* What is the difference between static and dynamic variables in Python?
  + **Static Variables**: These are variables that are shared among all instances of a class. They are defined within the class but outside any instance methods.
    - a = 0
  + **Dynamic Variables**: These are instance variables that are unique to each instance of a class. They are defined within the methods of the class.
    - * Var = 0
      * for i in range(1,10): var = i
* Explain the purpose of "pop","popitem","clear() " in a dictionary with suitable examples8?
  + pop: Removes a specified key and returns the corresponding value
  + popitem: Removes and returns the last inserted key-value pair
  + clear: Removes all items from the dictionary
  + my\_dict = {'a': 1, 'b': 2, 'c': 3} item = my\_dict.popitem(), value = my\_dict.pop('b'), my\_dict.clear()
* What do you mean by FrozenSet? Explain it with suitable examples8?
  + A frozenset is an immutable set, meaning you cannot add or remove elements from it after its creation
  + my\_set = frozenset([1, 2, 3, 4])
  + my\_set.add(5) # Raises AttributeError
* Differentiate between mutable and immutable data types in Python and give examples of mutable and immutable data types?
  + **Mutable Data Types**: Can be changed after their creation. Examples include lists, dictionaries, sets. my\_list = [1, 2, 3]
  + **Immutable Data Types**: Cannot be changed after their creation. Examples include strings, tuples, frozensets. my\_tuple = (1, 2, 3)
* What is \_\_init\_\_? Explain with an example?
  + \_\_init\_\_ is a special method in Python that initializes an object's attributes
* What is docstring in Python?Explain with an example?
  + A docstring is a string literal that appears as the first statement in a module, function, class, or method definition. It is used to describe the purpose of the function or class
  + def add(a, b): """This function adds two numbers.""" return a + b
  + print(add.\_\_doc\_\_)
* What are unit tests in Python?
  + Unit tests are a way to test individual units of source code to ensure they are functioning correctly. Python's unittest module is used for writing and running tests
* What is break, continue, and pass in Python?
  + break: Exits the current loop prematurely
    - for i in range(5): if i == 3: break print(i)
  + continue: Skips the rest of the code inside the loop for the current iteration
    - for i in range(5): if i == 3: continue print(i) # Output: 0 1 2 4
  + pass: Does nothing. It is a null statement used as a placeholder
    - for i in range(5): if i == 3: pass print(i) # Output: 0 1 2 3 4
* What is the use of self in Python?
  + self represents the instance of the class. It binds the attributes with the given arguments
* What are global, protected and private attributes in Python?
  + Global Attributes: Accessible from anywhere in the code.
  + Protected Attributes: Indicated by a single underscore (e.g., \_attribute). Suggests that they should not be accessed outside the class.
  + Private Attributes: Indicated by double underscores (e.g., \_\_attribute). Not directly accessible from outside the class
  + class Example:

public\_var = "I am public"

\_protected\_var = "I am protected"

\_\_private\_var = "I am private"

* + example = Example()

print(example.public\_var) # Output: I am public

print(example.\_protected\_var) # Output: I am protected

# print(example.\_\_private\_var) # Raises AttributeError

* What are modules and packages in Python?
  + **Modules**: A file containing Python code, which can define functions, classes, and variables. It can also include runnable code. e.g. app.py
  + **Packages**: A collection of modules in directories that give a package hierarchy. .e.g. import numpy as np
* What are lists and tuples? What is the key difference between the two?
  + **Lists**: Mutable, ordered sequence of elements
  + **Tuples**: Immutable sequences, typically used to store collections of heterogeneous data
* What is an Interpreted language & dynamically typed language? Write 5 differences between them?
  + **Interpreted Language**:
    - Execution: Interpreted languages are executed line by line by an interpreter.
    - Compilation: No separate compilation step; the code is executed directly.
    - Performance: Generally slower due to line-by-line execution.
    - Error Handling: Errors are found at runtime.
    - Portability: Easier to port to different platforms since there's no need for recompilation.
  + **Dynamically Typed Language**:
    - Type Checking: Types are checked at runtime.
    - Variable Declaration: No need to declare variable types explicitly.
    - Flexibility: More flexible, allowing for rapid prototyping and dynamic changes.
    - Error Detection: Type errors are found only during execution.
    - Code Readability: Can sometimes lead to code that's harder to read and debug due to the lack of explicit type declarations
* What are Dict and List comprehensions?
  + List Comprehensions:
    - Syntax: [expression for item in iterable if condition]
  + Dict Comprehensions:
    - Syntax: {key: value for item in iterable if condition}
* What are decorators in Python? Explain it with an example.Write down its use cases?
  + Decorators are a way to modify or extend the behavior of functions or methods
  + def my\_decorator(func):

def wrapper():

print("Something is happening before the function is called.")

func()

print("Something is happening after the function is called.")

return wrapper

@my\_decorator

* + def say\_hello():

print("Hello!")

say\_hello()

Use Cases:

* Logging: To log information when functions are called.
* Authorization: To check if a user has permissions to execute a function.
* Timing: To measure the time a function takes to execute
* How is memory managed in Python?
  + Python uses a combination of **reference counting** and **garbage collection** to manage memory. Objects are automatically freed when their reference count drops to zero. Additionally, Python's garbage collector handles circular references and other memory management tasks.
* What is lambda in Python? Why is it used?
  + A lambda function is a small anonymous function defined with the lambda keyword
  + Used for creating small, throwaway functions without needing to formally define a function
  + add = lambda x, y: x + y
  + print(add(2, 3)) # Output: 5
* Explain split and join functions in Python?
  + split(): Splits a string into a list of substrings based on a delimiter
    - s = "hello world"
    - print(s.split()) # Output: ['hello', 'world']
  + join(): Joins a list of strings into a single string with a specified delimiter
    - words = ['hello', 'world']
    - print(' '.join(words)) # Output: 'hello world'
* What are iterators , iterable & generators in Python?

Iterators:

* Objects that implement the \_\_iter\_\_() and \_\_next\_\_() methods

Iterable:

* Objects that can return an iterator with the \_\_iter\_\_() method

Generators:

* Special iterators created using functions and the yield keyword

def generator():

yield 1

yield 2

yield 3

gen = generator()

print(next(gen)) # Output: 1

* What is the difference between xrange and range in Python?

Python 2:

* range: Returns a list.
* xrange: Returns an iterator, more memory efficient.

Python 3:

* range: Behaves like xrange from Python 2, returning an iterator.
* Pillars of Oops?
  + - * Encapsulation: Bundling data and methods that operate on the data within one unit, like a class.
      * Abstraction: Hiding the complex implementation details and showing only the necessary features.
      * Inheritance: Mechanism for creating a new class from an existing class.
      * Polymorphism: Ability to use a common interface for multiple forms (data types)

class Parent:

pass

class Child(Parent):

pass

print(issubclass(Child, Parent)) # Output: True

* How does inheritance work in python? Explain all types of inheritance with an example?
* Types:
  + Single Inheritance: One parent class.
  + Multiple Inheritance: More than one parent class.
  + Multilevel Inheritance: Inheritance in a chain.
  + Hierarchical Inheritance: One parent class, multiple child classes.
  + Hybrid Inheritance: Combination of two or more types of inheritance

class Parent:pass

class Child(Parent):pass

class GrandChild(Child):pass

* What is encapsulation? Explain it with an example?
  + Restricting access to methods and variables to prevent direct modification
  + class Car:

def \_\_init\_\_(self):

self.\_\_speed = 0

def set\_speed(self, speed):

self.\_\_speed = speed

* + def get\_speed(self):

return self.\_\_speed

car = Car()

car.set\_speed(100)

print(car.get\_speed()) # Output: 100

* What is polymorphism?  Explain it with an example.
  + The ability to use a single interface for different data types
  + class Dog:

def sound(self):

return "Bark"

class Cat:

def sound(self):

return "Meow"

def make\_sound(animal):

print(animal.sound())

dog = Dog()

cat = Cat()

make\_sound(dog) # Output: Bark

make\_sound(cat) # Output: Meow

* What is the difference between Series & Dataframes?

Series:

* A one-dimensional labeled array capable of holding any data type (integers, strings, floating-point numbers, Python objects, etc.).
* Think of it as a single column in a table.
* Has an associated index

DataFrame:

* A two-dimensional labeled data structure with columns of potentially different types.
* Think of it as a table or a collection of Series.
* Contains both rows and columns, each with an index
* Create a database named Travel\_Planner in MySQL, and create a table named bookings in it, which has attributes (user\_id INT, flight\_id INT, hotel\_id INT, activity\_id INT, booking\_date DATE). Fill it with some dummy values. Now, you have to read the content of this table using Pandas as a DataFrame. Show the output.
* What is the difference between loc and iloc?

loc:

* Label-based data selection method.
* Can be used to select rows and columns by labels or a boolean array.
* Syntax: df.loc[rows, columns]

iloc:

* Integer-based (position-based) data selection method.
* Can be used to select rows and columns by integer position.
* Syntax: df.iloc[rows, columns]
* What is the difference between supervised and unsupervised learning?
  + In supervised learning we have a dependent variable for which we need to do analysis and problem statement revolve around it. For e.g. predicting birth wt of the child, load defaulters.
* In unsupervised machine learning we do not have any dependent variable. These kind of algos are used in segmentation tasks like profiling of customer, stores based on the similarity between them.
* Explain the bias-variance tradeoff.
  + Bias and variance tradeoff simple means Overfitting vs Underfitting scenario. In overfitting you model overlearns from training dataset and hence it produces high accuracy from training data but fails to perform similarly on test dataset hence it is called as low bias ad high variance
  + In case underfitting model created can’t understand variation in both train and test data hence results in lower accuracies in both train and test. Hence it is called high bias and high variance.
* What are precision and recall? How are they different from accuracy?
  + Precision is out of all the cases that were predicted positive how many were actually positive cases.
  + Recall is out of all the actual positive cases how much were predicted as positive.
* What is overfitting and how can it be prevented?
  + Definition: In overfitting you model overlearns from training dataset and hence it produces high accuracy from training data but fails to perform similarly on test dataset hence it is called as low bias ad high variance
  + Methods:
    1. L1 regularization
    2. Dropout layers (Neural networks)
    3. Pruning (decision trees)
    4. Cross validations
* Explain the concept of cross-validation.
  + Cross validation is technique where we measure effectiveness of the model over different data sets. For this purpose we divide dataset in k no’s and make sure that each chunk is tested as a test dataset with other forming a combination of train dataset
  + With this method we allow our model to be tested for each combination of our data
* What is the difference between a classification and a regression problem?
  + A classification problem has a dependent variable which is a binary variable that either yes or no. for e.g. whether a customer will default of not or a team will win or loose the match in IPL
  + A regression problem has a dependent variable in the continues form that it values value can range from – ∞ to + ∞. For e.g. birth weight of child, consumption electricity or GRP prediction
* Explain the concept of ensemble learning.
  + Ensemble learning is a technique where we make use of multiple models for prediction of one problem. Here we combine output of each model and get the accuracy and hence it is called as ensemble learning.

* What is gradient descent and how does it work?
  + Gradient descent is an optimization algorithm used to minimize a function by iteratively moving towards the minimum value of that function. It's widely used in machine learning and deep learning for optimizing the parameters of models, such as weights in neural networks.
  + **How Gradient Descent Works**
  + **Objective Function**: The process begins with defining an objective function (or loss function) that measures how well the model's predictions match the actual target values. The goal is to minimize this function.
  + **Initial Parameters**: Start with initial guesses for the parameters (weights) of the model. These can be random values or zeros.
  + **Compute the Gradient**: Calculate the gradient of the objective function with respect to each parameter. The gradient is a vector of partial derivatives, indicating the direction and rate of change of the function with respect to each parameter.
  + **Update the Parameters**: Adjust the parameters in the opposite direction of the gradient. This step is determined by the learning rate, a hyperparameter that controls how much to change the parameters.
  + **Iterate**: Repeat the process of computing the gradient and updating the parameters until convergence, which is when the changes in the objective function become negligible, or a predetermined number of iterations is reached
* Describe the difference between batch gradient descent and stochastic gradient descent.
  + Stochastic gradient descent solves problem associated with gradient descent like high computational resources. It does by passing one record per epoch and hence RAM requirement is slowest in this case. But the convergence is also very slow.
  + Batch stochastic gradient descent is upgraded version of stochastic gradient descent. What it does is it passes fix no of records per batch, and it passes the batches in one epoch. Hence it converges faster than stochastic gradient descent.
* What is the curse of dimensionality in machine learning?
  + Curse of dimensionality is associated with the higher number of variables in the dataset. i.e. when we have higher no of variables in the data our model is not capable of understanding variations in all of them and hence it results in lower accuracy
* Explain the difference between L1 and L2 regularization.
  + L1 is Lasso regression and L2 is Ridge regression. These are the techniques which are used to deal with the overfitting/ underfitting scenario in the modelling.
  + They are extended version of basic regression models and here we use Lambda parameter for penalizing the beta coefficients and hence slope value changes
  + In L1 slope tends to be zero for some variables and hence it also act as variable/dimension reduction technique. In L2 slops shrunk but they do not become zero
* What is a confusion matrix and how is it used?
  + Confusion matrix is a model evaluation technique, and it is used in case of classification models
  + It’s a 2\*2 matrix where we plot the Actual values for events and non-events vs Predicted values for events and non-events
  + With the help of confusion matrix we calculate important model evaluations parameters like TPR, FPR, Recall, Precision and F1 score
* Define AUC-ROC curve.
  + It’s a curve which we plot for TPR against FPR.
  + The curve gives information like how good the model vs is the random prediction of Yes or No
  + The curve is plotted for values of TPR vs FPR for different cutoff values
* Explain the k-nearest neighbors’ algorithm.
  + KNN is used for both classification and regression problems. It focuses on selecting value of K. lets suppose k =5 then algorithm will try to capture 5 nearest points to each point and then classify the point based on majority. It means if out of 5, 4 points are events then it will classify the point as events around which the 5 points distance is calculated
  + In case of KNN regression the mean of dependent variable is calculated and then point is given the mean based on closeness
* Explain the basic concept of a Support Vector Machine (SVM).
  + A Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification tasks, but it can also be adapted for regression. The basic concept of an SVM is to find the optimal hyperplane that separates data points of different classes in a feature space.
  + **Key Concepts of SVM**
  + **Hyperplane**: In an nnn-dimensional space, a hyperplane is an n−1n-1n−1-dimensional flat affine subspace that separates the data points into different classes. In two dimensions, it's a line; in three dimensions, it's a plane.
  + **Margin**: The margin is the distance between the hyperplane and the closest data points from each class. SVM aims to maximize this margin to ensure that the hyperplane is optimally positioned between classes, providing a buffer against misclassification.
  + **Support Vectors**: These are the data points that lie closest to the hyperplane. They are critical in defining the position and orientation of the hyperplane. The model only depends on these support vectors, not the entire dataset.
  + **Kernel Trick**: SVM can efficiently perform a non-linear classification using what is called the kernel trick, which implicitly maps the input features into high-dimensional feature spaces without explicitly computing the coordinates in that space. This allows the SVM to handle complex, non-linear boundaries.
  + **Linear vs. Non-linear SVM**:
  + **Linear SVM**: Directly finds a linear hyperplane to separate data.
  + **Non-linear SVM**: Uses kernel functions (like polynomial, radial basis function) to transform data into higher dimensions where a linear hyperplane can separate the classes.
  + **Objective**
  + The objective of an SVM is to find the hyperplane that maximizes the margin between the classes. This results in a robust model that can generalize well to new data.
  + Overall, SVMs are powerful tools for classification tasks, especially when the classes are well-separated and when the feature space is not too high-dimensional. They are effective in various applications, such as image classification, text categorization, and bioinformatics
* How does the kernel trick work in SVM?
  + The kernel trick in Support Vector Machines (SVM) allows the algorithm to handle non-linear classification problems by transforming the data into a higher-dimensional space without explicitly calculating the new coordinates. Instead of mapping data points using a function, the kernel trick uses a kernel function to compute the dot product between data points in this high-dimensional space. This makes it computationally efficient to find a linear hyperplane in the transformed space, resulting in a non-linear decision boundary in the original space. Common kernel functions include the polynomial, radial basis function (RBF), and sigmoid kernels
* What are the different types of kernels used in SVM and when would you use each?
  + Support Vector Machines (SVM) use various kernel functions to handle different types of data and decision boundaries. Here are some common types of kernels and their typical use cases:
  + **Linear Kernel**:
    - **Use Case**: Suitable for linearly separable data. It is the simplest kernel and is often used when the number of features is very large relative to the number of samples, as it can help avoid overfitting. It's computationally efficient and is often a good starting point for simple datasets.
  + **Polynomial Kernel**:
    - **Use Case**: Useful for problems where the relationship between class labels and attributes is polynomial. The degree ddd allows you to control the flexibility of the decision boundary. It’s appropriate for data with complex relationships that are not captured by a linear boundary.
  + **Radial Basis Function (RBF) Kernel / Gaussian Kernel**:
    - **Use Case**: The most popular and widely used kernel, suitable for non-linear data. It’s highly flexible and works well in most scenarios. The RBF kernel can map the input space into an infinite-dimensional space, allowing for highly complex decision boundaries.
  + **Sigmoid Kernel**:
    - **Use Case**: This kernel behaves like a neural network's activation function and can be used for certain types of neural network-like decision boundaries. It’s less commonly used compared to RBF and polynomial kernels.
  + **Choosing the Right Kernel**
    - **Linear Kernel**: Use when data is linearly separable or when you want a simpler model with a large number of features.
    - **Polynomial Kernel**: Use for polynomial relationships in the data and when you need more flexibility than a linear kernel.
    - **RBF Kernel**: Use as a default choice for non-linear data, especially when you want the flexibility to capture complex patterns.
    - **Sigmoid Kernel**: Use when modeling scenarios similar to neural networks, though other kernels might often be more effective.
    - Experimenting with different kernels and using techniques like cross-validation can help determine the best kernel for your specific problem
* What is the hyperplane in SVM and how is it determined?
  + The hyperplane in SVM is the decision boundary that separates different classes in the feature space. It is determined by solving an optimization problem to maximize the margin between the hyperplane and the closest data points from each class, ensuring the best separation between classes
* What are the pros and cons of using a Support Vector Machine (SVM)?
  + **Pros of SVM:**
    - **Effective in High-Dimensional Spaces**: Performs well with large feature sets.
    - **Robust to Overfitting**: Regularization helps prevent overfitting.
    - **Versatile**: Handles non-linear data using kernel functions.
    - **Optimal Margin Classifier**: Maximizes the margin between classes, improving generalization.
  + **Cons of SVM:**
    - **Computationally Intensive**: Training can be slow and memory-consuming, especially with large datasets.
    - **Parameter Tuning**: Requires careful selection of kernel and hyperparameters.
    - **Less Scalable**: May struggle with very large datasets.
    - **Complexity**: Decision boundaries can be hard to interpret.
    - 4o mini
* Explain the difference between a hard margin and a soft margin SVM.
  + **Hard Margin SVM**
    - **Definition**: A hard margin SVM seeks to find a hyperplane that perfectly separates all the data points of different classes with no misclassifications.
    - **Assumption**: Assumes that the data is linearly separable. If the data cannot be separated without errors, a hard margin SVM will fail to find a solution.
    - **Objective**: Maximize the margin between the classes while ensuring that all data points are correctly classified.
    - **Use Case**: Suitable for clean datasets where classes are distinctly separable without any noise or overlap.
  + **Soft Margin SVM**
    - **Definition**: A soft margin SVM allows for some misclassifications by introducing slack variables that permit some data points to be within the margin or even on the wrong side of the hyperplane.
    - **Assumption**: Can handle cases where the data is not perfectly separable. It balances between maximizing the margin and minimizing classification errors.
    - **Objective**: Maximize the margin while minimizing the classification error, controlled by a regularization parameter CCC. The parameter CCC determines the trade-off between having a larger margin and allowing some misclassifications.
    - **Use Case**: Suitable for real-world data where classes may overlap or contain noise
* Describe the process of constructing a decision tree.
  + Decision trees are nothing but the simple nested if else in programming language. The variable is selected based on information calculation. Higher the information gain that variable will be selected first to split. Then Gini impurity is calculated for then reach node and based on highest impurity between the nodes the further splitting happens
  + Once we have a pure node the splitting will not happen. Based pruning criteria given the splitting happens till that level
* Describe the working principle of a decision tree.
  + Decision trees are nothing but the simple nested if else in programming language. The variable is selected based on information calculation. Higher the information gain that variable will be selected first to split. Then Gini impurity is calculated for then reach node and based on highest impurity between the nodes the further splitting happens
  + Once we have a pure node the splitting will not happen. Based pruning criteria given the splitting happens till that level
* What is information gain and how is it used in decision trees?
  + Information gain is a criteria that is used to select the variable to split. It helps in variable selection for splitting in DT algorithm
* Explain Gini impurity and its role in decision trees.
  + After we split the variable and we have base root along with its root nodes we then calculate the Gini score for each node and select the most impure node to split the further
  + The node with higher Gini score will be most impure node
* What are the advantages and disadvantages of decision trees?
  + Adv:
    - It is not affected by missing value
    - It is not affected by imbalance dataset
    - It is not affected by outliers in the dataset
  + Dis Adv:
    - They tend to overfit
* How do random forests improve upon decision trees?
  + By averaging the results of many decision trees, random forests reduce the risk of overfitting. Each tree is trained on different subset of the data and hence it results in more generalized model
  + We get more accuracy and better model as we are combing the results of multiple model hence it results into more stable model
* How does a random forest algorithm work?
  + Random forest is an ensemble technique. It makes use of multiple DT models for its prediction
  + In RF we have multiple DT who train the model in parallel manner. Data is provided to each model and each model gives its prediction and majority is taken when it comes to deciding the final prediction
* What is bootstrapping in the context of random forests?
  + Bootstrapping is a technique used in the context of random forests to enhance model accuracy and robustness. In random forests, bootstrapping helps create multiple diverse decision trees by using different subsets of the training data.
  + **How Bootstrapping Works in Random Forests**
    - **Sampling with Replacement**: Bootstrapping involves creating multiple subsets of the original training data by sampling with replacement. This means some data points may appear multiple times in a subset, while others may not appear at all.
    - **Training Multiple Trees**: Each subset (bootstrap sample) is used to train a different decision tree. Since each tree is trained on a slightly different subset of the data, the trees are diverse and capture different aspects of the data.
    - **Aggregation**: Once all trees are trained, their predictions are aggregated (e.g., by voting for classification or averaging for regression) to make the final prediction. This aggregation helps improve the overall model accuracy and reduces overfitting.
    - **Benefits of Bootstrapping in Random Forests**
    - **Reduces Variance**: By combining multiple trees, each trained on different subsets of the data, bootstrapping helps reduce the variance of the model's predictions.
    - **Improves Robustness**: The diversity among the trees makes the model more robust to noise and overfitting.
    - **Provides an Estimate of Prediction Error**: Bootstrapping can be used to estimate the prediction error of the random forest through techniques like out-of-bag (OOB) error estimation, where data not included in a bootstrap sample (OOB data) is used to validate the model
* Explain the concept of feature importance in random forests.
  + Feature importance in random forests refers to a measure of how valuable each feature is in predicting the target variable. It helps to understand which features contribute most to the model’s performance and can guide feature selection or interpretation.
  + **How Feature Importance is Calculated**
  + **Tree-Based Method**:
    - **Impurity Reduction**: In random forests, feature importance is often calculated based on how much each feature decreases the impurity (such as Gini impurity or entropy) in decision trees. A feature is considered important if it significantly reduces impurity in nodes where it is used.
    - **Averaging Over Trees**: The importance scores are averaged across all the decision trees in the forest. Each tree contributes to the score based on how much the feature reduces impurity when it is used for splitting.
  + **Permutation Method**:
    - **Permutation Feature Importance**: Another approach involves shuffling the values of a feature and measuring the decrease in the model’s performance (e.g., accuracy or F1 score). The greater the decrease, the more important the feature is. This method is useful for assessing the importance in the context of the entire model’s performance.
  + **Interpretation**
    - **Higher Scores**: Features with higher importance scores are considered more influential in the model’s predictions. These features are likely to provide significant information for distinguishing between classes or predicting the target variable.
    - **Lower Scores**: Features with lower importance scores contribute less to the model’s predictive power and may be candidates for removal if feature reduction is desired.
    - **Use Cases**
    - **Feature Selection**: Identifying important features helps in selecting the most relevant variables, potentially improving model performance and reducing complexity.
    - **Model Interpretation**: Understanding feature importance provides insights into how the model makes predictions and which factors are driving its decisions.
  + **Summary**
    - Feature importance in random forests measures how much each feature contributes to the model’s accuracy by evaluating the reduction in impurity or performance when the feature is used. It helps in feature selection and provides insights into model behavior
* What are the key hyperparameters of a random forest and how do they affect the model?
  + Random forests have several key hyperparameters that influence the model's performance and behavior. Here’s a summary of the most important ones and how they affect the model:
  + **1. Number of Trees (n\_estimators)**
    - **Description**: The number of decision trees in the forest.
    - **Effect**: Increasing the number of trees generally improves the model's performance by reducing variance and overfitting, as it averages the predictions from more trees. However, it also increases computational cost and memory usage.
  + **2. Maximum Depth (max\_depth)**
    - **Description**: The maximum number of levels (depth) of each decision tree.
    - **Effect**: Controlling the depth helps prevent overfitting. Shallow trees may underfit the data, while deeper trees can capture more complexity but may overfit if not properly regulated.
  + **3. Minimum Samples Split (min\_samples\_split)**
    - **Description**: The minimum number of samples required to split an internal node.
    - **Effect**: Higher values prevent the model from learning overly specific patterns by ensuring that nodes are split only when there are enough samples. This can reduce overfitting.
  + **4. Minimum Samples Leaf (min\_samples\_leaf)**
    - **Description**: The minimum number of samples required to be at a leaf node.
    - **Effect**: Setting this parameter ensures that leaf nodes contain a sufficient number of samples, which can help prevent overfitting by smoothing the model.
  + **5. Maximum Features (max\_features)**
    - **Description**: The number of features to consider when looking for the best split.
    - **Effect**: Limits the number of features used for each split, introducing more randomness and improving the diversity of trees. It helps reduce correlation between trees and improves generalization.
  + **6. Bootstrap (bootstrap)**
    - **Description**: Whether bootstrap samples are used when building trees.
    - **Effect**: When set to True, each tree is trained on a random sample of the data with replacement (bootstrapping). Setting it to False uses the entire dataset for each tree, which can lead to more correlated trees and reduced diversity.
  + **7. Criterion (criterion)**
    - **Description**: The function used to measure the quality of a split (e.g., "gini" for Gini impurity, "entropy" for information gain).
    - **Effect**: Determines how the quality of a split is evaluated. Different criteria can lead to different tree structures and affect model performance.
  + **8. Random State (random\_state)**
    - **Description**: Seed used by the random number generator.
    - **Effect**: Ensures reproducibility of the results by initializing the random number generator to a fixed state. This makes the results consistent across different runs.
  + **Summary**
    - **n\_estimators**: More trees generally improve performance but increase computation.
    - **max\_depth**: Controls tree complexity; deeper trees capture more details but risk overfitting.
    - **min\_samples\_split** and **min\_samples\_leaf**: Prevent overfitting by requiring a minimum number of samples to split nodes or be at leaf nodes.
    - **max\_features**: Limits the number of features considered for splits, adding randomness and diversity.
    - **bootstrap**: Controls whether to use bootstrapped samples for tree training.
    - **criterion**: Defines how splits are evaluated.
    - **random\_state**: Ensures reproducibility of results.
* Describe the logistic regression model and its assumptions.
  + **Logistic Regression** is a model used for binary classification, predicting the probability of an outcome using the logistic function.
  + **Key Points:**
    - **Model**: Estimates the probability of a binary outcome (0 or 1) based on predictor variables.
    - **Logistic Function**: Transforms the linear combination of predictors into a probability between 0 and 1.
    - **Prediction**: Uses a threshold (typically 0.5) to classify the outcome.
    - **Assumptions:**
    - **Linearity in Logits**: The log odds of the outcome is a linear function of the predictors.
    - **Independence**: Observations are independent.
    - **No Multicollinearity**: Predictors should not be highly correlated.
    - **Large Sample Size**: Requires a sufficient number of samples.
    - **Binary Outcome**: Used for binary outcomes; multinomial logistic regression is used for multiple classes.
* How does logistic regression handle binary classification problems?
  + Logistic regression handles binary classification problems by modeling the probability that a given input belongs to a particular class. Here’s how it works:
  + **Probability Estimation**:
  + **Logistic Function**: It uses the logistic (or sigmoid) function to convert the linear combination of input features into a probability.
  + **Decision Boundary**:
  + **Thresholding**: The model uses a threshold (usually 0.5) to decide the class label. If the predicted probability is greater than the threshold, the output is classified as 1; otherwise, it is classified as 0.
  + **Model Training**:
    - **Maximum Likelihood Estimation (MLE)**: Logistic regression estimates the coefficients (b0b\_0b0​ and b1b\_1b1​) by maximizing the likelihood of the observed data. This is done by adjusting the coefficients to best fit the data.
    - **Binary Output**:
    - The output is a probability value between 0 and 1, representing the likelihood of the positive class. The binary classification is determined by comparing this probability to the threshold.
    - **Summary**
    - Logistic regression handles binary classification by modeling the probability of an outcome using the logistic function, setting a threshold to classify the result, and training the model using maximum likelihood estimation to find the best-fitting coefficients
* Explain the concept of the cost function in logistic regression.
  + In logistic regression, the **cost function** (also known as the loss function or objective function) quantifies how well the model’s predictions match the actual outcomes. It is used to measure the model's performance and guide the optimization process during training.
  + **Concept of the Cost Function**
  + **Definition**:
  + The cost function in logistic regression calculates the error between the predicted probabilities and the actual binary outcomes. The goal is to minimize this error to improve the model's accuracy.
  + **Binary Cross-Entropy Loss**:
  + The most commonly used cost function in logistic regression is the **binary cross-entropy loss** (or log loss).
  + **Interpretation**:
    - **Penalty for Incorrect Predictions**: The cost function penalizes incorrect predictions more heavily. If the predicted probability is close to 1 when the actual label is 0, or close to 0 when the actual label is 1, the cost will be high.
    - **Log Loss**: The log function in the cost function ensures that the penalty increases logarithmically for incorrect predictions. It provides a smoother gradient, which is useful for gradient-based optimization algorithms.
    - **Optimization**:
    - **Minimization**: The objective during training is to find the model parameters (coefficients) that minimize the cost function. This is typically achieved using optimization algorithms like gradient descent.
  + **Summary**
    - The cost function in logistic regression, specifically the binary cross-entropy loss, measures the difference between predicted probabilities and actual binary outcomes. It penalizes incorrect predictions and guides the optimization process to adjust model parameters for better accuracy
* How can logistic regression be extended to handle multiclass classification?
  + To handle multiclass classification problems with logistic regression, the model is extended from binary to multiple classes. This is achieved using one of two common approaches:
  + **1. One-vs-Rest (OvR) or One-vs-All (OvA)**
  + **Concept**:
  + **Separate Binary Classifiers**: This approach involves training a separate binary classifier for each class. Each classifier distinguishes between one class (the "positive" class) and all other classes (the "negative" class).
  + **How It Works**:
  + **Training**: For a dataset with KKK classes, KKK separate logistic regression models are trained. Each model outputs the probability of its corresponding class versus all other classes.
  + **Prediction**: For a new instance, each classifier produces a probability score. The class with the highest probability across all classifiers is selected as the final prediction.
  + **Example**:
  + For a classification problem with three classes (A, B, and C), three models are trained:
  + Model 1: Class A vs. Class B & C
  + Model 2: Class B vs. Class A & C
  + Model 3: Class C vs. Class A & B
  + **2. Softmax Regression (Multinomial Logistic Regression)**
  + **Concept**:
  + **Single Model for All Classes**: This approach generalizes logistic regression to handle multiple classes by modeling the probability of each class directly.
  + **How It Works**:
    - **Training**: The model is trained to maximize the likelihood of the observed classes across all KKK classes simultaneously. This involves minimizing the multinomial cross-entropy loss.
    - **Prediction**: For a new instance, the softmax function produces probabilities for each class. The class with the highest probability is chosen as the final prediction.
    - **Example**:
    - For a classification problem with three classes (A, B, and C), the model outputs three probabilities, one for each class. The class with the highest probability is selected.
  + **Summary**
    - **One-vs-Rest (OvR)**: Trains multiple binary classifiers, each for a different class versus all others. Class with the highest score is chosen.
    - **Softmax Regression (Multinomial Logistic Regression)**: Uses a single model with the softmax function to output probabilities for all classes simultaneously. The class with the highest probability is selected.
    - Both approaches extend logistic regression to handle multiclass problems, with Softmax Regression being a more direct generalization for multiple classes
* What is the difference between L1 and L2 regularization in logistic regression?
  + **L1 Regularization** and **L2 Regularization** are techniques used to prevent overfitting in logistic regression by adding a penalty to the cost function based on the magnitude of the model's coefficients. Here’s a brief comparison:
  + **L1 Regularization (Lasso)**
    - **Penalty**: Adds the sum of the absolute values of the coefficients to the cost function. Penalty=λ∑j∣bj∣\text{Penalty} = \lambda \sum\_{j} |b\_j|Penalty=λj∑​∣bj​∣
    - **Effect**: Can drive some coefficients to exactly zero, effectively performing feature selection.
    - **Usage**: Useful for sparse models where some features are irrelevant or should be excluded.
  + **L2 Regularization (Ridge)**
    - **Penalty**: Adds the sum of the squared values of the coefficients to the cost function. Penalty=λ∑jbj2\text{Penalty} = \lambda \sum\_{j} b\_j^2Penalty=λj∑​bj2​
    - **Effect**: Shrinks coefficients towards zero but usually does not make them exactly zero, leading to a more balanced model.
    - **Usage**: Useful for handling multicollinearity and keeping all features but with smaller weights.
    - **Summary**
    - **L1 Regularization**: Adds the absolute values of coefficients; can zero out some coefficients (sparse model).
    - **L2 Regularization**: Adds the squared values of coefficients; shrinks all coefficients towards zero (smooth model).
* What is XGBoost and how does it differ from other boosting algorithms?
  + **XGBoost (Extreme Gradient Boosting)** is an advanced implementation of gradient boosting that is known for its high performance and scalability. Here's a brief overview and its differences from other boosting algorithms:
  + **XGBoost**
    - **Algorithm**: An optimized gradient boosting algorithm that builds an ensemble of decision trees in a sequential manner, where each tree corrects the errors of the previous ones.
    - **Key Features**:
    - **Regularization**: Incorporates L1 (Lasso) and L2 (Ridge) regularization to prevent overfitting.
    - **Speed**: Utilizes parallel processing and efficient data handling to speed up training.
    - **Handling Missing Data**: Can automatically handle missing values in the dataset.
    - **Tree Pruning**: Employs a more sophisticated tree pruning method compared to traditional boosting algorithms.
  + **Differences from Other Boosting Algorithms**
  + **Gradient Boosting (GBM)**:
    - **XGBoost**: Uses regularization and efficient computation for faster training and better generalization.
    - **GBM**: Traditional implementation may be slower and less regularized, leading to potential overfitting.
  + **AdaBoost**:
    - **XGBoost**: Builds trees sequentially with gradient optimization and regularization.
    - **AdaBoost**: Focuses on adjusting weights of misclassified instances and combines weak classifiers into a strong one.
  + **LightGBM**:
    - **XGBoost**: Uses a depth-first approach for tree construction.
    - **LightGBM**: Uses a histogram-based approach and is optimized for speed and memory efficiency.
  + **CatBoost**:
    - **XGBoost**: Requires manual handling of categorical features.
    - **CatBoost**: Automatically handles categorical features and uses a different approach for boosting.
  + **Summary**
    - XGBoost is a high-performance gradient boosting algorithm with features like regularization, speed optimization, and advanced tree pruning. It differs from other boosting algorithms by offering improved efficiency and regularization techniques, making it suitable for large datasets and complex models.
* Explain the concept of boosting in the context of ensemble learning.
  + Boosting models are different compared to bagging models. In boosting models multiple DT models are used and they are arranged in a series manner
  + It means that the output of each model act as input to the next model resulting into more fine training of models and achieving better accuracy
  + In booting we are always comparing a base model and try to reduce its error in the current iteration. Then that model is push to next model
* How does XGBoost handle missing values?
  + XGBoost can handle missing values internally. It does this by automatically learning the best imputation strategy based on the data. When XGBoost encounters a missing value, it tries both left and right branches of a split and learns which direction is more appropriate. During training, it assigns the optimal direction to handle missing values. This process allows XGBoost to handle missing data without requiring explicit imputation beforehand
* What are the key hyperparameters in XGBoost and how do they affect model performance?

 n\_estimators:

* Number of trees (boosting rounds).
* Higher values can lead to better performance but also increase training time and risk of overfitting.

 learning\_rate (or eta):

* Controls the contribution of each tree.
* Lower values require more trees but can improve model performance by making training more robust to overfitting.

 max\_depth:

* Maximum depth of a tree.
* Higher values can capture more complex patterns but also increase risk of overfitting.

 min\_child\_weight:

* Minimum sum of instance weight (hessian) needed in a child.
* Higher values prevent overfitting by making the model more conservative.

 subsample:

* Fraction of samples used for building each tree.
* Values less than 1.0 prevent overfitting by adding randomness, but too low values can lead to underfitting.

 colsample\_bytree:

* Fraction of features used for building each tree.
* Like subsample, it helps prevent overfitting by adding feature randomness.

 gamma:

* Minimum loss reduction required to make a split.
* Higher values make the algorithm more conservative and prevent overfitting.

 lambda (L2 regularization term):

* L2 regularization on leaf weights.
* Helps prevent overfitting by penalizing large leaf weights.
* alpha (L1 regularization term):
* L1 regularization on leaf weights.
* Similar to lambda, but it can lead to sparsity in leaf weights
* Describe the process of gradient boosting in XGBoost.
  + Initialize the model with a constant value, typically the mean of the target values for regression or log-odds for classification.
  + Iterate over the following steps for a specified number of boosting rounds:
  + Compute the residuals or pseudo-residuals (negative gradients of the loss function) based on the current model predictions.
  + Fit a new tree (weak learner) to these residuals.
  + Update the model by adding the new tree, scaled by the learning rate.
  + Update the weights of the data points to focus more on poorly predicted points (optional, for adaptive boosting)
* What are the advantages and disadvantages of using XGBoost?

Advantages:

1. Performance: XGBoost is known for its high performance and ability to handle large datasets efficiently.
2. Flexibility: It supports various objective functions and evaluation metrics, making it suitable for different types of tasks.
3. Handling Missing Values: XGBoost can automatically handle missing data without requiring explicit imputation.
4. Regularization: Built-in L1 and L2 regularization help prevent overfitting.
5. Parallel Processing: XGBoost can utilize multiple cores for parallel processing, speeding up training.
6. Early Stopping: It supports early stopping based on a validation set, preventing overfitting.

Disadvantages:

1. Complexity: XGBoost has many hyperparameters that can be challenging to tune.
2. Memory Usage: It can be memory-intensive, especially with large datasets.
3. Training Time: While efficient, training can still be time-consuming for very large datasets or complex models.
4. Overfitting: Like other powerful models, XGBoost can overfit if not properly regularized and if hyperparameters are not tuned correctly.
5. Interpretability: Models can be difficult to interpret, especially with many trees and deep structures

* What do you mean by Measure of Central Tendency and Measures of Dispersion .How it can be calculated.

Measures of Central Tendency describe the center of a dataset:

* Mean: Average of all values.
* Median: Middle value when sorted.
* Mode: Most frequently occurring value.

Measures of Dispersion describe the spread of a dataset:

* Range: Difference between the highest and lowest values.
* Variance: Average squared deviation from the mean.
* Standard Deviation: Square root of variance, showing spread in the same units as the data.
* Interquartile Range (IQR): Range within which the middle 50% of values lie (Q3 - Q1).
* What do you mean by skewness.Explain its types.Use graph to show.

Skewness measures the asymmetry of a probability distribution about its mean. It indicates whether the data are skewed to the left (negative skewness) or to the right (positive skewness). In simple terms, it tells you if the data distribution is lopsided.

Types of Skewness

* Positive Skewness (Right Skewness):
  + Description: The right tail of the distribution is longer or fatter than the left tail. Most data points are concentrated on the left side, with a few larger values extending the tail to the right.
  + Example: Income distribution where most people earn below average, but a few high earners pull the average up.
* Negative Skewness (Left Skewness):
  + Description: The left tail of the distribution is longer or fatter than the right tail. Most data points are concentrated on the right side, with a few smaller values extending the tail to the left.
  + Example: Age at retirement where most people retire around the average age, but some retire early.
* Zero Skewness (Symmetric Distribution):
  + Description: The distribution is symmetric around the mean. The left and right tails are mirror images of each other.
  + Example: A normal distribution.
* Explain PROBABILITY MASS FUNCTION (PMF) and PROBABILITY DENSITY FUNCTION (PDF). and what is the difference between them?

Probability Mass Function (PMF):

* Used for: Discrete random variables.
* Definition: Gives the probability of each specific value.
* Example: Rolling a die—probability of each face (1/6).

Probability Density Function (PDF):

* Used for: Continuous random variables.
* Definition: Describes the density of probability over a range; the area under the curve gives the probability.
* Example: Height of individuals—bell-shaped curve in a normal distribution.

Difference:

* PMF provides probabilities for discrete values; PDF provides densities for continuous values. For PDFs, the probability is found by integrating over an interval
* What is correlation. Explain its type in details. what are the methods of determining correlation

Correlation measures the relationship between two variables.

Types of Correlation

1. Positive Correlation: Both variables increase together (e.g., height and weight).
2. Negative Correlation: One variable increases while the other decreases (e.g., exercise and body fat).
3. Zero Correlation: No relationship between variables (e.g., shoe size and intelligence).

Methods of Determining Correlation

1. Pearson Correlation: Measures linear relationships between continuous variables (range: -1 to 1).
2. Spearman's Rank Correlation: Measures monotonic relationships (ordinal or non-linear, range: -1 to 1).
3. Kendall's Tau: Measures ordinal associations, useful for smaller samples.
4. Point-Biserial Correlation: Measures the relationship between a continuous variable and a binary categorical variable

* Discuss the 4 differences between correlation and regression.

 Purpose:

* Correlation: Measures the strength and direction of a linear relationship between two variables.
* Regression: Models the relationship between a dependent variable and one or more independent variables to predict the dependent variable.

 Output:

* Correlation: Produces a correlation coefficient (r) that ranges from -1 to 1.
* Regression: Produces a regression equation that can be used to predict the dependent variable.

 Directionality:

* Correlation: Does not imply causation or directionality; it only measures association.
* Regression: Implies a direction of the relationship, indicating how changes in the independent variable affect the dependent variable.

 Equation:

* Correlation: No specific equation, just a numerical measure of association.
* Regression: Provides a specific equation (e.g., y=mx+cy = mx + cy=mx+c) to describe the relationship.
* Find the most likely price at Delhi corresponding to the price of Rs. 70 at Agra from the following data:

Coefficient of correlation between the prices of the two places +0.8.

* 92
* In a partially destroyed laboratory record of an analysis of correlation data, the following results only are legible: Variance of x = 9, Regression equations are: (i) 8x−10y = −66; (ii) 40x − 18y = 214. What are the mean values of x and y, the coefficient of correlation between x and y, the σ of y.
  + Mean values: xˉ=13\bar{x} = 13xˉ=13, yˉ=17\bar{y} = 17yˉ​=17
  + Coefficient of correlation: r≈0.84r \approx 0.84r≈0.84
  + Standard deviation of yyy: σy≈1.35\sigma\_y \approx 1.35σy​≈1.35
* What is Normal Distribution? What are the four Assumptions of Normal Distribution? Explain in detail.

Definition:

* A normal distribution is a symmetric, bell-shaped distribution where most values cluster around the mean.

Assumptions:

1. Symmetry: The distribution is symmetric around the mean.
2. Mean, Median, Mode: They are all equal.
3. Asymptotic: The tails approach the x-axis but never touch it.
4. 68-95-99.7 Rule: Approximately 68% of data falls within ±1 SD, 95% within ±2 SD, and 99.7% within ±3 SD of the mean

* Which of the following options are correct about Normal Distribution Curve.
* Within a range 0.6745 of σ on both sides the middle 50% of the observations occur i,e. mean ±0.6745σ covers 50% area 25% on each side.
* Mean ±1S.D. (i,e.μ ± 1σ) covers 68.268% area, 34.134 % area lies on either side of the mean.
* Mean ±2S.D. (i,e. μ ± 2σ) covers 95.45% area, 47.725% area lies on either side of the mean.
* Mean ±3 S.D. (i,e. μ ±3σ) covers 99.73% area, 49.856% area lies on the either side of the mean.
* Only 0.27% area is outside the range μ ±3σ.
* All the options provided are correct
* The mean of a distribution is 60 with a standard deviation of 10. Assuming that the distribution is normal,

what percentage of items be (i) between 60 and 72, (ii) between 50 and 60, (iii) beyond 72 and (iv) between 70 and 80?

* Between 60 and 72:
* Probability = 0.3849 (38.49%)
* Between 50 and 60:
* Probability = 0.3413 (34.13%)
* Beyond 72:
* Probability for Z > 1.2 = 0.1151 (11.51%)
* Between 70 and 80:
* Probability between Z = 1 and Z = 2 = 0.1359 (13.59%)
* 15000 students sat for an examination. The mean marks was 49 and the distribution of marks had a standard deviation of 6. Assuming that the marks were normally distributed what proportion of students scored (a) more than 55 marks, (b) more than 70 marks
  + More than 55 marks:
    - Probability = 0.1587 (15.87%)
  + More than 70 marks:
    - Probability = 0.0002 (0.02%)
* If the height of 500 students are normally distributed with mean 65 inch and standard deviation 5 inch.

How many students have height : a) greater than 70 inch. b) between 60 and 70 inch.

 Greater than 70 inches:

* Probability = 0.1587 (15.87%)

 Between 60 and 70 inches:

* Probability between Z = -1 and Z = 1 = 0.6826 (68.26%)
* What is the statistical hypothesis? Explain the errors in hypothesis testing.b)Explain the Sample. What are Large Samples & Small Samples?

Statistical Hypothesis:

* A hypothesis about a population parameter that can be tested using statistical methods.

Errors in Hypothesis Testing:

1. Type I Error: Rejecting a true null hypothesis (False Positive).
2. Type II Error: Failing to reject a false null hypothesis (False Negative).

Sample:

* Large Sample: Typically n≥30n \geq 30n≥30, results are more stable.
* Small Sample: Typically n<30n < 30n<30, results may be less reliable
* A random sample of size 25 from a population gives the sample standard derivation to be 9.0. Test the hypothesis that the population standard derivation is 10.5.Hint(Use chi-square distribution).
  + 17.6
* 100 students of a PW IOI obtained the following grades in Data Science paper :

Grade :[A, B, C, D, E]

Total Frequency :[15, 17, 30, 22, 16, 100] Using the χ 2 test , examine the hypothesis that the distribution of grades is uniform.

* 9.4
* What is the purpose of blueprint in Flask?

Why Use Flask Blueprints?

* + Modularity: Blueprints allow you to break down your application into smaller, manageable components. Each blueprint can have its own routes, static files, and templates.
  + Reusability: You can reuse blueprints across different projects. For example, you might have an authentication blueprint that can be reused in multiple applications.
  + Organization: By using blueprints, you can organize your code better, making it easier to navigate and maintain, especially as your application grows.
  + Collaboration: Blueprints facilitate collaboration by allowing different team members to work on different parts of the application without conflicts.
  + Scalability: As your application grows, blueprints help manage the complexity by isolating features and components