Course: Machine Learning– Fall 2024/2025

Course instructor: Rami Oran

By: Hamza Hazem Jabari

21110127

Machine Learning

Technical Report

Table of Contents

[1 Part 1 and Part 2 3](#_Toc188726327)

[1.1 Q 1) 3](#_Toc188726328)

[1.1.1 a) CEO Salary Analysis 3](#_Toc188726329)

[1.1.2 b) Product Success Prediction 3](#_Toc188726330)

[1.1.3 c) USD/Euro Exchange Rate Prediction 3](#_Toc188726331)

[1.1.4 Summary Table 4](#_Toc188726332)

[1.2 Q2) 4](#_Toc188726333)

[1.2.1 Lane Detection 5](#_Toc188726334)

[1.2.2 Traffic Sign Recognition 5](#_Toc188726335)

[1.2.3 Driving Pattern Analysis 5](#_Toc188726336)

[1.2.4 Collision Prediction 6](#_Toc188726337)

[1.2.5 Anomaly Detection 6](#_Toc188726338)

[1.2.6 Summary Table 6](#_Toc188726339)

[1.2.7 Key Takeaways 7](#_Toc188726340)

[1.3 Q3) 8](#_Toc188726341)

[1.4 Q4) 8](#_Toc188726342)

[1.5 Q5) 10](#_Toc188726343)

[1.6 Q6) 11](#_Toc188726344)

[1.7 Q7) 11](#_Toc188726345)

[1.8 Q8) 12](#_Toc188726346)

[1.9 Q9) 13](#_Toc188726347)

[1.10 Q10) 14](#_Toc188726348)

[1.11 Q11) 15](#_Toc188726349)

[1.12 Q12) 16](#_Toc188726350)

[2 Comprehensive Report: Customer Churn Prediction Using Machine Learning 17](#_Toc188726351)

[2.1 Introduction to Machine Learning 17](#_Toc188726352)

[2.2 Problem Statement 18](#_Toc188726353)

[3 Machine Learning Models Overview 19](#_Toc188726354)

[3.1 Logistic Regression 19](#_Toc188726355)

[3.2 Random Forest 20](#_Toc188726356)

[3.3 Gradient Boosting (XGBoost) 21](#_Toc188726357)

[3.4 Naive Bayes 22](#_Toc188726358)

[4 Model Development 23](#_Toc188726359)

[4.1 Data Preprocessing 23](#_Toc188726360)

[4.1.1 Hyperparameter Tuning 23](#_Toc188726361)

[4.2 Model Evaluation 23](#_Toc188726362)

[4.2.1 Performance Metrics 23](#_Toc188726363)

[4.2.2 Results 24](#_Toc188726364)

[4.2.3 Model-Specific Insights 24](#_Toc188726365)

[4.3 Strengths and Weaknesses Summary 24](#_Toc188726366)

[4.3.1 Future Improvements 24](#_Toc188726367)

[4.3.2 Conclusion 25](#_Toc188726368)

[4.4 Differences Between Bootstrap, Bagging, and Boosting 25](#_Toc188726369)

[4.4.1 Bootstrap 25](#_Toc188726370)

[4.4.2 Bagging (Bootstrap Aggregating) 26](#_Toc188726371)

[4.4.3 Boosting 26](#_Toc188726372)

[4.4.4 Comparison Table 26](#_Toc188726373)

[4.4.5 Key Differences 27](#_Toc188726374)

[4.4.6 Conclusion 27](#_Toc188726375)

[5 References 28](#_Toc188726376)

# Part 1 and Part 2

## Q 1)

﻿Explain whether each scenario below is a classification or a regression problem and indicate whether we are most interested in inference or prediction. Finally, provide *n* (number of observations) and *p* (number of features).

1. ﻿We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.
2. ﻿We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.
3. ﻿We are interested in predicting the % of change in the USD/Euro exchange rate in relation to the weekly changes in the world stock markets. ﻿Hence, we collect weekly data for all of 2012. For each week we record the % of change in the USD/Euro, the % change in the US market, the % change in the British market, and the % change in the German market.

### **a) CEO Salary Analysis**

* **Problem Type**: **Regression** (CEO salary is a continuous numerical value).
* **Focus**: **Inference** (Understanding which factors affect CEO salary).
* **n (observations)**: 500 (top firms).
* **p (features)**: 3 (profit, number of employees, industry).

**Explanation**:

* Regression is used because the target variable (CEO salary) is continuous.
* The goal is to understand relationships between features and salary (inference), not just predict salaries.

### **b) Product Success Prediction**

* **Problem Type**: **Classification** (Success/Failure is a binary outcome).
* **Focus**: **Prediction** (Determine success of a new product).
* **n (observations)**: 20 (previous products).
* **p (features)**: 13.

**Explanation**:

* Classification is required for binary outcomes (success or failure).
* The primary goal is to predict future outcomes for new products.

### **c) USD/Euro Exchange Rate Prediction**

* **Problem Type**: **Regression** (% change in exchange rate is continuous).
* **Focus**: **Prediction** (Forecast future exchange rate changes).
* **n (observations)**: 52 (weeks in 2012).
* **p (features)**: 3 (% change in US, British, and German markets).

**Explanation**:

* Regression is used to predict a continuous numerical value (% change).
* The focus is on forecasting future values based on historical trends.

### **Summary Table**

| **Scenario** | **Problem Type** | **Focus** | **n** | **p** |
| --- | --- | --- | --- | --- |
| (a) | Regression | Inference | 500 | 3 |
| (b) | Classification | Prediction | 20 | 13 |
| (c) | Regression | Prediction | 52 | 3 |

**Key Takeaways**:

* **Regression vs. Classification**: Determined by whether the target variable is continuous or categorical.
* **Inference vs. Prediction**: Depends on whether the goal is to understand relationships (inference) or forecast outcomes (prediction).

## Q2)

You are part of the university team developing an AI system for an autonomous car. The project involves handling several tasks to ensure the car can navigate safely and efficiently. Below are some of the tasks you need to solve:

1. **Lane Detection**: The car must identify lanes on the road using video feed from its cameras.
2. **Traffic Sign Recognition**: The system needs to classify and recognize traffic signs (e.g., stop signs, speed limits).
3. **Driving Pattern Analysis**: Analyze large datasets of driving patterns from previous trips to group similar driving behaviors for further analysis (e.g., aggressive vs. defensive driving).
4. **Collision Prediction**: Predict whether a collision is likely based on real-time sensor data (e.g., LIDAR and radar).
5. **Anomaly Detection**: Detect unusual patterns in sensor data, such as faulty readings from a sensor.

For each of the five tasks listed above:

1. Determine whether it is a **supervised** or **unsupervised** learning problem. Justify your reasoning.
2. Suggest a suitable machine learning algorithm for each task. Explain clearly why that algorithm is appropriate. Additionally, include the time complexity for both training and prediction of your chosen algorithm.
3. For each task evaluate why machine learning is essential for the overall design of the autnomous car?

### **Lane Detection**

**a) Learning Type**: **Supervised Learning**

* **Justification**: Requires labeled training data (images/videos with annotated lanes) to train the model to recognize lane markings.  
  **b) Algorithm**: **Convolutional Neural Networks (CNN)**
* **Why**: CNNs excel at image segmentation and feature extraction from visual data.
* **Time Complexity**:
  + Training: O(n⋅d⋅k) (depends on image size *d*, layers *k*, and data size *n*).
  + Prediction: O(d⋅k) (real-time processing feasible).  
    **c) ML Essential**: Enables generalization across diverse road conditions (e.g., lighting, weather) without manual rule-coding.

### **Traffic Sign Recognition**

**a) Learning Type**: **Supervised Learning**

* **Justification**: Needs labeled images of traffic signs (e.g., "stop," "speed limit") to classify new instances.  
  **b) Algorithm**: **CNN with Transfer Learning (e.g., ResNet)**
* **Why**: Pre-trained CNNs reduce training time while maintaining high accuracy.
* **Time Complexity**:
  + Training: *O*(*n*⋅*d*⋅*k*) (high due to fine-tuning).
  + Prediction: *O*(*d*⋅*k*) (efficient for real-time use).  
    **c) ML Essential**: Handles variations in sign appearance (e.g., occlusion, angles) better than rigid algorithms.

### **Driving Pattern Analysis**

**a) Learning Type**: **Unsupervised Learning**

* **Justification**: No labeled data; goal is to group similar behaviors (e.g., "aggressive" vs. "defensive").  
  **b) Algorithm**: **K-Means Clustering**
* **Why**: Simple and effective for partitioning data into clusters.
* **Time Complexity**:
  + Training: *O*(*n*⋅*k*⋅*d*⋅iterations) (scales with data size *n*, clusters *k*, features *d*).
  + Prediction: *O*(*k*⋅*d*) (fast cluster assignment).  
    **c) ML Essential**: Discovers hidden patterns in driving data to inform safety strategies.

### **Collision Prediction**

**a) Learning Type**: **Supervised Learning**

* **Justification**: Requires labeled historical data (collision vs. non-collision events) for training.  
  **b) Algorithm**: **Random Forest**
* **Why**: Handles high-dimensional sensor data and provides probabilistic outputs.
* **Time Complexity**:
  + Training: *O*(*n*log*n*⋅*m*) (for *m* trees).
  + Prediction: *O*(*m*⋅depth) (fast decision-making).  
    **c) ML Essential**: Processes real-time sensor data (LIDAR, radar) to predict collisions faster than human reaction times.

### **Anomaly Detection**

**a) Learning Type**: **Unsupervised Learning**

* **Justification**: Anomalies (e.g., faulty sensor readings) are rare and often unlabeled.  
  **b) Algorithm**: **Isolation Forest**
* **Why**: Efficiently isolates anomalies in high-dimensional data.
* **Time Complexity**:
  + Training: *O*(*n*) (linear time).
  + Prediction: *O*(1) (constant time per data point).  
    **c) ML Essential**: Detects subtle, unknown anomalies that rule-based systems might miss.

### **Summary Table**

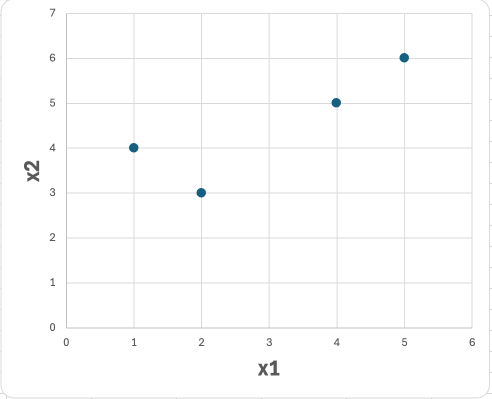
| **Task** | **Learning Type** | **Algorithm** | **Training Time Complexity** | **Prediction Time Complexity** | **Why ML is Essential?** |
| --- | --- | --- | --- | --- | --- |
| **Lane Detection** | Supervised | SVM, or deep learning algorithms | High *O*(*n*⋅*d*⋅*k*)) | Moderate (*O*(*d*⋅*k*)) | Effective for image classification |
| **Traffic Sign Recognition** | Supervised | SVM, or deep learning algorithms | High (*O*(*n*⋅*d*⋅*k*)) | Low (*O*(*d*⋅*k*)) | kernel-based separation in high-dimensional spaces |
| **Driving Pattern Analysis** | Unsupervised | K-Means | Moderate (*O*(*n*⋅*k*⋅*d*)) | Low (*O*(*k*⋅*d*)) | Simple yet powerful for clustering driving patterns and detecting anomalies. |
| **Collision Prediction** | Supervised | XGBoost | Moderate (*O*(*n*log*n*⋅*m*)) | Low (*O*(*m*⋅depth)) | Superior accuracy and speed for real-time sensor data analysis. |
| **Anomaly Detection** | Unsupervised | K-Means | Low (*O*(*n*)) | Very Low (*O*(1)) | Simple yet powerful for clustering driving patterns and detecting anomalies. |

### **Key Takeaways**

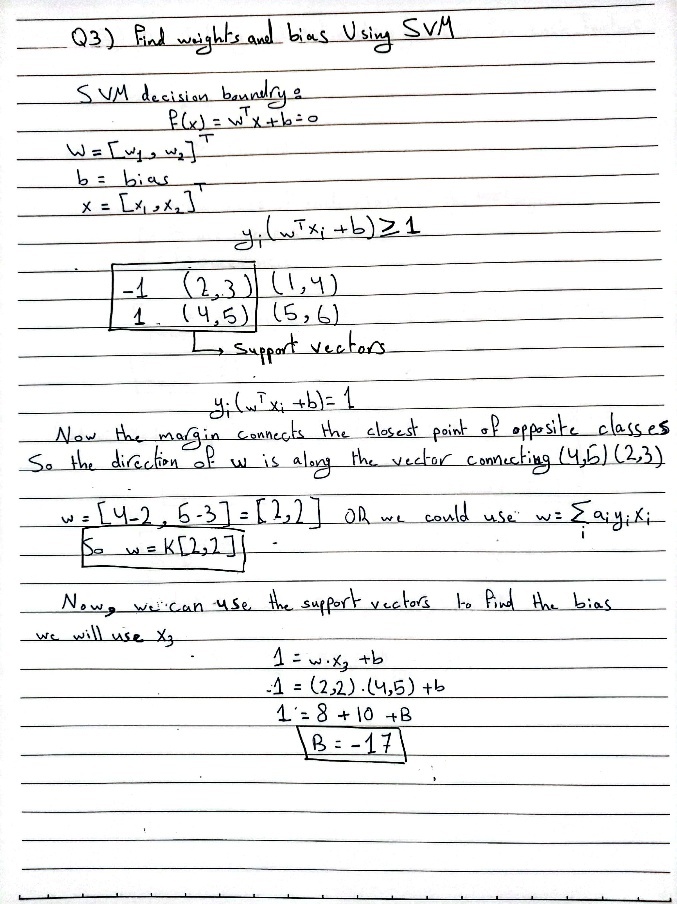
* **Supervised vs. Unsupervised**:
  + Supervised tasks require labeled data (e.g., collision labels, traffic sign categories).
  + Unsupervised tasks focus on pattern discovery without labels (e.g., driving behavior clustering).
* **Algorithm Selection**: Driven by data type (images vs. sensor data), real-time needs, and interpretability.
* **ML Necessity**: Enables adaptability to dynamic environments (e.g., varying road conditions) and real-time decision-making.

## Q3)

Based on the plot below, find the weights ***w*** and the bias ***b***based on a decision boundary found using SVM*.* Explain the steps followed in detail.



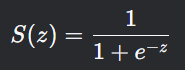
|  |  |  |
| --- | --- | --- |
| **x1** | **x2** | **Label** |
| 1 | 4 | -1 |
| 2 | 3 | -1 |
| 4 | 5 | 1 |
| 5 | 6 | 1 |



A notebook with writing on it

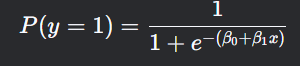
AI-generated content may be incorrect.

## Q4)

The **sigmoid function** (logistic function) is defined as:

It maps any real number *z* (from linear regression output *z*=*β*0​+*β*1​*x*1​+⋯+*βn*​*xn*​) to a value between **0 and 1**. This transformation is critical for solving **classification problems** because:

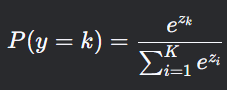
1. **Probability Interpretation**: The output S(z) represents the probability of belonging to class 1 (e.g., "churn").
2. **Non-Linear Decision Boundary**: The linear part (*z*) captures feature relationships, the sigmoid introduces non-linearity, enabling separation of classes.
3. **Bounded Output**: Unlike linear regression, which predicts unbounded values, the sigmoid ensures predictions are probabilities in [0,1] which is suitable for classification.

**Example**:  
Suppose we predict whether a student passes (y=1) or fails (y=0) based on study hours (*x*). Logistic regression computes *z*=*β*0​+*β*1​*x*, then applies the sigmoid:

If  P(*y*=1) ≥ 0.5, classify as "pass"; otherwise, "fail."

**Can Sigmoid Be Used for Multiclass Classification?**

**Yes, but with limitations**:

1. **Binary or Multilabel Classification**:
   * Sigmoid works for **binary** (one class vs. rest) or **multilabel** (multiple non-exclusive labels, e.g., "cat" and "dog" in an image).
   * Example: Use independent sigmoid outputs for each label.
2. **Mutually Exclusive Classes (Single-Label Multiclass)**:
   * Use the **softmax function** instead. It generalizes sigmoid for *K* classes by ensuring probabilities sum to 1:
   * Example: Classifying an image into "cat," "dog," or "bird."

**Key Differences**:

| **Scenario** | **Function** | **Output Range** | **Use Case** |
| --- | --- | --- | --- |
| Binary Classification | Sigmoid | [0,1] per class | Yes/No, Pass/Fail |
| Multiclass (Single-Label) | Softmax | Probabilities sum to 1 | Mutually exclusive classes (e.g., animal types) |
| Multilabel | Sigmoid | [0,1] per label | Multiple non-exclusive labels (e.g., tags) |

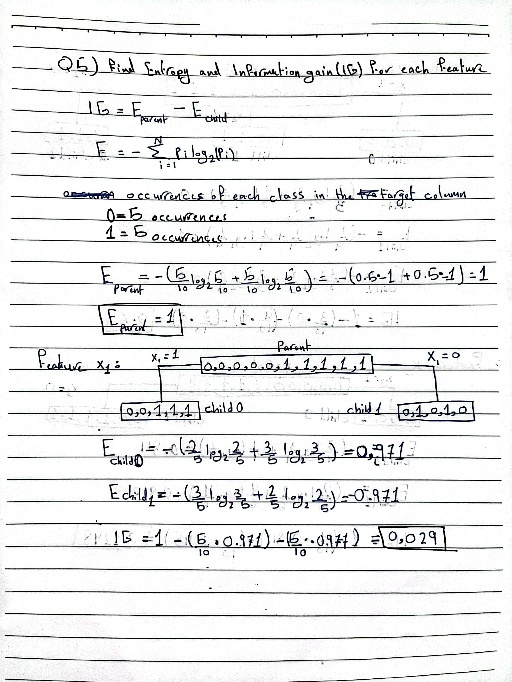
**Summary**

* **Sigmoid in Logistic Regression**: Transforms linear outputs to probabilities for **binary classification**.
* **Multiclass Adaptation**:
  + **Softmax**: Preferred for mutually exclusive classes.
  + **Sigmoid**: Used in multilabel or one-vs-rest approaches (but probabilities don’t sum to 1).

**Example of Sigmoid in Multilabel**:  
Predicting diseases in a patient (e.g., diabetes, hypertension, asthma) where multiple conditions can coexist. Each disease uses a separate sigmoid classifier.

## Q5)

Using the dataset in the table below, find the entropy and information gain (IG) for each feature. Then, continue with the same approach to construct and draw the full decision tree learned from this data (without any pruning).

A close-up of a paper with mathematical equations

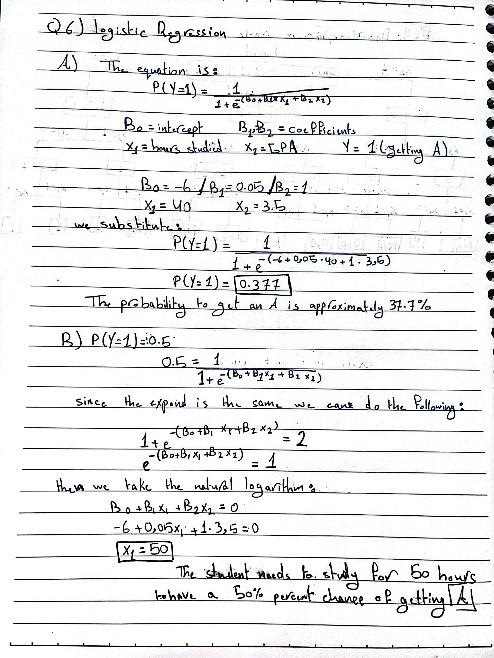
AI-generated content may be incorrect.A diagram of a flowchart

AI-generated content may be incorrect.A table with numbers and symbols

Description automatically generated

## Q6)

Suppose we collect data for a group of students in a class with variables   
*X1* = hours studied, *X2* = undergraduate GPA, and *Y* = receive an A. We fit a logistic regression model and produce **estimated** **coefficients** *B0* = -6, *B1* = 0.05, *B2* = 1

* 1. Estimate the probability that a student who studies for 40 hours and has an undergraduate GPA of 3.5 gets an A in the class.
  2. How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

## Q7)

***Consider a simple regression problem with the following data:***

|  |  |
| --- | --- |
| **Instance** | **True Value** |
| 1 | 10 |
| 2 | 15 |
| 3 | 20 |

You are using gradient boosting with the following setup:

1. **Initial Prediction:** f0(x)=12 for all instances.
2. **Learning Rate:** η=0.1.

**Tasks:**

1. Calculate the initial residuals after f0(x).
2. Update predictions after the first weak learner.
3. Compute the residuals for the second iteration.
4. Update predictions after the second weak learner.
5. A close-up of a paper

   AI-generated content may be incorrect.A close-up of a paper

   AI-generated content may be incorrect.Explain how the algorithm improves the prediction after each iteration?

## Q8)

A paper with writing on it

AI-generated content may be incorrect.A close-up of a paper

AI-generated content may be incorrect.Given the following classifier outputs and thresholds, calculate the True Positive Rate (TPR) and False Positive Rate (FPR) for the threshold of 0.6, and explain how these points contribute to the ROC curve.  Explain the purpose of the ROC curve? When is it favored to be used over a confusion matrix?

|  |  |  |
| --- | --- | --- |
| **Instance** | **True Label** | **Predicted Probability** |
| 1 | Positive | 0.8 |
| 2 | Negative | 0.7 |
| 3 | Positive | 0.65 |
| 4 | Negative | 0.4 |
| 5 | Positive | 0.3 |

## Q9)

What are the key signs of underfitting and overfitting during the model evaluation process? What are the primary causes of overfitting, and how can it be mitigated? Provide at least three techniques

**Underfitting**:

* **Signs**:
  1. **High Training Error**: The model performs poorly, especially on the training data.
  2. **High Validation/Test Error**: Weak generalization to unseen data.
  3. **Very Simple Model**: The model fails to capture basic patterns in the data (e.g., using a linear model for non-linear data).
* **Example**: A decision tree with a depth of 1 applied to a complex dataset.

**Overfitting**:

* **Signs**:
  1. **Low to Non-Training Error**: Near-perfect (accuracy =1) performance on training data.
  2. **High Validation/Test Error**: Huge drop in performance on new data.
  3. **Large Gap Between Training and Validation Metrics**: Point to memorization of noise in training data.
* **Example**: A deep neural network with many parameters trained on a small dataset.

**Primary Causes of Overfitting**

1. **Excessive Model Complexity**:
   * A model with too many parameters (e.g., high-degree polynomials, deep trees) learns noise instead of patterns.
2. **Insufficient Training Data**:
   * Limited data fails to represent the true distribution (generalization), leading to memorization.
3. **Noisy or Irrelevant Features**:
   * Redundant or irrelevant features confuse the model.

**Mitigation Techniques for Overfitting**

1. **Regularization**:
   * **How**: Add penalty terms (e.g., L1/L2) to the loss function to constrain model weights.
   * **Example**:
     + L1 (Lasso): Encourages sparsity by shrinking less important features to zero.
     + L2 (Ridge): Reduces the magnitude of all weights.
   * **Why**: Simplifies the model by discouraging overly complex solutions.
2. **Cross-Validation**:
   * **How**: Split data into multiple folds (e.g., k-fold) to validate performance across different subsets.
   * **Example**: 5-fold cross-validation ensures the model generalizes well across all partitions.
   * **Why**: Identifies overfitting early by testing robustness on unseen splits.
3. **Pruning (for Decision Trees)**:
   * **How**: Remove branches that provide little predictive power (e.g., based on significance thresholds).
   * **Example**: Cutting off nodes that only improve accuracy by < 1% on training data.
   * **Why**: Reduces tree depth, limiting the model’s ability to fit noise.

**Summary**

* **Underfitting**: Fix by increasing model complexity, adding features, or reducing regularization.
* **Overfitting**: Address by simplifying the model, collecting more data, or applying regularization/cross-validation.

By balancing model complexity and data sufficiency, you ensure robust generalization to new data.

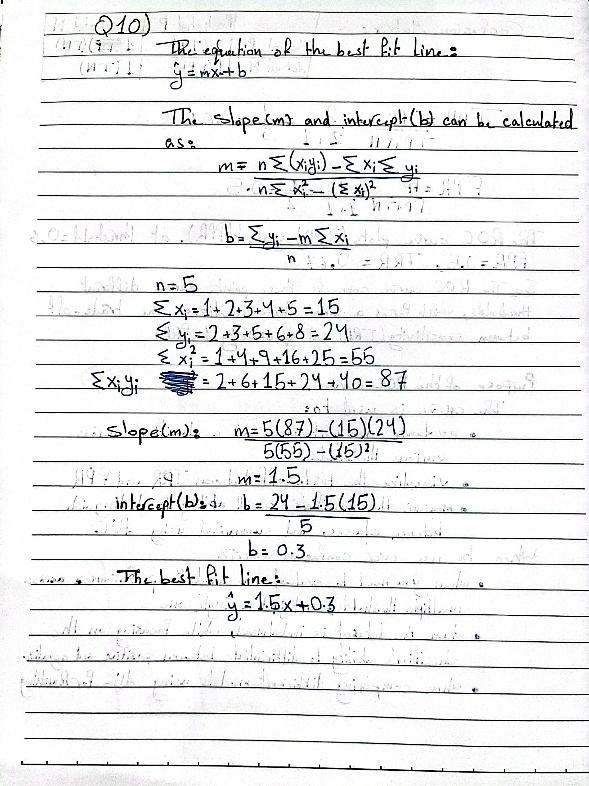
## Q10)

You are given the following dataset for simple linear regression:

|  |  |
| --- | --- |
| **x** | **y** |
| 1 | 2 |
| 2 | 3 |
| 3 | 5 |
| 4 | 6 |
| 5 | 8 |

**Tasks:**

1. Calculate the **slope (m)** and **intercept (b)** of the best-fit line using the least squares method (closed form expression).
2. Write the equation of the best-fit line =mx+b.



## Q11)

You are given the following data for a regression model:

|  |  |
| --- | --- |
| **Observed (y)** | **Predicted ()** |
| 3 | 2.8 |
| 4 | 4.2 |
| 5 | 5.0 |
| 6 | 6.1 |
| 7 | 6.9 |

**Tasks:**

1. Calculate the **mean of the observed values**
2. Compute the **total sum of squares (SST)**
3. Calculate the **residual sum of squares (SSE)**
4. Calculate R2 **(coefficient of determination)** ​
5. A close-up of a notebook

   AI-generated content may be incorrect.Interpret the value of R2.

## Q12)

You are given the following dataset with a single independent variable X and a dependent variable Y:

|  |  |
| --- | --- |
| **X** | **Y** |
| 1 | 2 |
| 2 | 5 |
| 3 | 10 |
| 4 | 17 |
| 5 | 26 |

You are tasked with fitting two different polynomial regression models and determining which one provides the best fit to the data. The models are as follows:

* Model 1 (2nd-degree polynomial):

=β0 + β1.X + β2.X2

The coefficients for Model 1 are given as:

β0​=1, β1​=1, β2​=1

* Model 2 (3rd-degree polynomial):

=β0 + β1.X + β2.X2 + β3.X3

The coefficients for Model 2 are given as:

β0​=0, β1​=1, β2​=1, β3​=−0.2

Tasks:

1. Step 1: Calculate the predicted values for both models using the given coefficients.
2. Step 2: Calculate the R2 (coefficient of determination) for both models.
3. Step 3: Compare the R2 values of both models.
4. A close-up of a paper

   AI-generated content may be incorrect.A close-up of a paper with mathematical equations

   AI-generated content may be incorrect.Step 4: Interpret the results.

# **Comprehensive Report: Customer Churn Prediction Using Machine Learning**

## **Introduction to Machine Learning**

Machine Learning (ML) is a subset of artificial intelligence that enables systems to learn patterns from data and make predictions or decisions without explicit programming. It is broadly categorized as:

* **Supervised Learning**: Models learn from labeled data (e.g., classification, regression).
* **Unsupervised Learning**: Models identify patterns in unlabeled data (e.g., clustering).
* **Reinforcement Learning**: Models learn through trial and error using feedback.

**Applications**: ML is important in healthcare, finance, marketing, and customer analytics. For businesses, predicting customer churn (i.e., identifying customers likely to cancel subscriptions) is critical to reduce attrition and enhance retention strategies.

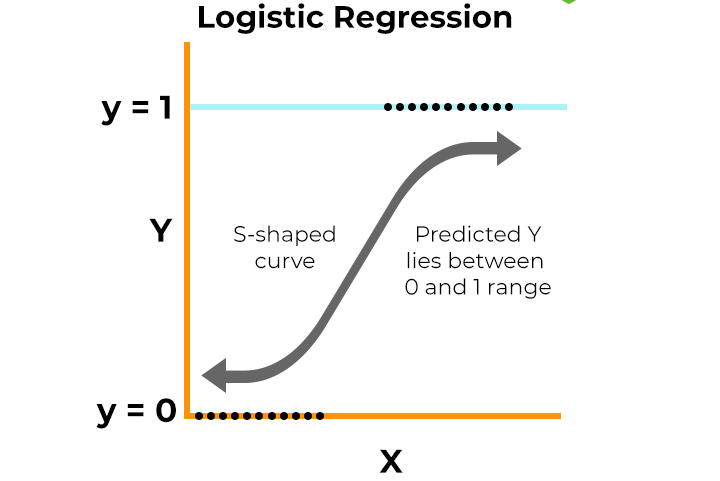
## **Problem Statement**

**Objective**: Predict whether a customer will churn (binary classification) using historical data.

* **Target Variable**: Churned (0 = retained, 1 = churned).
* **Dataset**: The dataset consists of 5,000 customer records, capturing various aspects of their demographic, behavioral, and transactional information. It includes details such as customer age, gender, subscription length, region, and payment method. Additionally, the dataset tracks customer interactions, such as the number of support tickets raised and satisfaction scores, as well as financial data like monthly spend and discounts offered. The "Last\_Activity" field indicates how recently a customer has engaged with the service. Finally, the dataset includes a churn indicator, showing whether a customer has ended their subscription. Some variables, like Age and Satisfaction\_Score, contain missing values. This dataset provides valuable insights into customer behavior and can be used for analyzing churn patterns, satisfaction levels, and spending trends.
* The dataset contains information about 5,000 customers and includes the following 12 variables:
* Customer\_ID: A unique identifier for each customer (object type).  
  Age: The age of the customer (float64), with some missing values.  
  Gender: The gender of the customer (object type).  
  Subscription\_Length: The length of time (in months) the customer has been subscribed (int64).  
  Region: The region where the customer is located (object type).  
  Payment\_Method: The method used by the customer to make payments (object type).  
  Support\_Tickets\_Raised: The number of support tickets raised by the customer (int64).  
  Satisfaction\_Score: A score indicating customer satisfaction (float64), with some missing values.  
  Discount\_Offered: The discount offered to the customer (float64).  
  Last\_Activity: The number of days since the customer last interacted with the service (int64).  
  Monthly\_Spend: The amount spent by the customer per month (float64).  
  Churned: Indicates whether the customer has churned (1 = yes, 0 = no) (int64).
* The source of the data is Kaggle at: <https://www.kaggle.com/datasets/akashanandt/streaming-service-data>

# **Machine Learning Models Overview**

## **Logistic Regression**



**Definition**:  
A **classification algorithm** used to predict the probability of a binary outcome (e.g., 0, 1) by modeling the relationship between input features and the log-odds of the target variable.

**How It Works**:

1. **Linear Equation**: Compute a weighted sum of input features:

*z*=*β*0​+*β*1​*x*1​+*β*2​*x*2​+⋯+*βn*​*xn*​

1. **Sigmoid Function**: Map *z* to a probability between 0 and 1:

​A number with white text

AI-generated content may be incorrect.

1. **Decision Boundary**: Classify as *y*=1 if *P*(*y*=1)≥0.5; otherwise, *y*=0.

**Strengths**:

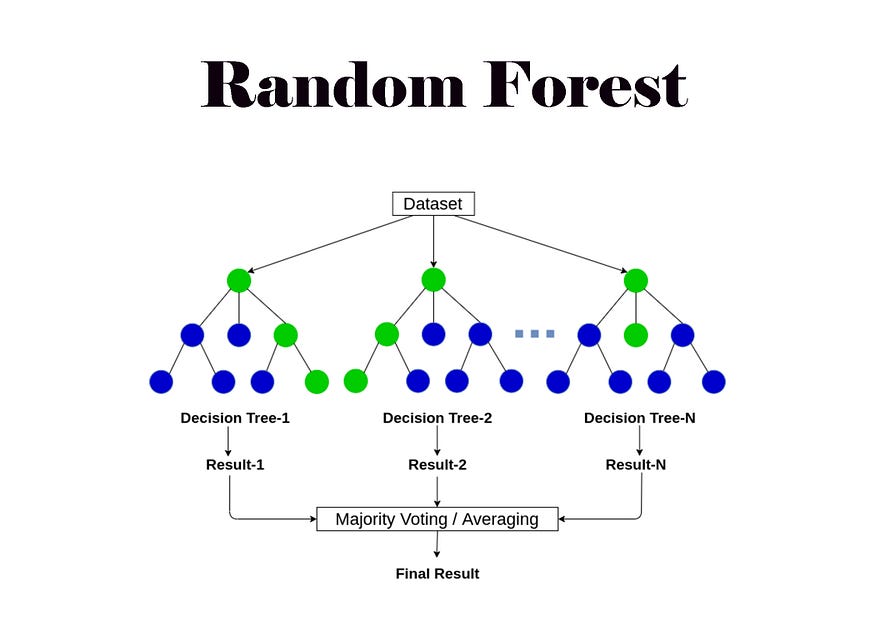
* Interpretable (coefficients show feature importance).
* Fast to train.

**Weaknesses**:

* Assumes linear relationship between features and log-odds.

**Use Case:** Predicting customer churn (binary classification).

## **Random Forest**



**Definition**:  
An **ensemble method** that builds multiple decision trees and aggregates their predictions (through majority vote for classification or averaging for regression).

**How It Works**:

1. **Bagging**: Train each tree on a random subset of the data (with replacement).
2. **Feature Randomness**: At each split, consider a random subset of features.
3. **Aggregation**: Combine predictions from all trees to reduce variance.

**Use Case**: Handling high-dimensional data.

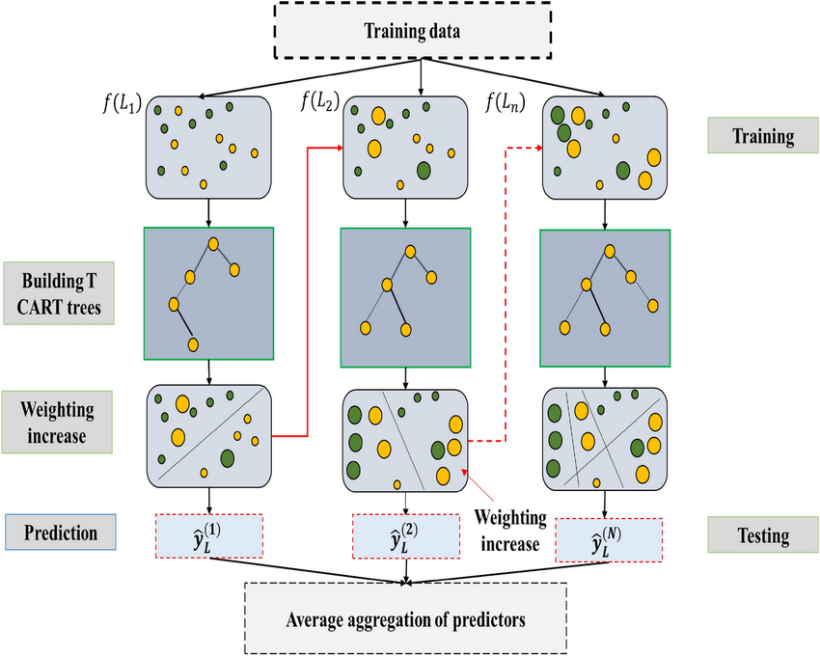
**Strengths**:

* Robust to overfitting.
* Handles non-linear relationships.

**Weaknesses**:

* Computationally expensive.
* Less interpretable than linear models.

## **Gradient Boosting (XGBoost)**



**Definition**:  
An **ensemble method** that builds decision trees sequentially, where each new tree corrects errors made by previous trees.

**How It Works**:

1. **Initial Prediction**: Start with a simple model (e.g., mean of the target).
2. **Residual Calculation**: Compute errors (residuals) from the current model.
3. **Train Next Tree**: Fit a new tree to predict the residuals.
4. **Update Model**: Combine predictions from all trees using a learning rate η*η*:

*Fm*​(*x*)=*Fm*−1​(*x*)+*η*⋅*hm*​(*x*)  
where *hm*​(*x*) is the new tree.

**Use Case**: High-accuracy tasks like customer churn prediction.

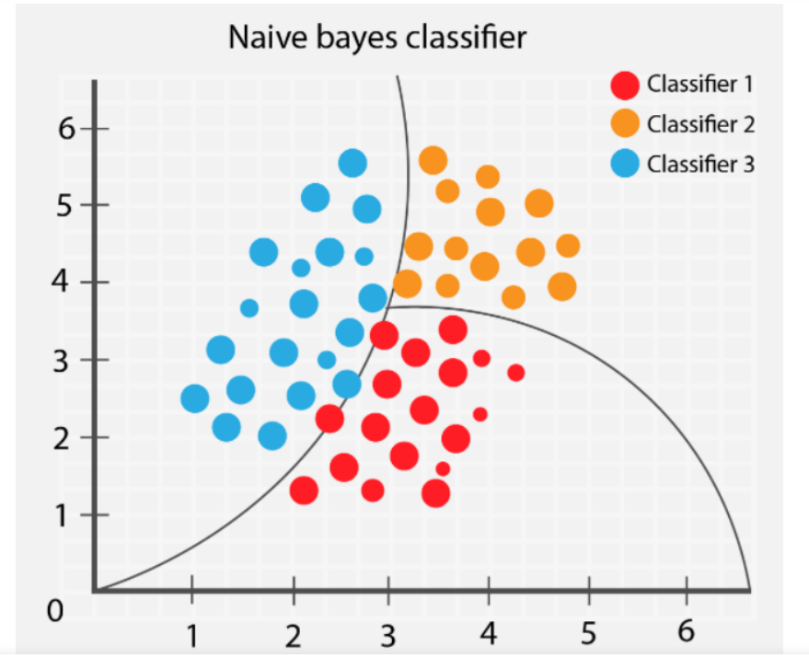
**Strengths**:

* State-of-the-art performance.
* Handles missing values and outliers.

**Weaknesses**:

* Requires careful hyperparameter tuning.

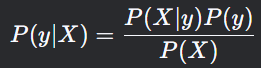
## **Naive Bayes**



**Definition**:  
A **probabilistic classifier** based on Bayes’ theorem, assuming feature independence (the "naive" assumption).

**How It Works**:

1. **Bayes’ Theorem**: Calculate posterior probability:



1. **Feature Independence**: Simplify *P*(*X*∣*y*) as:

*P*(*X*∣*y*)=*P*(*x*1​∣*y*)⋅*P*(*x*2​∣*y*)⋅⋯⋅*P*(*xn*​∣*y*)

1. **Prediction**: Choose the class with the highest *P*(*y*∣*X*).

**Use Case**: Text classification (e.g., spam detection).

**Strengths**:

* Fast and efficient for small datasets.
* Works well with high-dimensional data.

**Weaknesses**:

* Fails if features are correlated.

**Summary**

| **Model** | **Type** | **Key Mechanism** | **Best For** |
| --- | --- | --- | --- |
| **Logistic Regression** | Linear | Sigmoid probability mapping | Binary classification tasks |
| **Random Forest** | Ensemble | Bagging + feature randomness | Complex, noisy data |
| **Gradient Boosting** | Ensemble | Sequential error correction | High-accuracy prediction |
| **Naive Bayes** | Probabilistic | Bayes’ theorem + independence assumption | Text/quick classification tasks |

# **Model Development**

## **Data Preprocessing**

* **Handling Missing Values**:
  + Numeric: Filled with median (e.g., Age).
  + Categorical: Filled with mode.
* **Feature Engineering**:
  + Numeric features scaled using StandardScaler.
  + Categorical features are encoded with OneHotEncoder.

### **Hyperparameter Tuning**

* **GridSearchCV**: Thorough search over predefined hyperparameters with 5-fold cross-validation.
* **Key Hyperparameters**:

| **Model** | **Hyperparameter** | **Role** |
| --- | --- | --- |
| Logistic Regression | C (Regularization) | Controls overfitting |
| Random Forest | n\_estimators | Number of decision trees |
| Gradient Boosting | learning\_rate | Step size for error correction |
| Naive Bayes | var\_smoothing | Stabilizes variance estimates |

## **Model Evaluation**

### **Performance Metrics**

* **Accuracy**: Overall correctness (suitable for balanced data).
* **Precision**: Proportion of true positives among predicted positives (avoids false alarms).
* **Recall**: Proportion of actual positives correctly identified (critical for churn detection).
* **F1 Score**: Harmonic mean of precision and recall.
* **ROC-AUC**: Measures separability of classes (robust to imbalance).

### **Results**

| **Model** | **Accuracy (Test)** | **ROC-AUC** | **Training Time** |
| --- | --- | --- | --- |
| Gradient Boosting | **99.40%** | 0.**9999** | High (5.74) |
| Random Forest | 99.13% | 0.9998 | Moderate (2.78) |
| Logistic Regression | 81.00% | 0.8500 | Low (0.21) |
| Naive Bayes | 79.55% | 0.8200 | **Very Low (0.05)** |

**Key Observations**:

* **Gradient Boosting** outperformed the rest of the models due to its iterative error-correction mechanism.
* Near-perfect AUC (0.9999) suggests exceptional class separation.

### **Model-Specific Insights**

* **Gradient Boosting**:
  + **Feature Importance**: Satisfaction\_Score, Last\_Activity, and Support\_Tickets\_Raised were top predictors.
  + **Confusion Matrix**: High true positives (445) and true negatives (549).
* **Logistic Regression**: Lower performance due to non-linear data relationships.

## **Strengths and Weaknesses Summary**

| **Model** | **Strengths** | **Weaknesses** |
| --- | --- | --- |
| **Gradient Boosting** | High accuracy, handles complex patterns | Computationally intensive, risk of overfitting |
| **Random Forest** | Robust, handles non-linearity | Less interpretable, slower than linear models |
| **Logistic Regression** | Simple, interpretable | Limited to linear relationships |
| **Naive Bayes** | Fast, works with small data | Assumes feature independence |

### **Future Improvements**

1. **Feature Engineering**: Create interaction terms (e.g., Subscription\_Length × Monthly\_Spend).
2. **Explainability**: Use SHAP values to interpret Gradient Boosting predictions.
3. **Data Validation**: Investigate near-perfect AUC for potential data leakage that would result in overfitting.

### **Conclusion**

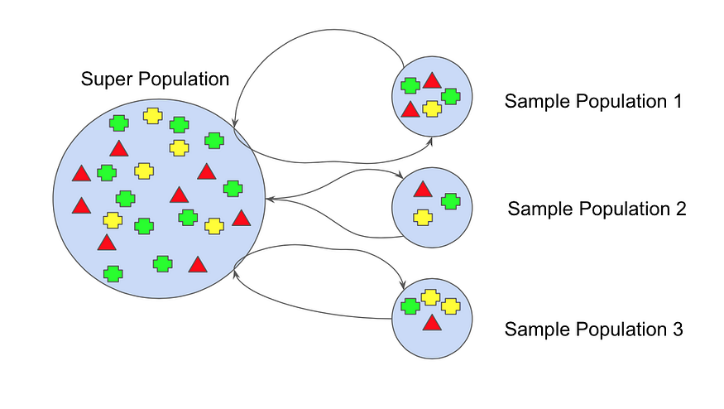
Gradient Boosting appeared as the best model to predict churn customers, achieving 99.4% test accuracy. Its ability to capture complex feature interactions and iterative learning process made it perfect for this task. However, the almost perfect scores warrant further validation to ensure robustness. Machine learning, when applied carefully, offers powerful tools for customer retention strategies, but model selection must balance performance.

## **Differences Between Bootstrap, Bagging, and Boosting**

**Ensemble**

* **Definition**: An ensemble method is responsible for combining multiple individual models (mostly described as "base learners" or "weak learners") to create a stronger model.
* **Purpose**: The main goal of ensemble techniques is to improve the predictive performance and reduce the risk of overfitting by leveraging the strengths of several models.
* **Examples**: Common ensemble techniques include bootstrapping, bagging, and boosting.

### **Bootstrap**



* **Definition**: A statistical resampling technique where multiple datasets are created by randomly sampling **with replacement** from the primary dataset. Each bootstrap sample has the same size as the original data.
* **Purpose**: Estimates the variability of statistics (e.g., mean, variance) or model performance.
* **Key Feature**: Helps measure uncertainty without needing additional data.
* **Example**: Calculating confidence intervals for a model’s accuracy.

### **Bagging (Bootstrap Aggregating)**

A diagram of a algorithm

AI-generated content may be incorrect.

* **Definition**: Ensemble method that trains **multiple base models independently** on different bootstrap samples. Predictions are aggregated (e.g., majority vote for classification, average for regression).
* **Purpose**: Reduces **variance** and prevents overfitting, specifically for high-variance models like deep decision trees.
* **Key Features**:
  + Parallel training of models.
  + Effective for noisy data.
* **Example**: **Random Forest** (bagging + random feature selection).

### **Boosting**

A diagram of a robot and a sequence of a robot

AI-generated content may be incorrect.

* **Definition**: Ensemble method that trains **models sequentially**, where each new model aims on correcting the errors made by previous models. Instances that are misclassified get higher weights in the following iterations.
* **Purpose**: Reduces **bias** and **variance** by combining weak learners (e.g., shallow trees) into a strong learner.
* **Key Features**:
  + Sequential training (models depend on previous ones).
  + Prioritizes hard-to-predict instances.
* **Example**: **AdaBoost**, **Gradient Boosting Machines (GBM)**, **XGBoost**.

### **Comparison Table**

| **Aspect** | **Bootstrap** | **Bagging** | **Boosting** |
| --- | --- | --- | --- |
| **Core Idea** | Resampling with replacement | Combine models trained on bootstrap samples | Sequentially correct errors of prior models |
| **Training Style** | Not an ensemble method | Parallel training of independent models | Sequential training of dependent models |
| **Primary Goal** | Estimate statistical uncertainty | Reduce variance | Reduce bias and variance |
| **Model Dependency** | N/A | Independent models | Dependent models |
| **Handling Overfitting** | N/A | Effective for high-variance models | Prone to overfitting if not regularized |
| **Examples** | Confidence intervals | Random Forest | AdaBoost, GBM, XGBoost |

### **Key Differences**

1. **Resampling vs. Ensemble Methods**:
   * Bootstrap is a **general resampling technique**.
   * Bagging and boosting are **ensemble methods** that use bootstrap aggregation (bagging) or weighting (boosting) to enhance model’s performance.
2. **Parallel vs. Sequential**:
   * Bagging trains models in parallel.
   * Boosting trains models sequentially, learning from mistakes.
3. **Error Focus**:
   * Bagging averages out noise.
   * Boosting corrects errors by highlighting misclassified instances.
4. **Use Cases**:
   * **Bagging**: Ideal for complex models prone to overfitting (e.g., deep trees).
   * **Boosting**: Effective for improving weak learners (e.g., shallow trees).

### **Conclusion**

* **Bootstrap** is foundational for uncertainty estimation.
* **Bagging** reduces variance by averaging diverse models.
* **Boosting** builds strong learners by iteratively refining weak ones.
* Choose **bagging** for noisy data and high-variance models; use **boosting** to improve accuracy on complex patterns.

# References

* 1. www.w3schools.com. (n.d.). *Python Machine Learning*. [online] Available at: <https://www.w3schools.com/python/python_ml_getting_started.asp>.
  2. GeeksforGeeks. (2021). *Machine Learning with Python*. [online] Available at: <https://www.geeksforgeeks.org/machine-learning-with-python/>.
  3. GeeksforGeeks (2024). *Random forest algorithm in machine learning*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>.
  4. GeeksforGeeks (2024). *Understanding Logistic Regression*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/understanding-logistic-regression/>.
  5. GeeksForGeeks (2020). *ML - Gradient Boosting*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/ml-gradient-boosting/>.
  6. Kumar, N. (2019). *Naive Bayes Classifiers - GeeksforGeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/naive-bayes-classifiers/>.
  7. GeeksforGeeks. (2020). *Handling Imbalanced Data for Classification*. [online] Available at: <https://www.geeksforgeeks.org/handling-imbalanced-data-for-classification/>.
  8. Nautiyal, D. (2017). *Underfitting and Overfitting in Machine Learning*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>.
  9. GeeksForGeeks (2017). *Decision Tree Introduction with example*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/decision-tree-introduction-example/>.
  10. GeeksforGeeks. (2023). *A Comprehensive Guide to Ensemble Learning*. [online] Available at: <https://www.geeksforgeeks.org/a-comprehensive-guide-to-ensemble-learning/>.