearly Separable: Data points are linearly separable if it is possible to draw a hyperplane that perfectly separates the classes without any misclassification.

4. Non-linearly Separable: In cases where data points are not linearly separable, SVMs can use kernel functions to map the data into a higher-dimensional space where they might become separable.

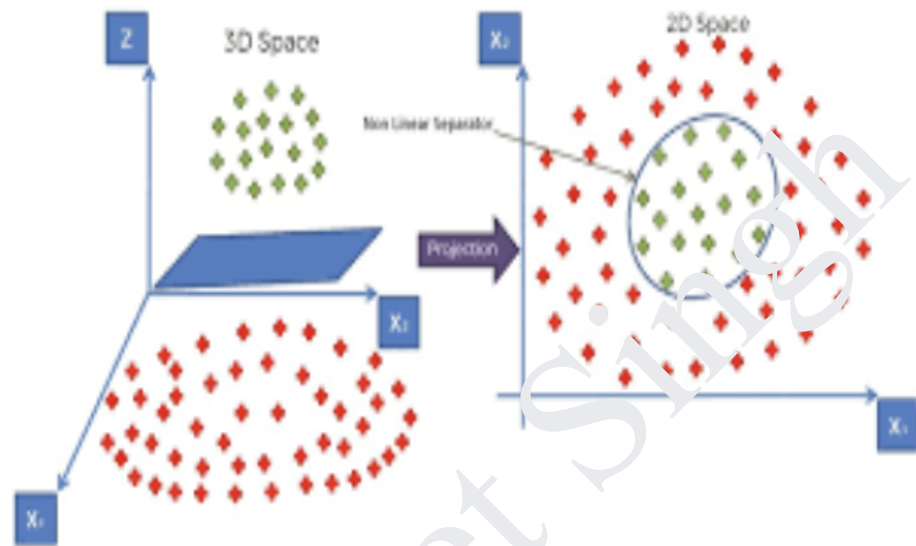
SVMs in Linearly Separable Case:

SVMs aim to find the hyperplane with the maximum margin between the classes. The margin is the distance between the hyperplane and the nearest data points from each class.

SVMs in Non-Linearly Separable Case:

When classes are not linearly separable, SVMs use kernel functions to transform the data into a higher-dimensional space where they might

become linearly separable. Common kernel functions include the polynomial kernel and radial basis function (RBF) kernel.



Margins and Support Vectors

Support vectors are the data points that lie closest to the hyperplane. The margin is computed as the distance between the support vectors of different classes. SVM aims to maximise this margin. Support vectors play a crucial role in determining the position and orientation of the hyperplane.

Advantages of SVM:

1. Effective in High-Dimensional Spaces: SVMs perform well in high-dimensional spaces, making them suitable for problems with many features.
2. Robust to Overfitting: SVMs aim to maximise the margin, leading to better generalisation and improved performance on unseen data.
3. Flexibility with Kernels: SVMs can model non-linear relationships using various kernel functions.
4. Few Parameters: SVMs have few hyperparameters to tune, making them less prone to overfitting due to excessive parameter tuning.

Disadvantages of SVM:

1. Computational Complexity: Training an SVM can be computationally expensive, especially for large datasets.
2. Sensitivity to Scaling: SVMs are sensitive to feature scaling, so proper preprocessing is important.
3. Kernel Choice: The choice of kernel and its parameters can impact the model's performance.
4. Interpretability: SVMs can be less interpretable than simpler models like decision trees.

Examples of SVM:

1. Text Classification: SVMs can be used for sentiment analysis, spam detection, and text categorization.
2. Image Classification: SVMs have been used for classifying objects in images, such as identifying handwritten digits.
3. Bioinformatics: SVMs can classify proteins, genes, and patients in medical studies.

In summary, Support Vector Machines (SVMs) are versatile and powerful machine learning algorithms used for classification and regression tasks. They aim to find the hyperplane that maximises the margin between classes, and they can handle both linearly and nonlinearly separable data through the use of kernel functions. SVMs offer advantages like effectiveness in high-dimensional spaces and robustness to overfitting, but they can be computationally expensive and require careful parameter tuning.

Some of the areas where SVM is used are Image classification, Face Detection and Text categorisation

● DECISION TREE

A Decision Tree is a popular supervised machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the feature space into subsets based on the values of input

features, ultimately leading to a tree-like structure of decisions that can be used for making predictions.

Key Concepts:

1. Node: A decision tree consists of nodes that represent features or attributes. The top node is called the root node.

2. Branch: A branch represents a decision or a test on an attribute.

3. Leaf: A leaf node represents a class label (in classification) or a predicted value (in regression).

4. Depth: The depth of a tree is the length of the longest path from the root to a leaf.

Building a Decision Tree:

1. **Selecting the Best Attribute:** The algorithm selects the attribute that best splits the data into purest subsets, using measures like Information Gain or Gini Impurity.

2. **Splitting:** The data is split into subsets based on the selected attribute's values.

3. **Repeating:** The process is repeated recursively for each subset, creating a tree-like structure.

4. **Stopping Criteria:** The algorithm stops when a certain stopping criterion is met, such as reaching a maximum depth, achieving a minimum number of samples in a leaf, or when further splits do not significantly improve the purity or prediction accuracy.

Information Gain and Gini Impurity:

Information Gain: It measures how much a feature reduces the uncertainty (entropy) in the target variable. The attribute with the highest Information Gain is selected for splitting. It's calculated as the difference between the entropy of the parent node and the weighted average of the entropies of the child nodes.

Gini Impurity: It measures the probability of a randomly selected element being misclassified. The attribute with the lowest Gini Impurity is chosen.

for splitting. It's calculated as $(1 - \sum (p_i)^2)$, where (p_i) is the probability of class (i) in the node.

Advantages of Decision Trees:

1. Interpretability: Decision trees are easy to interpret and visualise, making them useful for explaining decisions to non-technical stakeholders.
2. Handling Non-Linearity: Decision trees can capture nonlinear relationships in data.
3. Feature Importance: Decision trees can provide insights into feature importance for prediction.
4. Handles Missing Data: Decision trees can handle missing values and make predictions even when some attributes are missing.

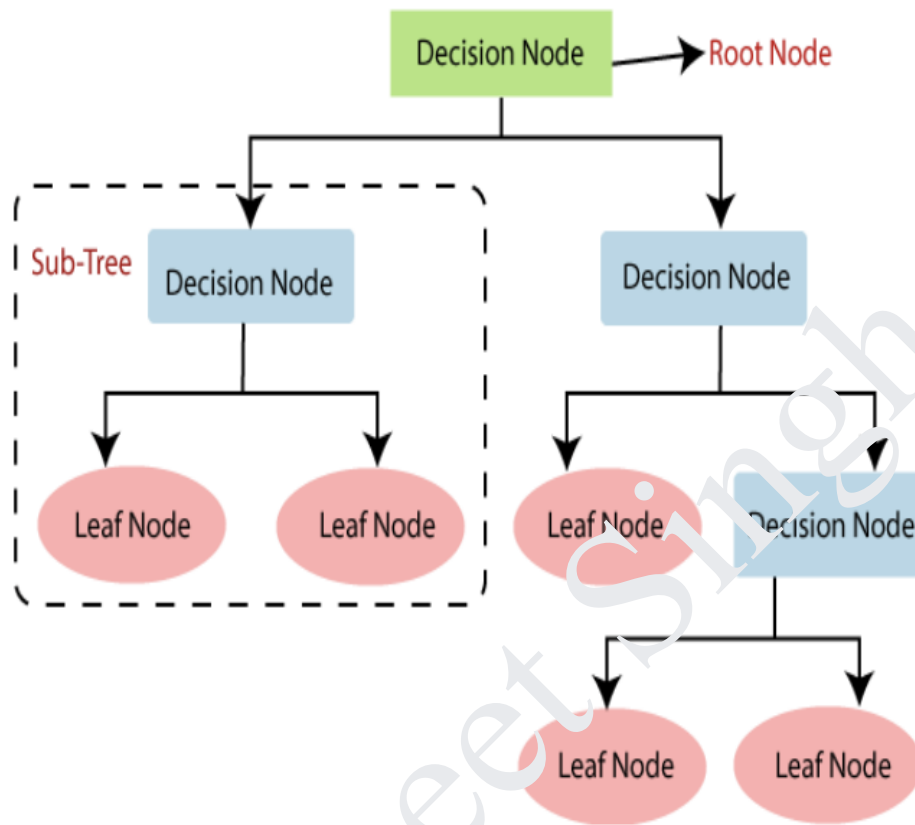
Drawbacks of Decision Trees:

1. Overfitting: Decision trees can overfit if they become too deep or complex.
2. Instability: Small changes in data can lead to different tree structures.
3. Bias towards Dominant Classes: In classification tasks, decision trees can be biased towards dominant classes.

Examples of Decision Tree:

1. Credit Scoring: Deciding whether to approve a credit card application based on income, credit history, etc.
2. Medical Diagnosis: Predicting whether a patient has a certain disease based on symptoms and test results.
3. Customer Churn: Predicting whether a customer will churn based on various features like usage behaviour and demographics.

In summary, a Decision Tree is a versatile algorithm used for classification and regression tasks. It creates a tree-like structure based on the Information Gain or Gini Impurity to make decisions and predictions. Decision trees are interpretable and handle non-linear relationships well, but they can be prone to overfitting and instability.



- **RANDOM FOREST**

Random Forest is a powerful ensemble learning algorithm used for classification, regression, and other tasks. It combines multiple decision trees to create a more robust and accurate model by reducing overfitting and improving generalisation.

Key Concepts:

1. Ensemble Learning: Random Forest is an ensemble method that builds multiple decision trees and combines their predictions to make a final decision.

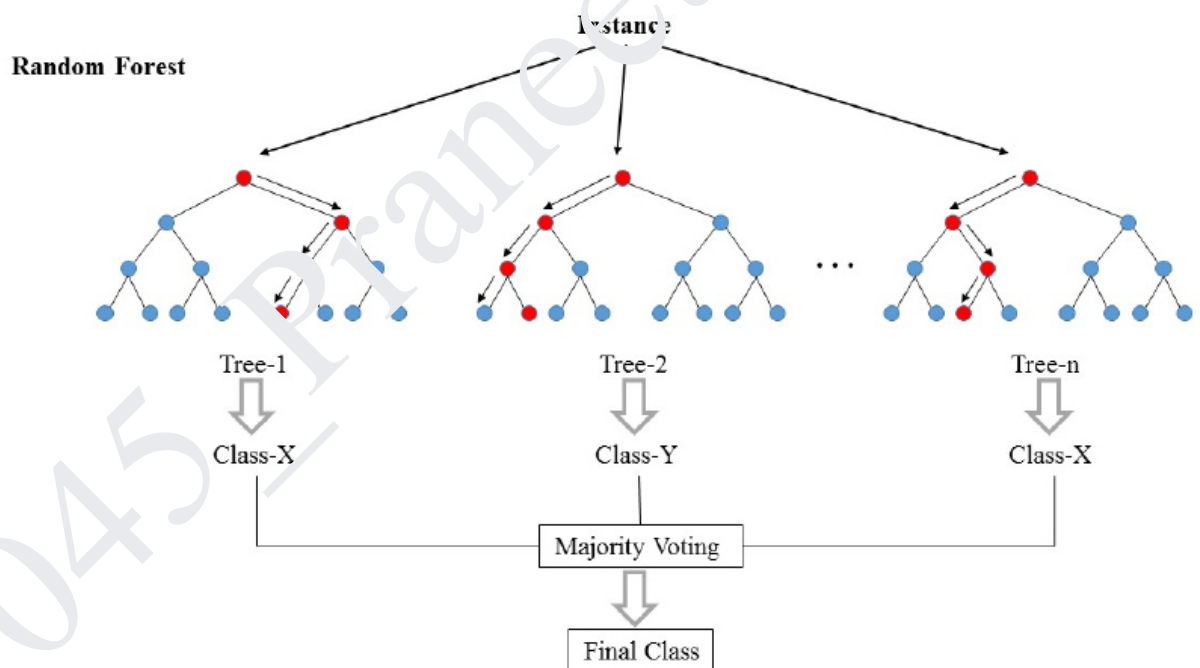
2. Bootstrap Aggregating (Bagging): Each decision tree in the Random Forest is trained on a bootstrapped subset of the original data, created by random sampling with replacement.

3. Feature Randomness: At each split of a decision tree, a random subset of features is considered, which helps to decorrelate trees and reduce overfitting.

4. Voting and Averaging: For classification, the final prediction is determined by majority voting among the decision trees. For regression, the final prediction is the average of the predictions from all trees.

Building a Random Forest:

1. **Bootstrapping:** Randomly sample the training dataset with replacement to create multiple subsets (bootstrap samples).
2. **Building Decision Trees:** For each bootstrap sample, build a decision tree using a subset of features at each split.
3. **Aggregating Predictions:** Combine predictions from all decision trees to make the final prediction (classification) or average (regression).



Advantages of Random Forest:

1. **Reduced Overfitting:** Random Forest reduces overfitting compared to a single decision tree due to the combination of multiple trees.

2. **Robustness:** It performs well on noisy and missing data due to the ensemble approach.
3. **Feature Importance:** Random Forest can provide insights into feature importance for better understanding of the data.
4. **Versatility:** It works well for both classification and regression tasks.
5. **Non-Linearity Handling:** Random Forest can capture nonlinear relationships in data.

Drawbacks of Random Forest:

1. **Complexity:** Random Forest can be computationally expensive and memory-intensive, especially for a large number of trees.
2. **Lack of Interpretability:** While it provides feature importance, the final model may be less interpretable than a single decision tree.

Examples of Random Forest:

1. **Image Classification:** Random Forest can be used to classify objects in images, such as classifying handwritten digits.
2. **Medical Diagnostics:** Predicting disease diagnosis based on patient data, including symptoms, test results, and medical history.
3. **Stock Price Prediction:** Predicting stock prices based on historical market data and various indicators.

In summary, Random Forest is a versatile and powerful ensemble learning algorithm that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It is suitable for various tasks and can handle noisy and missing data. While it may be computationally expensive, its robustness and ability to handle complex relationships make it a popular choice in machine learning.

REGRESSION

Regression is a fundamental concept in statistics and machine learning that involves modelling the relationship between a dependent variable (also known

as the target or response variable) and one or more independent variables (also known as predictors or features). The goal of regression is to understand how changes in the independent variables are associated with changes in the dependent variable, allowing us to make predictions or infer insights about the data.

Key Concepts:

1. **Dependent Variable (Y):** The variable we are trying to predict or explain. It is the outcome or response of interest.
2. **Independent Variables (X):** The variables used to predict or explain the dependent variable. These are also known as predictor variables or features.
3. **Regression Line/Function:** The mathematical equation that represents the relationship between the dependent and independent variables. In simple linear regression, this is a straight line; in multiple linear regression, it's a hyperplane in higher dimensions.

Types of Regression:

1. **Simple Linear Regression:** Involves a single independent variable to predict the dependent variable. The relationship is modelled as a straight line.
2. **Multiple Linear Regression:** Involves multiple independent variables to predict the dependent variable. The relationship is modelled as a hyperplane.
3. **Polynomial Regression:** Allows for non-linear relationships by fitting a polynomial function to the data.
4. **Ridge Regression and Lasso Regression:** Regularized regression techniques that prevent overfitting by adding penalty terms to the regression equation.
5. **Logistic Regression:** Despite its name, logistic regression is used for binary classification, not regression. It models the probability of an event occurring based on the independent variables.

Advantages of Regression:

1. **Interpretability:** Regression models provide insights into the relationships between variables, helping to understand the driving factors.
2. **Predictive Power:** Regression can make accurate predictions when the relationship between variables is well-defined.

3. **Versatility:** Regression is widely applicable in various fields, including economics, social sciences, finance, and more.

Drawbacks of Regression:

1. **Assumptions:** Regression assumes a linear relationship between variables, which might not always hold.
2. **Overfitting:** Complex regression models can overfit the training data, resulting in poor generalisation to new data.
3. **Outliers:** Regression is sensitive to outliers, which can disproportionately influence the model.

Examples of Regression:

1. **Real Estate:** Predicting house prices based on features like area, location, number of bedrooms, etc.
2. **Economics:** Modelling the relationship between GDP and factors like inflation, unemployment, and interest rates.
3. **Healthcare:** Predicting a patient's blood sugar levels based on factors like age, weight, and diet.

In summary, regression is a crucial statistical and machine learning technique used to model and understand the relationships between variables. It allows us to make predictions and draw insights from data, making it an essential tool in various fields.

● **LINEAR REGRESSION**

Linear Regression is a fundamental statistical technique used to model the relationship between a dependent variable (also known as the response or target variable) and one or more independent variables (also known as predictors or features). It assumes a linear relationship between the variables and aims to find the best-fitting line that represents this relationship.

Key Concepts:

1. **Dependent Variable (Y):** The variable we want to predict or explain. It is the outcome of interest.

2. **Independent Variables (X):** The variables used to predict the dependent variable. These are the features that influence the outcome.
3. **Regression Line:** The line that best fits the data points, representing the linear relationship between the independent and dependent variables.
4. **Regression Coefficients:** The coefficients (slopes) of the regression equation that determine the impact of each independent variable on the dependent variable.

Equation of Simple Linear Regression:

For simple linear regression (one dependent variable and one independent variable), the equation of the regression line is:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

Where:

- Y is the dependent variable (response),
- X is the independent variable (feature),
- β_0 is the intercept (the value of Y when $(X = 0)$),
- β_1 is the regression coefficient (slope), indicating the change in Y for a unit change in X ,
- ε is the error term, accounting for the variability not explained by the model.

Fitting the Regression Line:

The goal is to find the values of β_0 and β_1 that minimise the sum of squared differences between the actual Y values and the predicted Y values (obtained from the regression line). This is typically done using optimization techniques.

Coefficient Interpretation:

- β_0 : It represents the intercept of the regression line, indicating the value of Y when X is zero (which might not have a practical interpretation depending on the context).
- β_1 : It represents the change in the dependent variable Y for a unit change in the independent variable X .

Variance and Bias:

- **Bias:** It refers to the error introduced by approximating a real-world problem, which may be extremely complicated, by a simplified model. Linear regression can have bias if the true relationship between variables is non-linear.
- **Variance:** It refers to the variability of model predictions for different training datasets. High variance can result from an overly complex model that fits the noise in the data.

Assumptions of Linear Regression:

1. **Linearity:** The relationship between the variables is linear.
2. **Independence:** The errors are independent of each other.
3. **Homoscedasticity:** The variability of the errors is constant across all levels of the independent variable.
4. **Normality:** The errors are normally distributed.

Advantages of Linear Regression:

1. **Interpretability:** Coefficients can provide insights into the relationships between variables.
2. **Simplicity:** Linear regression is easy to implement and understand.
3. **Versatility:** It can be used for prediction, explanation, and hypothesis testing.

Drawbacks of Linear Regression:

1. **Limited Flexibility:** Linear regression assumes a linear relationship between variables.
2. **Sensitive to Outliers:** Outliers can disproportionately affect the regression line.
3. **Non-Constant Variance:** Violation of assumptions can lead to unreliable results.

In summary, linear regression is a widely used technique for modelling and predicting relationships between variables. It provides a simple yet powerful way to understand the impact of independent variables on a dependent variable.

• LOGISTIC REGRESSION

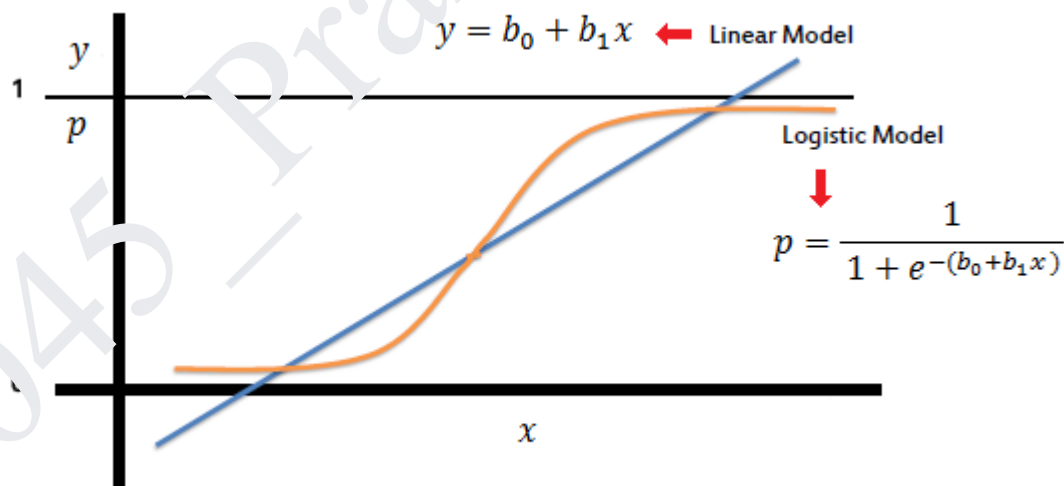
Logistic Regression is a statistical method used for binary classification problems, where the goal is to predict whether an instance belongs to one of two classes. Despite its name, it is used for classification, not regression. It models the probability that an instance belongs to a particular class.

Key Concepts:

Dependent Variable (Y): In logistic regression, the dependent variable represents the binary outcome (0 or 1).

Independent Variables (X): These are the input features used to predict the binary outcome.

Logistic Function (Sigmoid): Logistic regression uses the logistic (sigmoid) function to transform the linear combination of inputs into a probability value between 0 and 1.



Decision Boundary:

In binary classification, a threshold is chosen (usually 0.5), and if the predicted probability is above the threshold, the instance is classified as 1; otherwise, it's classified as 0.

Advantages of Logistic Regression:

Interpretability: Coefficients can be interpreted as the change in the log-odds for a unit change in the independent variable.

Probability Output: Logistic regression provides probabilities that can be useful for decision-making.

Simplicity: Logistic regression is relatively simple and easy to implement.

Drawbacks of Logistic Regression:

Linearity Assumption: Logistic regression assumes a linear relationship between inputs and log-odds, which may not hold in some cases.

Limited to Binary Classification: Logistic regression is designed for binary classification and may need extensions for multi-class problems.

Sensitive to Outliers: Outliers can influence the coefficients and predictions.

Examples of Logistic Regression:

Email Spam Detection: Classifying emails as spam or not spam based on features like subject, sender, and content.

Medical Diagnosis: Predicting whether a patient has a certain disease based on symptoms and test results.

Credit Risk Assessment: Determining the creditworthiness of a loan applicant based on various financial factors.

In summary, logistic regression is a widely used method for binary classification problems. It models the probability of an instance belonging to a specific class using the logistic function. Logistic regression provides interpretable results and is a valuable tool for a variety of applications.

DIFFERENCE BETWEEN LINEAR AND LOGISTIC

LINEAR REGRESSION	LOGISTIC REGRESSION
A linear approach that models the relationship between a dependent variable and one or more independent variables	A statistical model that predicts the probability of an outcome that can only have two values
Used to solve regression problems	Used to solve classification problems (binary classification)
Estimates the dependent variable when there is a change in the independent variable	Calculates the possibility of an event occurring
Output value is continuous	Output value is discrete
Uses a straight line	Uses an S curve or sigmoid function
Ex: predicting the GDP of a country, predicting product price, predicting the house selling price, score prediction	Ex: predicting whether an email is spam or not, predicting whether the credit card transaction is fraud or not, predicting whether a customer will take a loan or not

● **MULTIPLE LINEAR REGRESSION**

Multiple Linear Regression is an extension of simple linear regression that allows us to model the relationship between a dependent variable and multiple independent variables. It is a versatile statistical method used for predicting a continuous outcome based on the values of two or more predictor variables.

Key Concepts:

Dependent Variable (Y): The variable we want to predict or explain. It is the outcome of interest.

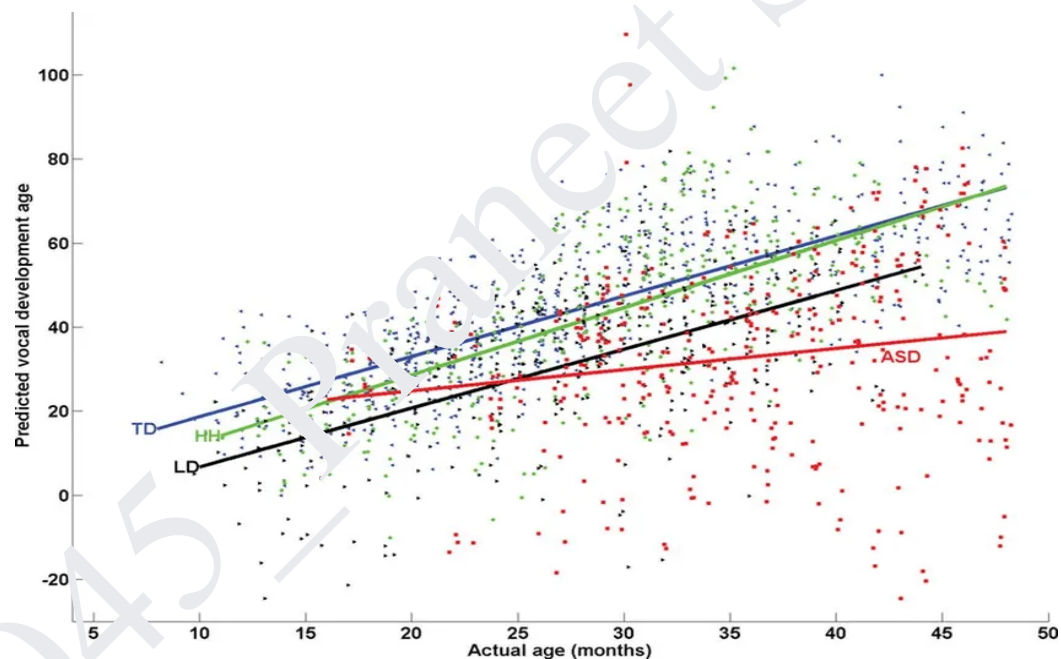
Independent Variables (X1, X2, ... Xn): These are the predictor variables used to predict the dependent variable.

Regression Equation: In multiple linear regression, the relationship between the dependent variable and independent variables is represented by a linear equation.

Equation of Multiple Linear Regression:

The equation for multiple linear regression is:

$$Y = \beta_0 + \beta_1.X_1 + \beta_2.X_2 + \dots + \beta_n.X_n + \epsilon$$



Fitting Multiple Linear Regression:

The goal is to find the values of $\beta_0, \beta_1, \beta_2, \dots, \beta_n$ that minimise the sum of squared distances between the actual Y and the predicted Y values.

Assumptions of Multiple Linear Regression:

Linearity: The relationship between the variables is linear.

Independence: The errors are independent of each other.

Homoscedasticity: The variability of the errors is constant across all levels of the independent variables.

Normality: The errors are normally distributed.

Advantages of Multiple Linear Regression:

Multiple Variables: It can model the relationships between multiple independent variables and the dependent variable simultaneously.

Interpretability: Coefficients can provide insights into the relationships between variables.

Prediction: It can be used for prediction when the assumptions are met.

Drawbacks of Multiple Linear Regression:

Assumption Violations: Violations of the assumptions can lead to unreliable results.

Overfitting: Including too many variables can lead to overfitting and poor generalisation to new data.

Examples of Multiple Linear Regression:

Housing Prices: Predicting house prices based on features like area, number of bedrooms, location, and other amenities.

Economic Forecasting: Modelling the relationship between GDP and factors like inflation, unemployment, and interest

CONFUSION MATRIX

A **confusion matrix** is a fundamental evaluation technique in machine learning used to assess the performance of a classification model. It provides a clear and detailed summary of the model's predictions and their correctness.

Components of a Confusion Matrix:

In a confusion matrix, the predictions of a model are compared to the actual outcomes, resulting in four key values:

	Actual Positive	Actual Negative
Predicted Positive	True Positives	False Positives
Predicted Negative	False Negatives	True Negatives

Performance Metrics from Confusion Matrix:

Accuracy: Measures the overall correctness of the model's predictions.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Precision: Measures the accuracy of positive predictions made by the model.

$$\text{Precision} = TP / (TP + FP)$$

Recall (Sensitivity or True Positive Rate): Measures the model's ability to identify all positive instances.

$$\text{Recall} = TP / (TP + FN)$$

Specificity (True Negative Rate): Measures the model's ability to identify all negative instances.

$$\text{Specificity} = \text{TN} / (\text{TN} + \text{FP})$$

F1-Score: Combines precision and recall, providing a balance between them.

$$\text{F1-Score} = (2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

ROC Curve (Receiver Operating Characteristic Curve): A graphical representation showing the trade-off between true positive rate (sensitivity) and false positive rate as the classification threshold varies.

AUC-ROC (Area Under the ROC Curve): Quantifies the overall performance of a classification model. A value closer to 1 indicates better performance.

Interpretation:

- High accuracy doesn't necessarily indicate a good model if the classes are imbalanced.
- High precision is important when minimising false positives is crucial (e.g., medical diagnoses).
- High recall is important when minimising false negatives is crucial (e.g., detecting fraud).

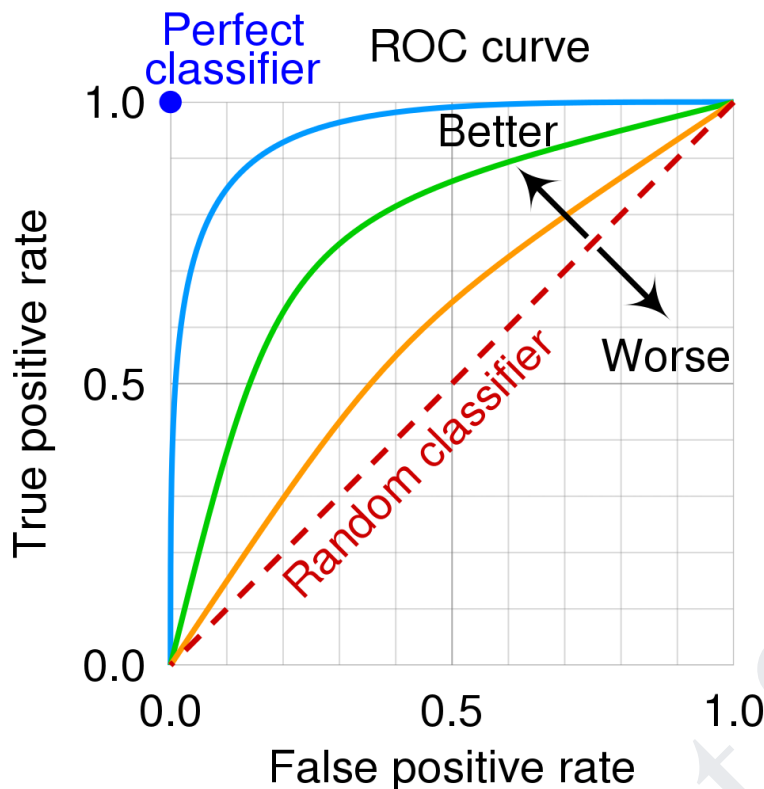
Use Cases:

Confusion matrices and associated metrics are essential for evaluating and comparing the performance of different classification models. They help in understanding the strengths and weaknesses of a model's predictions, making informed decisions about model adjustments or selection.

In summary, the confusion matrix is a valuable tool to evaluate a classification model's performance, providing insights into its ability to make correct and incorrect predictions for each class.

ROC CURVE

The Receiver Operating Characteristic (ROC) curve is a graphical representation used in machine learning to assess the performance of a binary classification model at various classification thresholds. It illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate as the threshold for classifying positive instances is varied.



Key Concepts:

True Positive Rate (Sensitivity): It is the proportion of actual positive instances that are correctly predicted as positive by the model.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

False Positive Rate: It is the proportion of actual negative instances that are incorrectly predicted as positive by the model.

$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$

ROC Curve Construction:

The ROC curve is created by plotting the true positive rate (sensitivity) on the y-axis against the false positive rate on the x-axis. Each point on the ROC curve represents a different threshold for classifying positive instances. A diagonal line (line of no-discrimination) represents a random classifier.

Interpretation of ROC Curve:

Higher ROC Curve: A curve that is closer to the upper-left corner of the plot indicates better performance, as it signifies a higher true positive rate for a given false positive rate.

Area Under the ROC Curve (AUC-ROC): AUC-ROC quantifies the overall performance of a classification model. A perfect classifier would have an AUC-ROC value of 1, while a random or poor classifier would have an AUC-ROC value close to 0.5.

Choosing a Model using ROC Curve:

When comparing multiple models, the one with the higher AUC-ROC is generally considered better. However, the choice of model should also consider the specific context and priorities. A point on the ROC curve can be chosen as the operating point based on the trade-off between sensitivity and specificity that aligns with the problem's requirements.

Advantages of ROC Curve:

Threshold Flexibility: ROC curve illustrates the impact of varying classification thresholds, providing insights into different operating points.

Imbalanced Datasets: ROC curve is effective even when dealing with imbalanced datasets, where one class significantly outnumbers the other.

Drawbacks of ROC Curve:

Limited to Binary Classification: ROC curve is designed for binary classification problems and may need extensions for multi-class problems.

Doesn't Show Calibration: ROC curve doesn't reveal the actual probabilities or predicted scores, focusing solely on the discrimination capacity of the model.

Use Cases of ROC Curve:

Medical Diagnostics: Evaluating a diagnostic test's ability to correctly identify patients with a particular disease.

Fraud Detection: Assessing a fraud detection model's performance in correctly classifying fraudulent transactions.

In summary, the ROC curve is a valuable visualisation tool that helps evaluate and compare the performance of binary classification models, especially when there is a need to balance sensitivity and specificity at different thresholds. The AUC-ROC provides a single scalar metric to summarise a model's overall performance.

RMSE

Root Mean Squared Error (RMSE) is a widely used metric in machine learning and statistics to measure the accuracy of a predictive model, particularly for regression tasks. It quantifies the average magnitude of the differences between predicted values and actual values. Lower RMSE values indicate better model performance.

Calculation:

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

Interpretation:

The RMSE provides a measure of how well the predicted values match the actual values on average. It gives more weight to larger errors due to the squaring operation, and the square root operation makes the RMSE value comparable to the original data scale.

Advantages of RMSE:

Sensitivity to Magnitude: RMSE is sensitive to the magnitude of errors, penalising larger errors more heavily.

Easily Interpretable: The RMSE value is in the same units as the original data, making it easy to interpret.

Drawbacks of RMSE:

Sensitivity to Outliers: Outliers with large errors can significantly impact the RMSE.

Units Dependency: RMSE is dependent on the units of the data, which can make comparisons between different datasets challenging.

Use Cases of RMSE:

RMSE is commonly used in various fields, including:

Regression Analysis: Assessing the accuracy of regression models by measuring the differences between predicted and actual values.

Forecasting: Evaluating the performance of time series forecasting models by comparing predicted and observed values.

Image Analysis: Measuring the quality of image reconstruction algorithms by comparing pixel values.

Example:

Consider a regression model predicting house prices. If the RMSE is calculated as \$10,000, it means, on average, the predicted house prices differ from the actual prices by \$10,000.

In summary, Root Mean Squared Error (RMSE) is a valuable metric for assessing the accuracy of regression models. It provides insight into how well the predicted values match the actual values and is particularly useful when comparing different models or assessing the performance of forecasting algorithms.

MSE

Mean Squared Error (MSE) is a common metric used in machine learning and statistics to measure the average squared differences between predicted values and actual values. It quantifies the average magnitude of errors in a predictive model, particularly in regression tasks.

Calculation:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \tilde{y}_i)^2$$

Interpretation:

The MSE provides a measure of how well the predicted values match the actual values on average. It gives more weight to larger errors due to the squaring operation. However, the squared values do not have the same units as the original data.

Advantages of MSE:

Sensitivity to Errors: MSE is sensitive to the magnitude of errors, penalizing larger errors more heavily.

Mathematical Properties: The use of squared errors simplifies mathematical calculations and optimizations.

Drawbacks of MSE:

Units Dependency: MSE is dependent on the units of the data, which can make comparisons between different datasets challenging.

Outliers Impact: Outliers with large errors can significantly impact the MSE.

Use Cases of MSE:

MSE is commonly used in various fields, including:

Regression Analysis: Assessing the accuracy of regression models by measuring the average squared differences between predicted and actual values.

Loss Function: MSE is often used as a loss function in training machine learning models.

Example:

Consider a regression model predicting exam scores. If the MSE is calculated as 100, it means, on average, the squared difference between predicted and actual scores is 100.

In summary, Mean Squared Error (MSE) is a useful metric for evaluating the performance of regression models. It provides insight into the average magnitude of errors and is commonly used as a loss function during model training. However, it's important to consider its limitations, such as sensitivity to outliers and dependence on data units.

MAE

Mean Absolute Error (MAE) is a metric used in machine learning and statistics to measure the average absolute differences between predicted values and actual values. It quantifies the average magnitude of errors in a predictive model, particularly in regression tasks.

Calculation:

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

Interpretation:

The MAE provides a measure of how well the predicted values match the actual values on average, without considering the direction of errors. It is less sensitive to outliers compared to squared error metrics like MSE.

Advantages of MAE:

Robustness to Outliers: MAE is less sensitive to outliers than squared error metrics like MSE, as it considers the absolute differences.

Interpretability: MAE is easy to interpret since it represents the average absolute error in the same units as the original data.

Drawbacks of MAE:

Lack of Squaring: Unlike squared error metrics, MAE does not penalise larger errors as heavily, which can be a drawback in some cases.

Use Cases of MAE:

MAE is commonly used in various fields, including:

Regression Analysis: Assessing the accuracy of regression models by measuring the average absolute differences between predicted and actual values.

Example:

Consider a regression model predicting temperature. If the MAE is calculated as 2 degrees Celsius, it means, on average, the absolute difference between predicted and actual temperatures is 2 degrees.

In summary, Mean Absolute Error (MAE) is a valuable metric for evaluating the performance of regression models. It provides insight into the average magnitude of errors and is particularly useful when sensitivity to outliers is a concern. However, it's important to consider its limitations, such as the lack of penalization for larger errors.

K-MEANS CLUSTERING

K-Means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into a specified number of distinct, non-overlapping groups or clusters. It aims to identify patterns and similarities in the data without any predefined labels.

Algorithm:

Initialization: Choose the number of clusters k and randomly initialise k cluster centroids in the feature space.

Assignment Step: Assign each data point to the nearest centroid, forming k clusters.

Update Step: Recalculate the centroids of each cluster based on the mean of the data points assigned to it.

Repeat: Repeat the assignment and update steps until convergence, which occurs when the centroids stabilise or a maximum number of iterations is reached.

Objective Function:

K-Means aims to minimise the sum of squared distances between data points and their assigned cluster centroids, often referred to as the "inertia" or "within-cluster sum of squares."

The diagram shows the objective function formula for K-Means clustering: $J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$. Annotations include:

- number of clusters** pointing to k in the first summation.
- number of cases** pointing to n in the second summation.
- case i** pointing to $x_i^{(j)}$.
- centroid for cluster j** pointing to c_j .
- Distance function** pointing to the term $\|x_i^{(j)} - c_j\|^2$.
- objective function** pointing to the entire equation $J = \dots$.

Interpretation:

K-Means clustering divides data points into clusters, where each cluster is represented by a centroid. It seeks to minimise the distance between data points and their assigned cluster centroids.

Advantages of K-Means:

Simplicity: K-Means is easy to understand and implement.

Scalability: It works well with large datasets.

Speed: K-Means can be efficient for many cases.

Drawbacks of K-Means:

Sensitive to Initialization: Different initializations can lead to different results.

Assumes Spherical Clusters: It assumes that clusters are spherical and equally sized.

Use Cases of K-Means:

Customer Segmentation: Grouping customers based on purchasing behaviour.

Image Compression: Reducing the number of colours in an image.

Anomaly Detection: Identifying outliers or anomalies in a dataset.

Example:

Suppose we have a dataset of customer spending habits. We want to cluster customers into three groups for targeted marketing. After applying K-Means, we find that customers are grouped into three clusters based on their spending patterns.

In summary, K-Means Clustering is a versatile algorithm used for unsupervised grouping of data points into clusters. It aims to minimise the distance between data points and cluster centroids and is widely used in various domains for pattern recognition and data exploration.

ELBOW METHOD

The Elbow Method is a technique used to determine the optimal number of clusters (k) for a K-Means Clustering algorithm. It helps find a balance between overfitting and underfitting the data by plotting the explained variance as a function of the number of clusters.

Steps:

Run K-Means: Apply the K-Means algorithm to the dataset for a range of k values.

Calculate Variance: For each k, calculate the sum of squared distances (inertia) between data points and their cluster centroids.

Plot Elbow Curve: Plot the inertia values against the corresponding k values.

Identify Elbow Point: Examine the plot. The "elbow point" is where the reduction in inertia begins to slow down, forming an "elbow" shape.

Interpretation:

The elbow point on the curve represents the point of diminishing returns in terms of explaining the variance. It suggests a suitable number of clusters where adding more clusters doesn't significantly reduce inertia, leading to a balance between model complexity and data fitting.

Use Cases of Elbow Method:

The Elbow Method is commonly used to determine the optimal number of clusters in K-Means Clustering for various applications, such as customer segmentation, image compression, and data exploration.

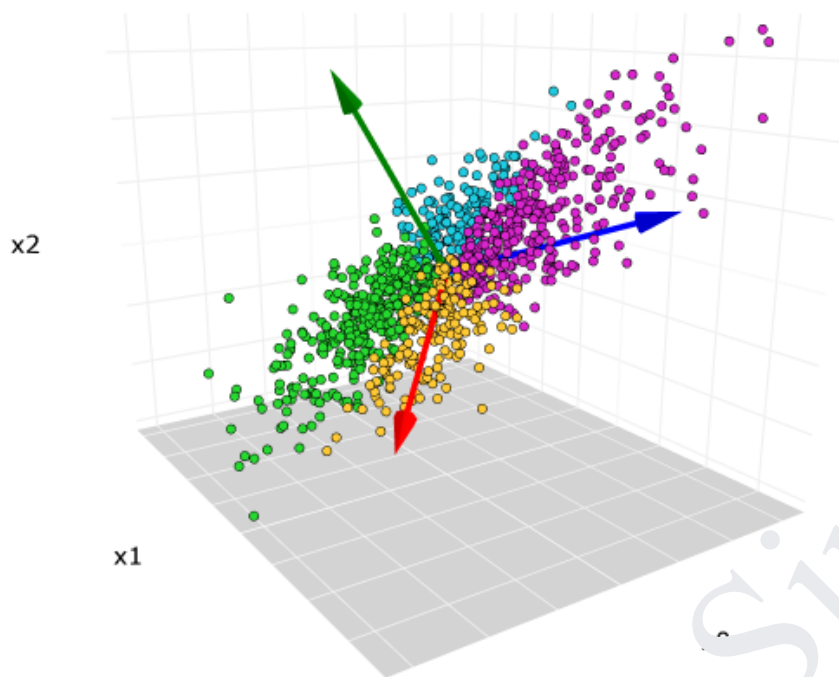
Example:

Suppose we have a dataset of customer spending habits and want to find the optimal number of customer segments. By applying the Elbow Method, we plot the inertia values against different k values and identify the point where the curve starts to flatten. If the elbow point is at $k=3$, it suggests that three clusters adequately explain the variance in the data.

In summary, the Elbow Method is a useful technique for selecting the optimal number of clusters in K-Means Clustering by analysing the trade-off between model complexity and data fitting.

PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in machine learning and statistics. It transforms high-dimensional data into a lower-dimensional space while preserving as much of the original variability as possible. PCA is particularly valuable for data visualisation, noise reduction, and feature extraction.



Key Concepts:

Variance and Covariance: PCA operates by maximising the variance of the projected data points while minimising the covariance between them.

Eigenvalues and Eigenvectors: PCA involves computing the eigenvalues and corresponding eigenvectors of the covariance matrix of the data. Eigenvectors represent the directions of maximum variance, and eigenvalues indicate the amount of variance along those directions.

PCA Algorithm

Standardise Data: Centre the data by subtracting the mean from each feature and optionally scaling to unit variance.

Compute Covariance Matrix: Calculate the covariance matrix of the standardised data.

Compute Eigenvectors and Eigenvalues: Solve the eigenvalue problem to find the eigenvectors and eigenvalues of the covariance matrix.

Select Principal Components: Sort the eigenvalues in descending order and select the top k eigenvectors corresponding to the largest eigenvalues, where k is the desired lower dimension.

Project Data: Multiply the original data by the selected eigenvectors to obtain the lower-dimensional representation.

Variance Explained:

The proportion of the total variance explained by each principal component is given by the ratio of its eigenvalue to the sum of all eigenvalues. This helps in assessing the importance of each component.

Advantages of PCA:

Dimensionality Reduction: PCA reduces the number of features while retaining the most important information.

Noise Reduction: High-variance noise is often captured by later principal components and can be filtered out.

Visualisation: PCA facilitates data visualisation in lower-dimensional space.

Drawbacks of PCA:

Interpretability: Principal components are linear combinations of original features, which may make them less interpretable.

Non-Linearity: PCA may not capture complex nonlinear relationships in the data.

Use Cases of PCA:

Image Compression: Reducing the dimensions of image data while preserving essential features.

Feature Extraction: Creating informative features from high-dimensional data.

Data Visualization: Visualising data in two or three dimensions.

Example:

Consider a dataset of various physical measurements of objects. By applying PCA, we can find the principal components that capture the most significant variations in the data. These components might represent dimensions like size, weight, and density, allowing us to reduce the data's dimensionality while retaining essential characteristics.

In summary, Principal Component Analysis (PCA) is a powerful technique for dimensionality reduction, noise reduction, and data visualisation. It transforms high-dimensional data into a lower-dimensional space, facilitating improved understanding and analysis.

NEURAL NETWORKS

Neural Networks, often referred to as Artificial Neural Networks (ANNs), are a fundamental concept in machine learning and artificial intelligence. They are inspired by the structure and function of the human brain's interconnected neurons and are used to model complex relationships and patterns in data.

Components of a Neural Network:

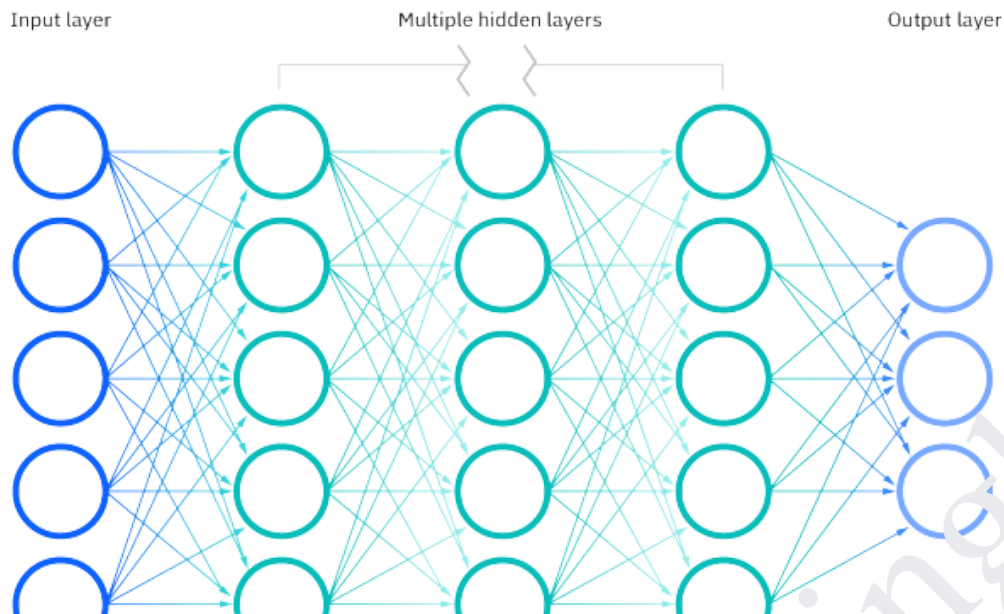
Neurons (Nodes): Neurons are the fundamental units of a neural network. Each neuron processes input data and produces an output.

Layers: Neurons are organised into layers. The three main types of layers are:

- **Input Layer:** Receives the raw input data.
- **Hidden Layers:** Intermediate layers that perform transformations on the data.
- **Output Layer:** Produces the final prediction or output.

Weights: Each connection between neurons has an associated weight that determines the strength of the connection. Weights are learned during the training process.

Activation Function: An activation function is applied to the weighted sum of inputs at each neuron. It introduces non-linearity into the network, enabling it to learn complex relationships.



Feedforward Process:

Input: Data is fed into the input layer.

Forward Propagation: The data flows through the hidden layers, and computations are performed using weights and activation functions.

Output: The final prediction or output is produced by the output layer.

Training Process:

Loss Function: A loss function measures the difference between the predicted output and the actual target. It quantifies how well the network is performing.

Backpropagation: The network adjusts its weights to minimise the loss. This is done by propagating the error backward from the output layer to the input layer and updating the weights using optimization algorithms like Gradient Descent.

Epochs: The training process is repeated over multiple iterations called epochs to refine the model's weights.

Deep Neural Networks (DNNs):

When neural networks have multiple hidden layers, they are referred to as Deep Neural Networks (DNNs). Deep learning involves training deep networks to automatically learn hierarchical features from the data.

Advantages of Neural Networks:

Non-Linearity: Neural networks can capture complex nonlinear relationships in data.

Feature Learning: DNNs can automatically learn relevant features from raw data.

Flexibility: Neural networks can be applied to a wide range of tasks, including image recognition, natural language processing, and reinforcement learning.

Drawbacks of Neural Networks:

Training Complexity: Deep networks can require a significant amount of computational resources and training time.

Overfitting: Complex neural networks are prone to overfitting on small datasets.

Black Box: The internal workings of deep networks can be challenging to interpret.

Use Cases of Neural Networks:

Image Recognition: Identifying objects and patterns in images.

Natural Language Processing: Language translation, sentiment analysis, and chatbots.

Game Playing: Training agents for playing games using reinforcement learning.

Example:

Suppose we want to build a neural network to classify images of handwritten digits. The network takes pixel values of the images as input, learns relevant features through hidden layers, and produces a probability distribution over possible digit classes at the output layer.

In summary, Neural Networks are a foundational concept in machine learning, enabling the modelling of complex relationships in data. They consist of interconnected neurons organised into layers, and through training, they can learn to make accurate predictions or classifications.

Implementation of Naive Bayes on a given .csv file (breast cancer dataset) :-

ABOUT DATASET

Description:

The Kaggle Breast Cancer Dataset is a widely used dataset in machine learning for classification tasks related to breast cancer diagnosis. It contains features extracted from digitised images of fine needle aspirates (FNAs) of breast mass. The dataset is used to predict whether a tumour is malignant (cancerous) or benign (non-cancerous) based on the extracted features.

Features:

The dataset contains the following features:

Radius Mean: Mean of distances from centre to points on the perimeter.

Texture Mean: Standard deviation of grey-scale values.

Perimeter Mean: Perimeter of the tumour.

Area Mean: Area of the tumour.

Smoothness Mean: Variation in radius lengths.

Compactness Mean: Compactness of the tumour.

Concavity Mean: Severity of concave portions of the contour.

Concave Points Mean: Number of concave portions of the contour.

Symmetry Mean: Symmetry of the tumour.

Fractal Dimension Mean: "Coastline approximation" - 1.

(Each of these features is also measured for the "Worst" and "Standard Error" variations, resulting in a total of 30 features.)

Applications:

The Kaggle Breast Cancer Dataset has applications in medical diagnostics, particularly in the field of oncology:

Breast Cancer Diagnosis: The primary use case is in classifying tumours as malignant or benign based on the extracted features. Machine learning models trained on this dataset can assist doctors in making accurate and early diagnoses.

Feature Engineering and Selection: Researchers and data scientists can use this dataset to explore various feature engineering techniques to extract relevant information from medical images.

Machine Learning Algorithms: The dataset is used to train and test various classification algorithms such as Logistic Regression, Support Vector Machines, Random Forests, and Neural Networks, helping to determine which algorithm performs best for this specific diagnostic task.


Model Evaluation: The dataset is also used for evaluating the performance of different models using metrics like accuracy, precision, recall, and F1-score.

Example:

A machine learning model trained on the Kaggle Breast Cancer Dataset could help radiologists and oncologists in distinguishing between malignant and benign tumours based on the extracted features. This can lead to more efficient diagnosis and potentially early detection of breast cancer, which is crucial for successful treatment.

In summary, the Kaggle Breast Cancer Dataset provides valuable data for developing and evaluating machine learning models for breast cancer diagnosis. Its features are derived from medical images and have significant applications in medical research and healthcare.

SOURCE CODE WITH THE OBTAINED OUTPUT

Jupyter Naive Bayes_ML_practical_13_code Last Checkpoint: a minute ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Code

```
In [8]: from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# Train the Gaussian Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = gnb.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Gaussian Naive Bayes model accuracy: {:.2f}%".format(accuracy * 100))
```

Gaussian Naive Bayes model accuracy: 94.30%

```
In [9]: from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import numpy as np
```

```
In [10]: # Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target
feature_names = data.feature_names
target_names = data.target_names
```

```
In [11]: # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# Train the Gaussian Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Out[11]: GaussianNB()

```
In [12]: # Make predictions on the testing set
y_pred = gnb.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Gaussian Naive Bayes model accuracy: {:.2f}%".format(accuracy * 100))
```

Gaussian Naive Bayes model accuracy: 94.30%

```
In [13]: # Print classification report
print("\nClassification Report:")
report = metrics.classification_report(y_test, y_pred, target_names=target_names)
print(report)
```

Classification Report:				
	precision	recall	f1-score	support
malignant	0.91	0.93	0.92	80
benign	0.96	0.95	0.96	148
accuracy			0.94	228
macro avg	0.94	0.94	0.94	228
weighted avg	0.94	0.94	0.94	228

```
In [14]: # Print confusion matrix
print("\nConfusion Matrix:")
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
Confusion Matrix:
[[ 74   6]
 [   7 141]]
```

```
In [15]: # Calculate class probabilities for a sample
sample = X_test[0]
class_probabilities = gnb.predict_proba([sample])[0]
class_labels = ["Negative", "Positive"]
print("\nClass Probabilities for Sample:")
for label, prob in zip(class_labels, class_probabilities):
    print("{}: {:.2f}%".format(label, prob * 100))
```

```
Class Probabilities for Sample:
Negative: 100.00%
Positive: 0.00%
```

```
In [25]: # Analyze feature importance
feature_importances = gnb.theta_[1]
sorted_indices = np.argsort(feature_importances)[-1:]
print("\nFeature Importance:")
for i in sorted_indices:
    print("{}: {:.2f}".format(feature_names[i], feature_importances[i]))
```

```
Feature Importance:
worst area: 547.28
mean area: 454.20
worst perimeter: 86.02
mean perimeter: 77.33
worst texture: 23.35
area error: 21.23
mean texture: 17.93
worst radius: 13.24
mean radius: 12.03
perimeter error: 2.02
texture error: 1.24
radius error: 0.29
worst symmetry: 0.27
worst compactness: 0.18
mean symmetry: 0.17
worst concavity: 0.16
worst smoothness: 0.12
mean smoothness: 0.09
mean compactness: 0.08
worst fractal dimension: 0.08
worst concave points: 0.01
mean fractal dimension: 0.01
mean concavity: 0.01
concavity error: 0.03
mean concave points: 0.01
compactness error: 0.02
symmetry error: 0.02
concave points error: 0.01
smoothness error: 0.01
fractal dimension error: 0.00
```