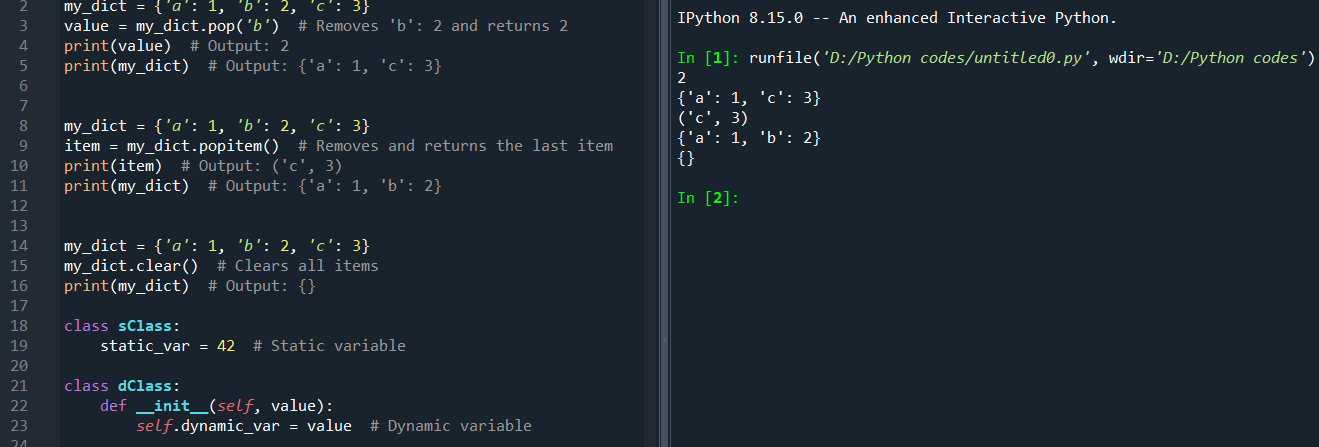
**ML ASSIGNMENT**

**1.1.1 What is the difference between static and dynamic variables in Python?**

Static variables, also known as class variables, have a fixed memory location throughout the execution of a program. They are declared within a class or a function and retain their values between function calls.

Dynamic variables, also called instance variables, are allocated memory during runtime. Unlike static variables, dynamic variables have a memory location that changes as the program executes.

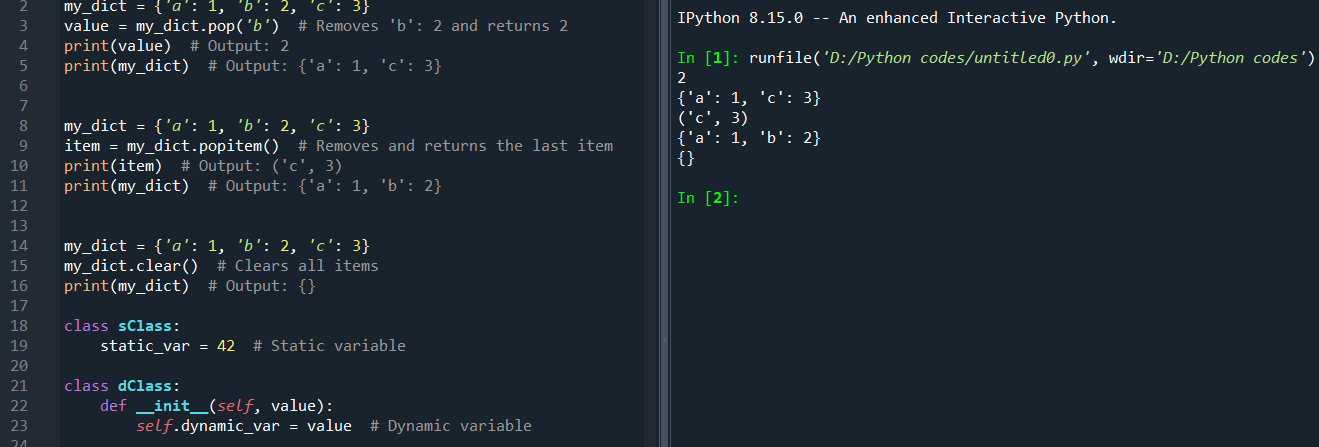


**1.1.2 Explain the purpose of "pop", "popitem", and "clear" in a dictionary with suitable examples.**

**1. pop:** Removes a specified key from the dictionary and returns its value. If the key is not found, an optional default value can be returned instead of raising a Key Error.

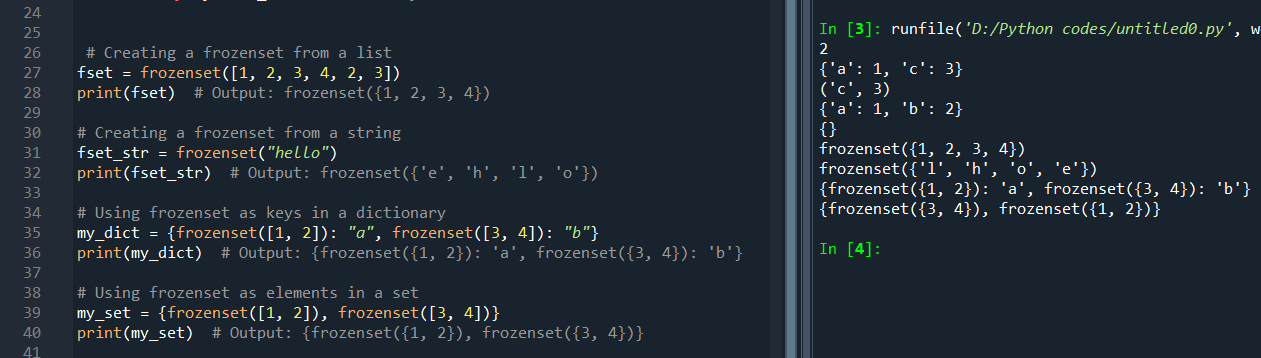
**2. popitem:** Removes and returns the last key-value pair as a tuple. This method is useful for destructively iterating through the dictionary.

**3. clear**: Removes all items from the dictionary, leaving it empty.



**3. What do you mean by Frozen Set? Explain it with suitable examples.**

A ‘frozen set’ in Python is an immutable version of a set. Once created, the elements of a frozen set cannot be changed, added, or removed. This makes frozen sets hashable and therefore suitable as keys in dictionaries or as elements of other sets. Like sets, frozen sets don’t allow duplicate elements. It does not maintain the order of elements.

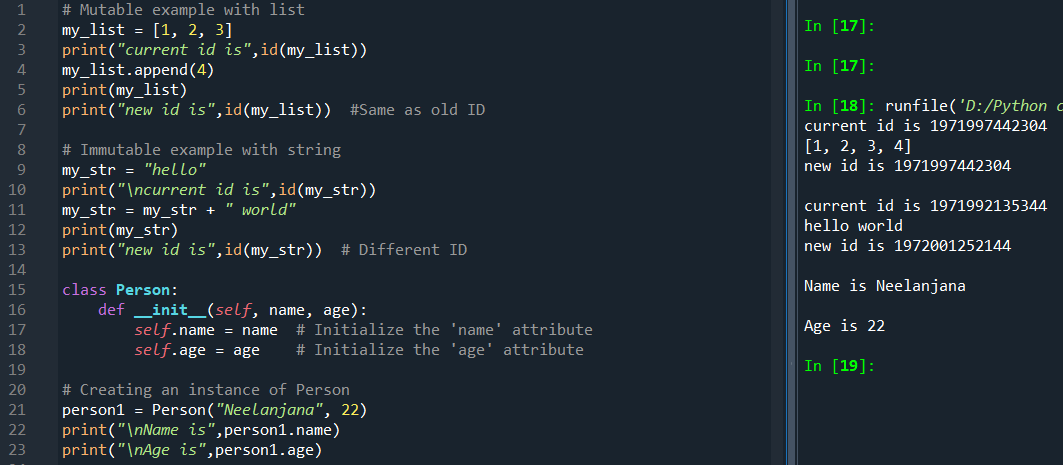


**4. Differentiate between mutable and immutable data types in Python and give examples of mutable and**

**immutable data types.**

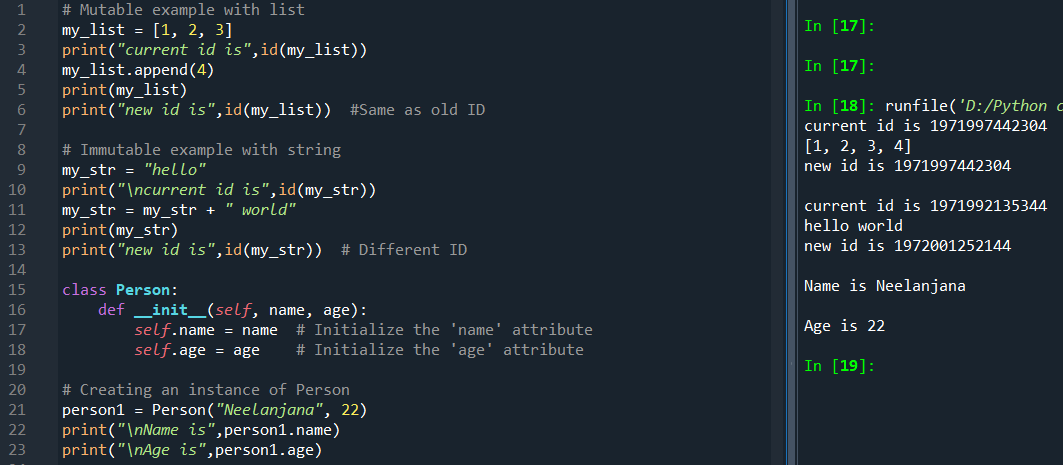
Mutable data types are those whose values can be modified after creation. This means the content of the object can be changed without changing its identity. It is useful when you need to change the size or content frequently, e.g. List, Dictionary, Set.

Immutable data types are those whose values cannot be changed after creation. Any modification to an immutable object, results in a new object being created. It is useful for fixed data and as keys in dictionaries due to their hashability, e.g. String, Tuple, Frozen set.



**5. What is \_\_init\_\_? Explain with an example.**

The \_\_init\_\_ method in Python is a special method used for initializing newly created objects of a class. It is commonly referred to as the constructor. When an object is instantiated, \_\_init\_\_ is automatically called to set up the object's initial state by assigning values to its properties. It is known as a "dunder" method (double underscore method). The first parameter is always self, which refers to the instance being created.



**6. What is docstring in Python? Explain with an example.**

A docstring in Python is a string literal that appears right after the definition of a function, method, class, or module. It is used to document the object and provide a convenient way of associating documentation with Python code. Typically, triple double quotes """ are used, but triple single quotes ''' can also be used. It must be the first statement in the function, method, class, or module. The docstring is stored in the \_\_doc\_\_ attribute of the object.

**7. What are unit tests in Python?**

Unit tests in Python are a type of automated testing that checks the smallest parts of an application, known as units, to ensure they function correctly. These tests are typically written to test individual functions or methods within a module. Each test is isolated from the others to ensure that the tests do not interfere with one another. It helps in catching errors early in the development process. Python's ‘unittest’ module is commonly used to write and run unit tests.

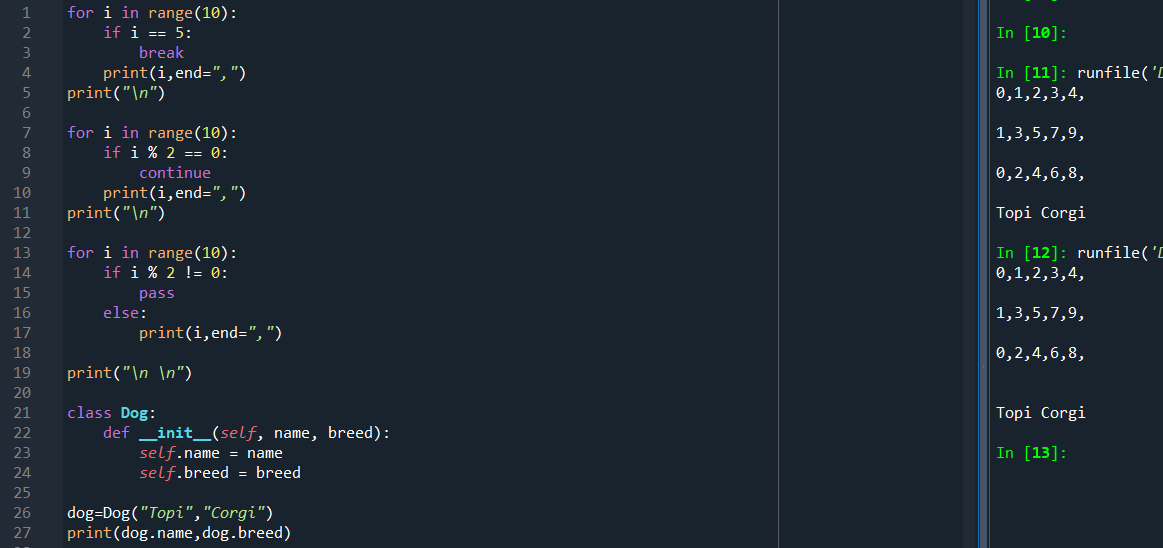
**8. What is break, continue, and pass in Python?**

In Python, break, continue, and pass are control flow statements to alter the behaviour of loops or as placeholders.

**1. break:** It exits the loop. When break is encountered, the loop ends and control goes to the first statement after the loop.

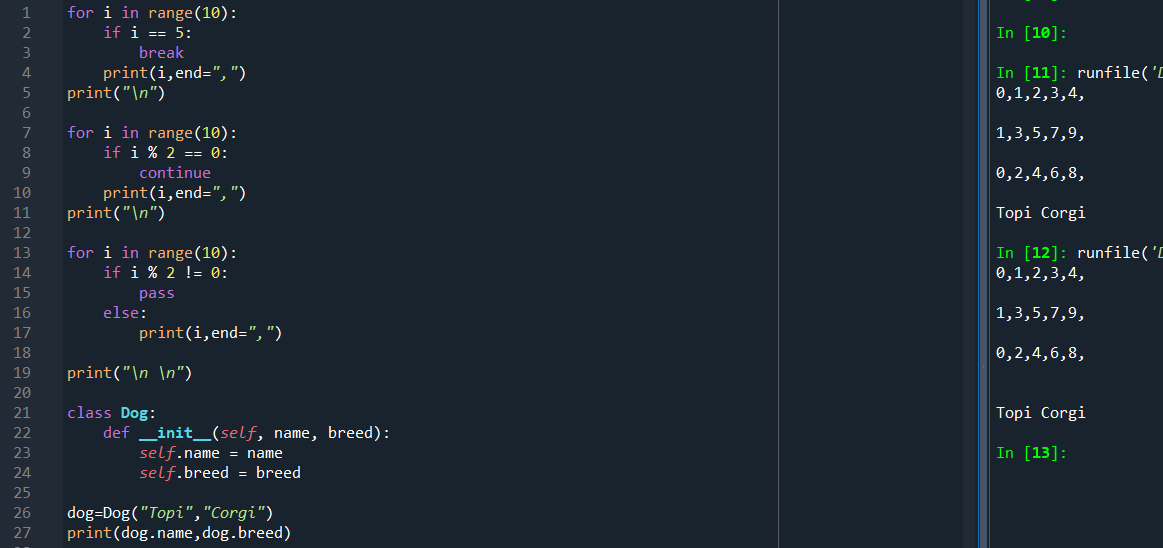
**2. continue:** It skips the rest of the code inside the current loop and goes to the next iteration. ‘continue doesn’t end the loop but moves to the next iteration.

**3. pass:** It does nothing. It’s a placeholder for future code. It is useful when you need to temporarily skip a part of the code that might raise an error.



**9. What is the use of self in Python?**

In Python “self” is a special parameter that refers to the instance of a class. It is used inside a class definition to refer to the class’s own instance variables and methods. When you define a class the first parameter in the constructor and instance methods must be “self”. This allows the instance of the class to refer to its own attributes and methods. The use of “self” helps in maintaining the uniqueness and identity of each instance of a class and helps in better organization and encapsulation of code. Also “self” enables object oriented programming in Python and makes the code more efficient and modular.

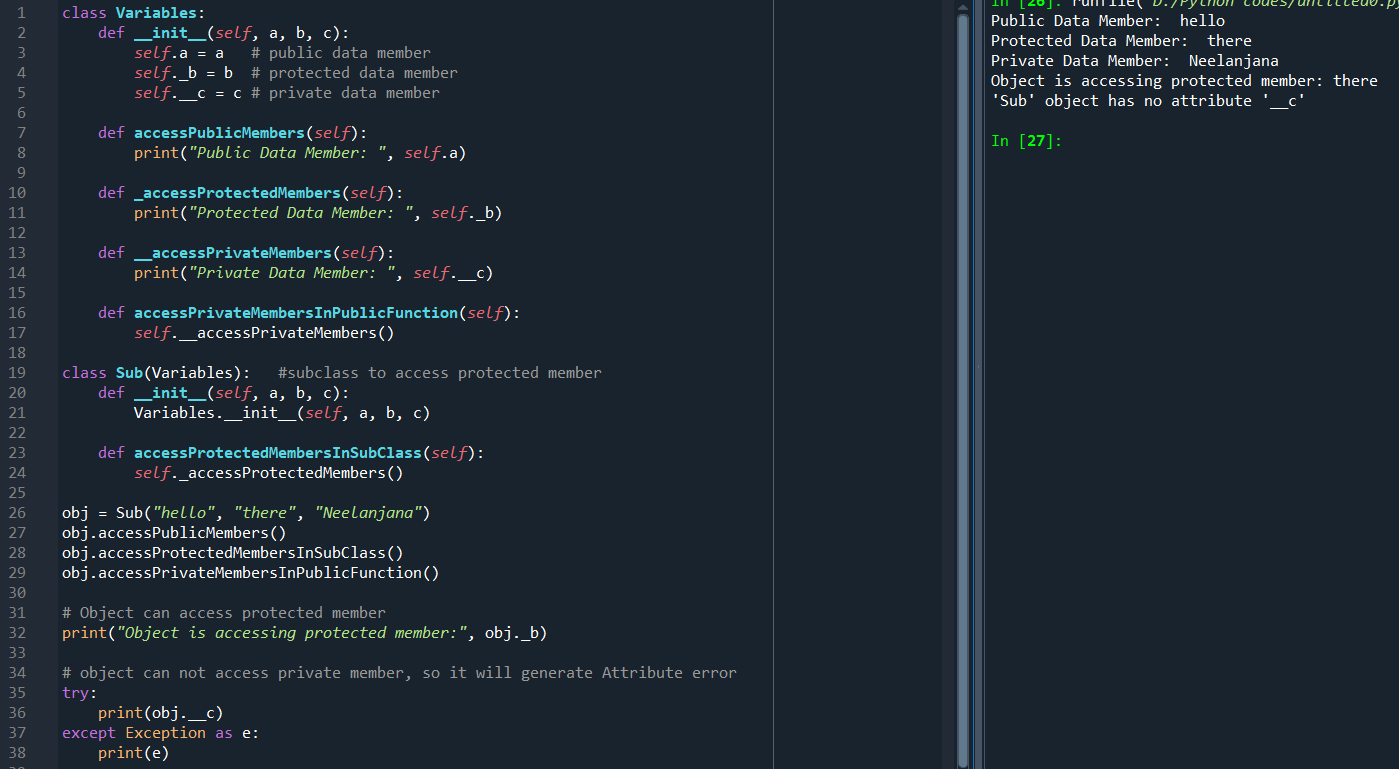


**10. What are global, protected, and private attributes in Python?**

**1. Global Attributes:** The global members of a class are easily accessible from any part of the program. All data members and member functions of a class are public by default.

**2. Protected Attributes:** The protected members of a class are only accessible to a class derived from it. Data members of a class are declared protected by adding a single underscore ‘\_’ symbol before the data member of that class.

**3. Private Attributes:** The private members of a class are accessible within the class only, the private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore ‘\_\_’ symbol before the data member of that class.



**1.1.11. What are modules and packages in Python?**

In Python, modules and packages are fundamental components for the organization and management of code. They help structure programs, promote code reuse, and make the code easier to understand and maintain.

**Modules:** A module is a single file (or files) imported under one import and used. Modules in Python can define functions, classes, and variables. They can also include runnable code.

**Packages:** A package is a directory that contains a special file named \_\_init\_\_.py along with one or more module files or sub-packages.

12. What are lists and tuples? What is the key difference between the two

13. What is an Interpreted language & dynamically typed language? Write 5 differences between them.

14. What are Dict and List comprehensions?

15. What are decorators in Python? Explain it with an example. Write down its use cases.

16. How is memory managed in Python?

17. What is lambda in Python? Why is it used?

18. Explain split() and join() functions in Python

19. What are iterators, iterable & generators in Python

20. What is the difference between xrange and range in Python

21. Pillars of Oops.

22. How will you check if a class is a child of another class

23. How does inheritance work in Python? Explain all types of inheritance with an example.

24. What is encapsulation? Explain it with an example.

1.1.25. What is polymorphism? Explain it with an example.

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

**Measure of central tendency:** It is a single value that attempts to describe a data set by identifying the central position within that set. It represents the centre point or typical value of the dataset.

**Common Measures:**

* **Mean**: It is the arithmetic average of a set of values. It can be calculated by adding all the dataset values and dividing it by the total number of data points.

Example: For the dataset [2, 3, 5, 7], the mean is (2+3+5+7)/4=4.25.

* **Median**: It is calculated by arranging the numbers in order and finding the middle value. If the count of numbers is odd, the median is the middle number. If the count is even, the median is the average of the two middle numbers.

Example: For the dataset [2, 3, 5, 7], the median is (3+5)/2​=4.

* **Mode**: The most frequently occurring value(s) in a dataset. It is calculated by identifying the number that appears most frequently.

Example: For the dataset [2, 3, 3, 5, 7], the mode is 3.

**Measures of dispersion:** It describes the spread or variability within a set of data. It indicates` how much the data values diverge from the central tendency.

**Common Measures:**

* **Range**: The difference between the highest and lowest values. *R=(Max−Min)*
* **Variance**: The average of the squared differences from the mean.
* **Standard Deviation**: The square root of the variance represents the average distance from the mean.
* **Interquartile Range (IQR)**: The range of the middle 50% of the data, calculated as the difference between the third quartile (Q3) and the first quartile (Q1). IQR=Q3−Q1.

**21. What do you mean by skewness? Explain its types. Use a graph to show.**

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. It indicates whether the data points are spread more to the left or the right of the distribution's peak.

**Positive Skewness (Right-Skewed)**:

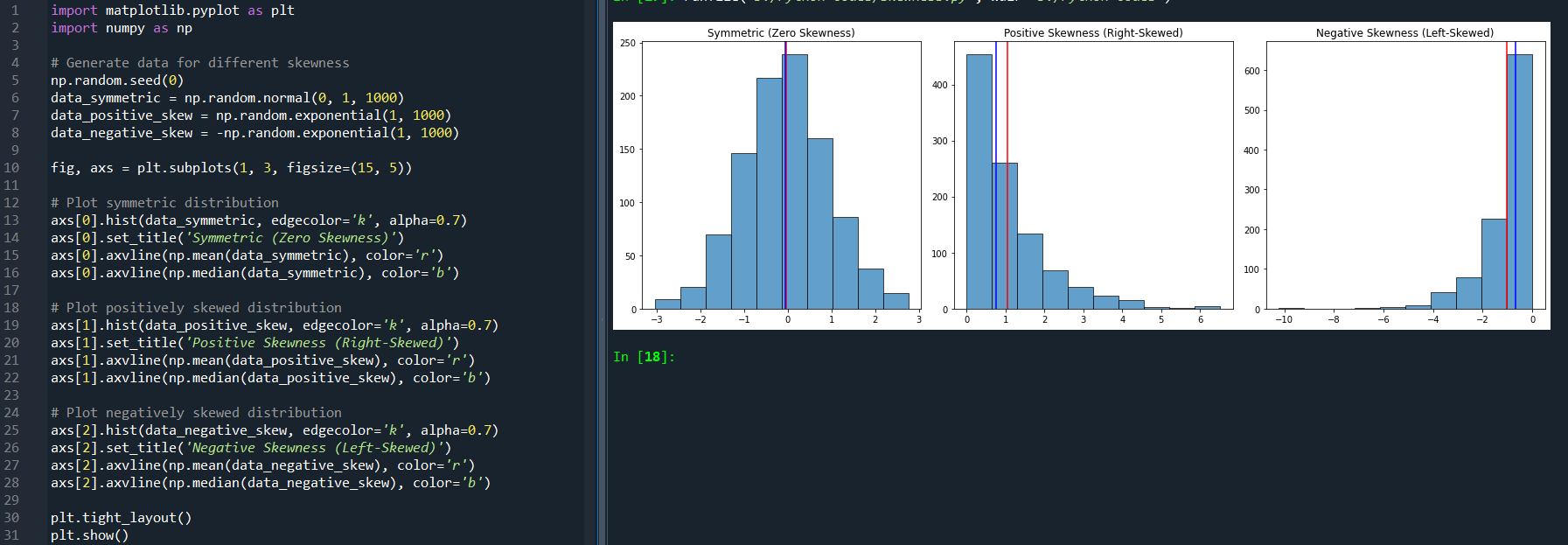
In a positively skewed distribution, the tail on the right side of the distribution is longer or fatter than the left side. Most of the data values (including the median) lie to the left of the mean. The mean is greater than the median.

**Negative Skewness (Left-Skewed)**:

In a negatively skewed distribution, the tail on the left side of the distribution is longer or fatter than the right side. Most of the data values (including the median) lie to the right of the mean. The mean is less than the median.

**Zero Skewness (Symmetric)**:

In a symmetric distribution, the left and right sides of the distribution are mirror images of each other. The mean and median are equal and lie at the centre of the distribution. This is also known as a normal distribution if the data follows a bell curve.



**22. Explain PROBABILITY MASS FUNCTION (PMF) and PROBABILITY DENSITY FUNCTION (PDF). and what is the difference between them?**

**Probability Mass function (PMF):**

Probability Mass Function is a mathematical function that describes the probability distribution of a **discrete random variable.** The PMF is represented as a histogram-like bar graph, where the x-axis represents the possible outcomes and the y-axis represents their corresponding probabilities.

The PMF of a discrete random variable assigns a probability to each possible value of the random variable. The probabilities assigned by the PMF must satisfy two conditions:

**a**. The probability assigned to each value must be non-negative (i.e., greater than or equal to zero).

**b**. The sum of the probabilities assigned to all possible values must equal 1.

Given a discrete random variable with a PMF ,the CDF F(x) is defined as:

**Probability Density Function (PDF):**

Probability Density Function is a mathematical function that describes the probability distribution of a**continuous random variable.** The probability density function (PDF) gives the density of the probability distribution over a range of values for a continuous random variable. The area under the PDF over a certain range of values gives the probability of the random variable falling within that range.

Given a continuous random variable with a PDF , the CDF is defined as:

**Key Differences between PMF and PDF:**

|  |  |
| --- | --- |
| **Probability Mass function (PMF)** | **Probability Density function (PDF)** |
| PMF is used for discrete random variables. | PDF is used for continuous random variables. |
| PMF gives the probability that the random variable is exactly equal to a particular value. | PDF gives the probability density at a point; probabilities for intervals are obtained by integrating the PDF over that interval. |
| For PMF, probabilities are summed over the possible values of the random variable. | For PDF, probabilities are found by integrating the PDF over an interval. |
| PMF values are probabilities and hence must be between 0 and 1. | PDF values can be greater than 1, as they represent density rather than direct probability. The area under the entire PDF curve must be 1, but individual values of the PDF can exceed 1. |

**23. What is correlation? Explain its type in detail. What are the methods of determining correlation?**

Correlation measures the strength and direction of a linear relationship between two variables. It quantifies how changes in one variable are associated with changes in another. The value of correlation ranges between -1 and 1, where **1** indicates a perfect positive linear relationship, **-1** indicates a perfect negative linear relationship, and **0** indicates no linear relationship.

**Types of Correlation:**

1. **Positive Correlation**:

* When one variable increases, the other variable also increases.
* Correlation coefficient is greater than 0 and up to 1.
* Example: Height and weight of individuals (typically taller people tend to weigh more).

1. **Negative Correlation**:

* When one variable increases, the other variable decreases.
* Correlation coefficient is less than 0 and down to -1.
* Example: The number of hours spent watching TV and scores on a test (more TV watching might be associated with lower test scores).

1. **Zero Correlation**:

* There is no relationship between the two variables.
* Correlation coefficient is around 0.
* Example: The number of tea cups consumed and the number of cars produced in a factory.

1. **Non-linear Correlation**:

* Variables may have a relationship, but it is not linear.
* This relationship might be quadratic, exponential, etc.
* Non-linear relationships are not captured by the linear correlation coefficient.

**Methods of Determining Correlation:**

1. **Pearson’s Correlation Coefficient**:

* Measures the linear relationship between two continuous variables.
* Assumes both variables are normally distributed.
* r ranges from -1 to 1.

1. **Spearman’s Rank Correlation Coefficient**:

* Non-parametric measure of rank correlation.
* Used when the data does not meet the assumptions of Pearson’s correlation, particularly for ordinal variables.
* ρ ranges from -1 to 1.

**25. Discuss the 4 differences between correlation and regression.**

|  |  |
| --- | --- |
| **Correlation** | **Regression** |
| It measures the strength and direction of a linear relationship between two variables. | It predicts the value of a dependent variable based on the value(s) of one or more independent variables. |
| It is used to quantify how closely related two variables are, without making any assumptions about causality. | It is used to model the relationship between variables, often to infer causality or make predictions. |
| Correlation is symmetric, meaning the correlation between X and Y is the same as between Y and X. It does not distinguish between dependent and independent variables. | Regression is asymmetric; it explicitly distinguishes between dependent and independent variables. |
| It is represented graphically using a scatterplot with no specific emphasis on predicting one variable from the other. | It is represented graphically using a scatterplot with a fitted regression line indicating the predicted values of the dependent variable. |
| The correlation coefficient r is always between -1 and 1, where r =1, -1, 0 represent a perfect positive linear relationship, perfect negative linear relationship, and no linear relationship, respectively. | The regression coefficients are not restricted to a specific range and are expressed in the units of the dependent variable per unit of the independent variable. A regression equation is of the form y=β0+β1x+ϵ, where y is the dependent variable, x is the independent variable, β0 is the intercept, β1 is the slope, and ϵ is the error term. |

**28. What is Normal Distribution? What are the four Assumptions of Normal Distribution? Explain in detail.**

The normal distribution, also known as the Gaussian distribution, is a continuous probability distribution that is symmetric and bell-shaped. It is characterized by its mean (µ) and standard deviation (σ), which determine the location of the centre and the spread of the distribution, respectively. In normal distribution, approximately 68% of the data falls within one standard deviation of the mean, 95% within two, and 99.7% within three.

### Assumptions of Normal Distribution:

* **Linearity and Additivity**: The relationship between the dependent variable and the independent variables is linear and additive. Each predictor variable contributes linearly to the response variable, and the combined effect of multiple predictors is simply the sum of their individual effects.
* **Independence**: The observations are independent of each other. This implies that the value of one observation does not influence or is not influenced by another. Independence ensures that the error terms are not correlated. If observations are dependent. In time series data, this would mean that the value of a variable at one point in time is not correlated with its value at another time.
* **Homoscedasticity**: The variance of errors is constant across all levels of the independent variables. This means that the error term in a regression model is the same across all observations, which ensures that the model’s predictions are equally reliable for all values of the independent variables. If the variance of errors changes (heteroscedasticity), it can indicate that the model is not capturing all the underlying patterns, leading to inefficient estimations.
* **Normality of Errors**: While the dependent variable does not need to be normally distributed, the distribution of errors should follow a normal distribution. This is crucial for conducting hypothesis tests and constructing confidence intervals. The normality of residuals implies that most of the errors are near the mean error, with few large errors. If the errors are not normally distributed, it can affect the reliability of confidence intervals and hypothesis tests, leading to incorrect inferences.

**29.Write all the characteristics or Properties of the Normal Distribution Curve.**

* **Symmetry:** The normal distribution is perfectly symmetrical around its mean (μ). This means that the left and right sides of the curve are mirror images of each other.
* **Mean, Median, and Mode:** In a normal distribution, the mean, median, and mode are all equal and located at the center of the distribution. This central point is the peak of the curve.
* **Bell-Shaped Curve:** The distribution forms a bell-shaped curve, which extends infinitely in both directions, approaching but never touching the horizontal axis (asymptotic).
* **Unimodal:** The normal distribution has a single peak (mode), indicating that it is unimodal. There is one value that is the most frequent in the dataset.
* **68-95-99.7 Rule (Empirical Rule):** Approximately 68% of the data lies within one standard deviation (σ) of the mean (μ). Approximately 95% of the data lies within two standard deviations of the mean. Approximately 99.7% of the data lies within three standard deviations of the mean.
* **Asymptotic Nature:** The tails of the normal distribution curve approach the horizontal axis but never touch it. This property indicates that there are infinitely small probabilities of extreme values.
* **Defined by Two Parameters:** The normal distribution is fully described by two parameters: the mean (μ), which determines the center, and the standard deviation (σ), which determines the spread or width of the distribution.
* **No Skewness:** The skewness of a normal distribution is zero, reflecting its perfect symmetry.
* **Continuous Probability Distribution:** The normal distribution is a continuous distribution, meaning that the random variable can take on any real number value within the range.
* **Kurtosis:** The kurtosis of a normal distribution is zero, indicating that it has the same level of peakedness as the standard normal distribution. It is considered mesokurtic.
* **Probability Density Function (PDF):** The probability density function of the normal distribution is given by:
* **Area Under the Curve:** The total area under the normal distribution curve is equal to 1, representing the total probability of all possible outcomes.
* **Standard Normal Distribution:** When the mean (μ) is 0 and the standard deviation (σ) is 1, the distribution is known as the standard normal distribution.
* **Additivity of Independent Normal Variables:** If two independent random variables are normally distributed, their sum is also normally distributed.
* **Central Limit Theorem:** The Central Limit Theorem states that the sum (or average) of a large number of independent, identically distributed random variables will be approximately normally distributed, regardless of the original distribution of the variables.

**30.Which of the following options are correct about Normal Distribution Curve.**

**(a) 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.**

**(b) Mean ±1S.D. (i,e.μ ± 1σ) covers 68.268% area, 34.134 % area lies on either side of the mean.**

**(c) Mean ±2S.D. (i,e. μ ± 2σ) covers 95.45% area, 47.725% area lies on either side of the mean.**

**(d) Mean ±3 S.D. (i,e. μ ±3σ) covers 99.73% area, 49.856% area lies on the either side of the mean.**

**(e) Only 0.27% area is outside the range μ ±3σ.**

All statements are correct, since in option (a) approximately 68% of the data lies within one standard deviation (σ) of the mean (μ), in option (b) approximately 95% of the data lies within two standard deviations of the mean, and in option (c) and (d) approximately 99.7% of the data lies within three standard deviations of the mean.

**31. 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?**

where μ=60 and σ=10

**(i) Between 60 and 72**

For 60:

For 72:

Using z-table, we find the area between these z-scores:

Area to the left of z = 0 is 0.5

Area to the left of z = 1.2 is 0.8849

Percentage of items between 60 and 72: 0.8849−0.5= 0.3849 =38.49%

**(ii) Between 50 and 60**

For 50:

For 60:

Using the z-table, we find the area between these z-scores:

Area to the left of z = -1 is 0.1587

Area to the left of z = 0 is 0.5

Percentage of items between 50 and 60: 0.5−0.1587=0.3413=34.13%

**(iii) Beyond 72**

For 72:

Using the z-table, we find the area to the right of z = 1.2:

Area to the left of z = 1.2 is 0.8849

Percentage of items beyond 72: 1−0.8849 = 0.1151=11.51%

**(iv) Between 70 and 80**

For 70: 0:

For 80: z=0:

Using the z-table, we find the area between these z-scores:

Area to the left of z = 1 is 0.8413

Area to the left of z = 2 is 0.9772

Percentage of items between 70 and 80: 0.9772−0.8413 =0.1359=13.59%

**32. 15000 students sat for an examination. The mean mark 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?**

μ= 49

σ= 6

Total number of students: 15,000

1. **For more than 55 marks:**

Using the z-table, we find the area between these z-scores:

Area to the left of z = 1 is 0.8413

Area to the right of z = 1 is 1-0.8413=0.1587

Percentage of students who scored more than 55 marks: 15.87%

Number of students who scored more than 55 marks: 0.1587 x 15000 = 2380.5 = 2381 students

**(b) For more than 70 marks:**

Using the z-table, we find the area between these z-scores:

Area to the left of z = 3.5 is 0.9998

Area to the left of z = 3.5 is 1-0.9998=0.0002

Percentage of students who scored more than 70 marks: 0.02%

Number of students who scored more than 70 marks: 0.0002 x 15000 = 3 students

**33. If the height of 500 students is normally distributed with a mean of 65 inches and standard deviation of 5 inches. How many students have a height: a) greater than 70 inches b) between 60 and 70 inches.**

μ= 65

σ= 5

Total number of students: 500

1. **Greater than 70 inches:**

Using the z-table, we find the area between these z-scores:

Area to the left of z = 1 is 0.8413

Area to the right of z = 1 is 1-0.8413=0.1587

Percentage of students with height greater than 70 inches: 15.87%

Number of students with height greater than 70 inches: 0.1587 x 500 = 79.35 = 79 students

1. **Between 60 and 70 inches:**

**For 60 inches:**

**For 70 inches:**

Using the z-table, we find the area between these z-scores:

Area to the left of z = -1 is 0.1587

Area to the left of z = 1 is 0.8413

Percentage of students with height between 60 and 70 inches: 0.8413-0.1587 = 0.6826 = 68.26%

Number of students with height between 60 and 70 inches: 0.6913 x 500 = 341.3 = 341 students

**34. What is the statistical hypothesis? Explain the errors in hypothesis testing. Explain Sample. What are Large Samples & Small Samples?**

**Statistical Hypothesis:**

A statistical hypothesis is an assumption or a claim about a population parameter. This could be about the mean, variance, proportion, or other characteristics of a population. Hypothesis testing is a method used to decide whether the data at hand sufficiently supports a particular hypothesis.

There are two types of hypotheses in hypothesis testing:

**Null Hypothesis (H0​)**: A null hypothesis is a type of statistical hypothesis that proposes that no statistical significance exists in a set of given observations. It often posits that any observed effect is due to random chance.

**Alternative Hypothesis (H1)**: An alternate hypothesis is a type of statistical hypothesis that proposes that statistically significant differences occur between two or more experimental or control groups.

**Errors in Hypothesis Testing:**

**Type I Error (False Positive)**:

* This error occurs when the null hypothesis is rejected when it is actually true.
* The probability of making a Type I error is denoted by α, also known as the significance level. Common values for α are 0.05, 0.01, and 0.10.
* For example, concluding that a new drug is effective when it is not.

**Type II Error (False Negative)**:

* This error occurs when the null hypothesis is not rejected when it is actually false.
* The probability of making a Type II error is denoted by β.
* The power of the test is 1−β, which is the probability of correctly rejecting the null hypothesis when it is false.
* For example, concluding that a new drug is not effective when it actually is.

**Sample:**

A sample is a subset of individuals or observations from a population used to infer characteristics about the entire population. Sampling is crucial because it is often impractical or impossible to study the whole population.

**Large Samples**:

* Typically, a sample is considered large if it has 30 or more observations (n ≥ 30).
* With large samples, the sampling distribution of the sample mean tends to be approximately normally distributed, regardless of the shape of the population distribution, due to the Central Limit Theorem.
* Large sample sizes provide more reliable and stable estimates of population parameters.
* Example: If we collect heights from 100 randomly selected students in a university, this is considered a large sample. Statistical methods assuming normal distribution can be applied even if the population distribution is unknown.

**Small Samples**:

* Samples with fewer than 30 observations (n < 30) are considered small.
* The sampling distribution of the sample mean may not be normally distributed if the underlying population is not normal.
* Special statistical methods, such as the t-distribution, are used to perform hypothesis tests and create confidence intervals with small samples.
* Small samples may be more prone to sampling variability and may not represent the population as accurately as large samples.
* Example: If we collect heights from 10 randomly selected students, this is a small sample. We must be cautious and often use methods like the t-distribution for analysis, especially if the population distribution is not known to be normal.

**35. 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).**

Null Hypothesis (H0): The population standard deviation σ=10.5

Alternative Hypothesis (H1): The population standard deviation σ≠10.5

Given Data:

Sample size n=25

Sample standard deviation s=9

Hypothesized population standard deviation σ0=10.5

= 17.63

df = n - 1 = 25 - 1 = 24

α = 0.05

Lower critical value =

Upper critical value =

Since 17.63 is between 13.848 and 36.415, the chi-square test provides inconclusive evidence regarding the population standard deviation. We cannot reject the null hypothesis that the population standard deviation 𝜎 = 10.5.

**37.100 students of a PW IOI obtained the following grades in the Data Science paper:**

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

**Total Frequency: [15, 17, 30, 22, 16, 100]**

**Using the χ2 test, examine the hypothesis that the distribution of grades is uniform.**

Null Hypothesis: Distribution of the grades is uniform

For a uniform distribution, each grade should have the same frequency. Since there are 100 students and 5 grades, the expected frequency for each grade is 100/5=20

|  |  |  |  |
| --- | --- | --- | --- |
| **Grade** | **Observed**  **Frequency (f0)** | **Expected**  **Frequency (fe)** |  |
| A | 15 | 20 | 1.25 |
| B | 17 | 20 | 0.45 |
| C | 30 | 20 | 5.00 |
| D | 22 | 20 | 0.20 |
| E | 16 | 20 | 0.80 |
| Total | 100 | 100 | 7.70 |

= 7.70

df = number of categories-1 = 5-1 = 4

For a significance level(α) of 0.05 and 4 degrees of freedom, the critical value from the chi-square distribution table is approximately 9.488.

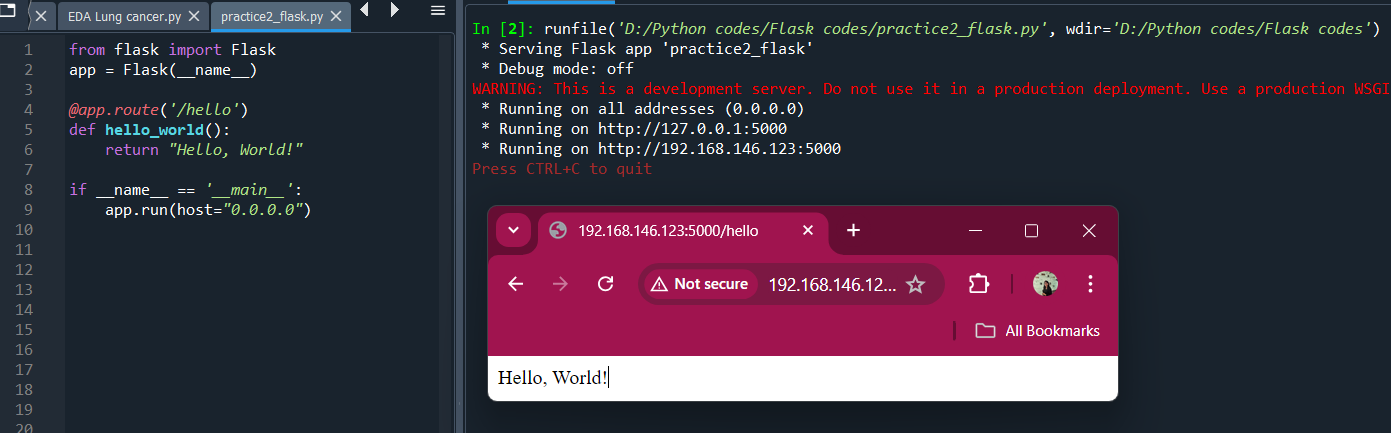
Since 7.70 is less than 9.488 we fail to reject the null hypothesis, therefore we conclude distribution of grades is uniform.

**38. Anova Test: To study the performance of three detergents and three different water temperatures, the following whiteness readings were obtained with specially designed equipment:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Water Temp** | **Detergent A** | **Detergent B** | **Detergent C** |
| Cold Water | 57 | 55 | 67 |
| Warm Water | 49 | 52 | 68 |
| Hot Water | 54 | 46 | 58 |

**39. How would you create a basic Flask route that displays "Hello, World!" on the homepage?**

* Import Flask: The Flask class is imported from the flask package.
* Create a Flask Instance: app = Flask(\_\_name\_\_) creates an instance of the Flask class.
* Define a Route: The @app.route('/') decorator defines a route for the homepage (/). When this route is accessed, the hello\_world() function is called, and its return value is displayed on the page.
* Run the Application: app.run(host= “0.0.0.0”) runs the Flask application.
* Access the Homepage: <http://127.0.0.1:5000/hello> is typed in a web browser, "Hello, World!" is displayed.

****

**40. Explain how to set up a Flask application to handle form submissions using POST requests.**

**50. Machine Learning**

**50.1 What is the difference between Series & Dataframe?**

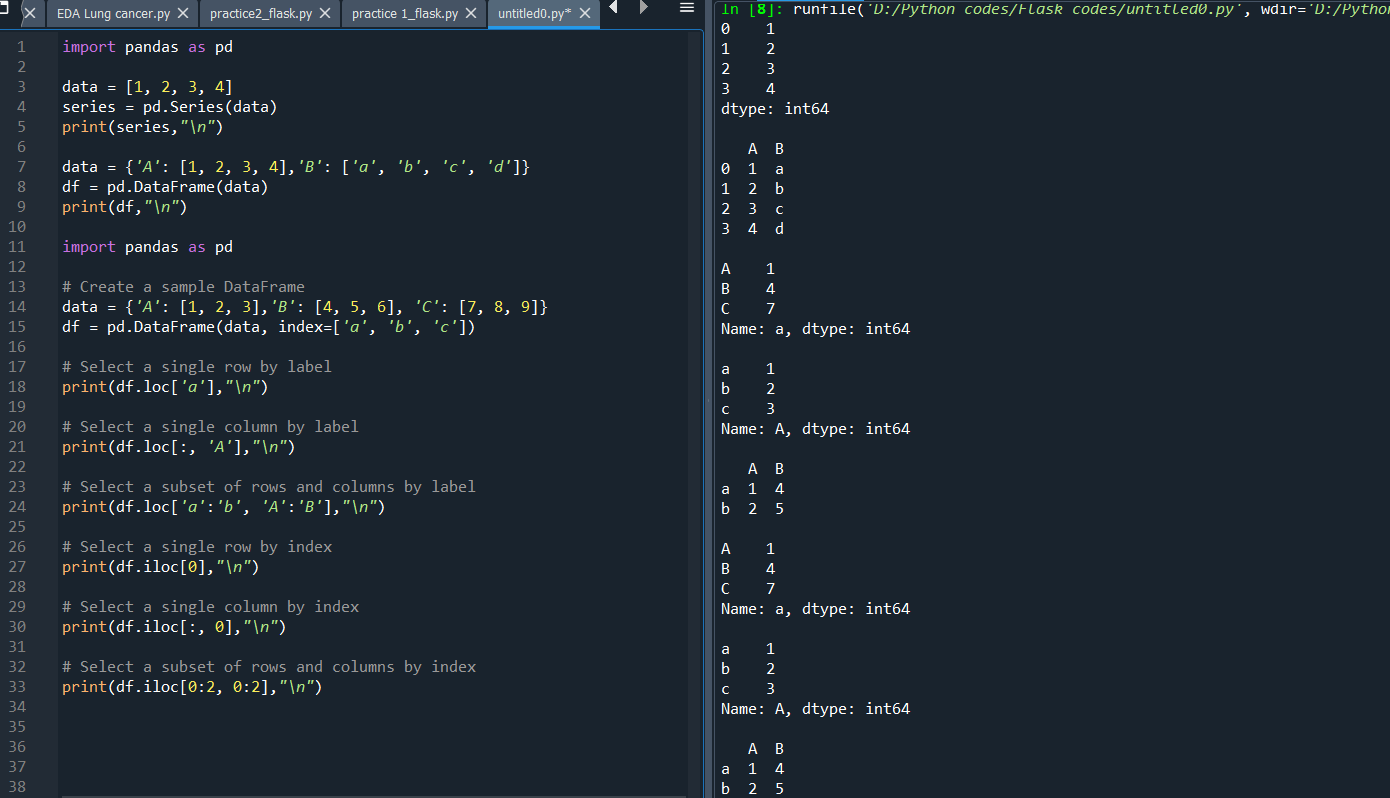
In pandas, Series and DataFrame are two fundamental data structures

**1. Series:**

* Definition: A Series is a one-dimensional labelled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).
* Structure: It can be seen as a single column of data.
* Index: Each element in a Series is associated with a unique label (an index), similar to an array.

**2. DataFrame:**

* Definition: A DataFrame is a two-dimensional labeled data structure with columns of potentially different types.
* Structure: It can be seen as a table or a collection of Series objects (each column in a DataFrame is a Series).
* Index: It has both row and column indices.



**50.3 Difference between loc and iloc.**

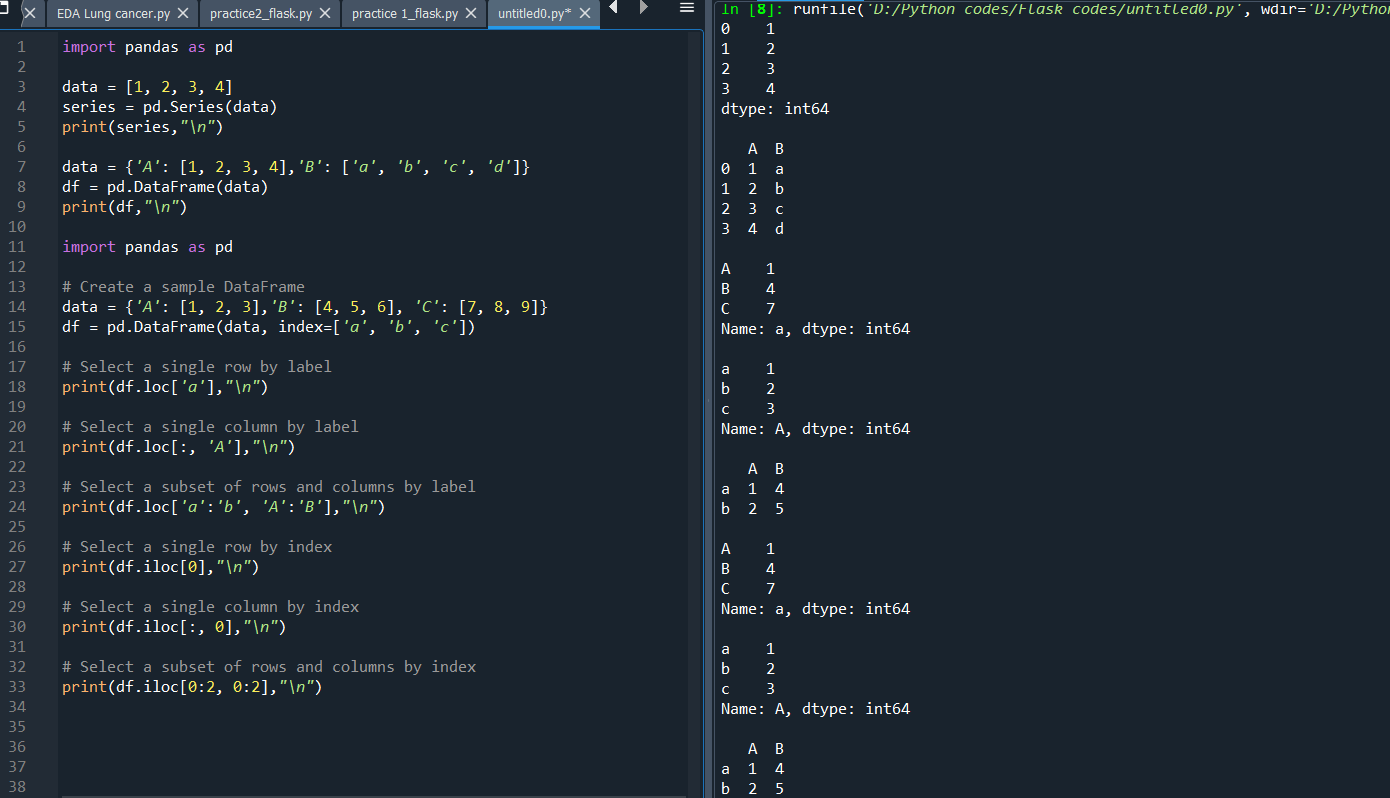
In pandas, loc and iloc are two primary methods for selecting data from a DataFrame.

1. **loc (Label-based indexing):**

* Definition: loc is used for label-based indexing. It allows you to select rows and columns by their labels (names).
* Usage:
  + Access by label: Used to select data using the row and column labels.
  + Boolean indexing: Can be used to select data based on a condition.

1. **iloc (Integer position-based indexing):**

* Definition: iloc is used for integer position-based indexing. It allows you to select rows and columns by their integer positions (indices).
* Usage:
  + Access by index: Used to select data using the row and column indices.
  + Slices: Can be used with slice notation to select data by range of indices.

****

**50.4 What is the difference between supervised and unsupervised learning?**

|  |  |  |
| --- | --- | --- |
|  | **Supervised Learning** | **Unsupervised Learning** |
| **Definition** | Supervised learning involves training a  model on a labeled dataset, which means  the input data comes with corresponding  output labels. The model learns to map  inputs to the correct output. | 1. Unsupervised learning involves training a 2. model on data that does not have labelled 3. responses. The model tries to learn the 4. underlying structure or distribution in the 5. data. |
| **Data** | * + **Labeled Data**: Each training example   + is a pair consisting of an input object   + (typically a vector) and a desired   + output value (label).   + **Examples**: (input, output) pairs, such   + as ([feature1, feature2, ...], label). | * + **Unlabeled Data**: Only input data is provided,   + and there are no corresponding output labels.   + **Examples**: [feature1, feature2, ...]. |
| **Goal** | The goal is to learn a mapping from  inputs to outputs that can generalize well  to new, unseen data. This mapping  allows the model to make predictions  on new data. | 1. The goal is to infer the natural structure 2. present within a set of data points. This can 3. include grouping similar data points together 4. or reducing the dimensionality of the data. |
| **Common**  **Algorithms** | * + Regression: Linear Regression,   + Polynomial Regression   + Classification: Logistic Regression,   + Support Vector Machines,   + Decision Trees, Random Forest,   + Neural Networks | * + Clustering: K-Means, Hierarchical   + Clustering, DBSCAN Dimensionality   + Reduction: Principal Component Analysis   + (PCA), t-Distributed Stochastic Neighbor   + Embedding (t-SNE)   + Association: Apriori, Eclat |
| **Use cases** | * + Spam detection in emails (classification)   + Predicting house prices (regression)   + Image recognition (classification)   + Sentiment analysis (classification) | * + Customer segmentation for marketing   + (clustering)   + Anomaly detection in network security   + (clustering)   + Market basket analysis for retail   + (association)   + Reducing the number of features in a dataset   + while retaining its variance (dimensionality   + reduction) |
| **Examples** | * + Given historical weather data (inputs)   + and weather conditions (outputs), predict future weather conditions.   + Given an image of a handwritten digit   + (input), classify it as one of the digits   + from 0 to 9 (output). | * + Given a set of customer data, group   + customers with similar purchasing behaviors   + together (clustering).   + Given a dataset of movies, find groups of   + similar movies based on user ratings   + (clustering).   + Identify patterns or relationships in   + transactional data, such as which items are   + frequently bought together (association). |

**50.5 Explain the bias-variance tradeoff.**

The bias-variance tradeoff is a fundamental concept in machine learning that addresses the balance between a model's ability to generalize to new data and its performance on the training data. It involves understanding how different sources of error affect a model's accuracy and how to minimize these errors to improve model performance.

**Bias:** Bias refers to the error introduced by approximating a real-world problem, which may be complex, by a simplified model. It represents the assumptions made by the model to make the target function easier to learn.

**Variance:** Variance refers to the model's sensitivity to the fluctuations in the training data. It captures how much the model's predictions would change if it were trained on a different training set.

**High Bias**: A model with high bias pays very little attention to the training data and oversimplifies the model. This leads to high training error and high-test error due to underfitting. Example: A linear regression model applied to a dataset that has a nonlinear relationship.

**High Variance**: A model with high variance pays a lot of attention to the training data, including noise and random fluctuations, and captures too much detail. This leads to low training error but high test error due to overfitting. Example: A decision tree model with many branches applied to a small dataset.

**Bias Variance Tradeoff:**

* **High Bias and Low Variance**: Models are simple with not enough complexity to capture the underlying patterns of the data (underfitting).
* **Low Bias and High Variance**: Models are too complex and capture noise along with the underlying patterns of the data (overfitting).
* **Optimal Model**: The goal is to find a model that appropriately balances bias and variance, minimizing the total error.

**Total Error =** Bias2+Variance+Irreducible Error

1. **High Bias (Underfitting)**: The model is too simple.

* **Training Error**: High
* **Test Error**: High
* **Behavior**: Fails to capture the underlying structure of the data.

1. **High Variance (Overfitting)**: The model is too complex.

* **Training Error**: Low
* **Test Error**: High
* **Behavior**: Captures noise along with the underlying structure of the data.

1. **Balanced Bias-Variance**: The model is appropriately complex.

* **Training Error**: Low
* **Test Error**: Low
* **Behavior**: Captures the underlying structure without fitting to noise.

**Managing the Tradeoff:**

1. **Model Complexity**:

Increasing complexity reduces bias but increases variance.

Decreasing complexity increases bias but reduces variance.

1. **Techniques**:

Regularization: Prevents overfitting by adding a penalty for complexity.

Cross-Validation: Helps estimate model performance and select optimal complexity.

More Data: Helps reduce variance by providing more training examples.

**50.6 What are precision and recall? How are they different from accuracy?**

**Precision:** Precision measures the proportion of true positive predictions among all positive predictions made by the model. Precision tells you how many of the predicted positive cases are actually positive. High precision indicates that the model makes fewer false positive predictions.

Precision=True Positives/ (True Positives + False Positives)​

**Recall:** Recall (or Sensitivity) measures the proportion of true positive predictions among all actual positive cases. Recall tells you how many of the actual positive cases were correctly identified by the model. High recall indicates that the model is able to identify most of the positive cases.

Recall=True Positives/ (True Positives + False Negatives)

**Accuracy:** Accuracy measures the proportion of correct predictions (both true positives and true negatives) among all predictions made. Accuracy tells you how often the model is correct overall, considering both positive and negative cases.

Accuracy= (True Positives + True Negatives)/ Total Number of Predictions

**Difference between Precision, Recall and Accuracy:**

1. **Focus**:

* **Precision**: Focuses on the quality of the positive predictions.
* **Recall**: Focuses on the quantity of the actual positives that were correctly predicted.
* **Accuracy**: Provides an overall measure of the model’s performance, including both positive and negative cases.

1. **Sensitivity to Class Imbalance**:

* **Precision**: High precision means that false positives are low, which is particularly important when the cost of false positives is high.
* **Recall**: High recall means that false negatives are low, which is important when the cost of missing positive cases is high.
* **Accuracy**: Can be misleading in cases of class imbalance, where one class is much more frequent than the other. For example, if 95% of cases are negative, a model that always predicts negative would still have high accuracy (95%) but would be ineffective at identifying the positive cases.

1. **Use Cases**:

* **Precision**: Important in scenarios where false positives are costly (e.g., spam detection where incorrectly marking a legitimate email as spam can be problematic).
* **Recall**: Important in scenarios where missing positive cases is costly (e.g., medical diagnoses where failing to identify a disease could be dangerous).
* **Accuracy**: Useful when class distribution is balanced and when both false positives and false negatives are equally important.

**50.7 What is overfitting and how can it be prevented?**

Overfitting occurs when a machine learning model learns the details and noise in the training data to the extent that it negatively impacts the model's performance on new data. This usually happens when the model is too complex, having too many parameters relative to the number of observations. As a result, the model fits the training data very well but fails to generalise to unseen data, leading to poor performance on the test set or in real-world applications.

**How to Prevent Overfitting:**

There are several strategies to prevent or mitigate overfitting:

1. **Simplify the Model**: Use fewer parameters or select a simpler model architecture.
2. **Regularization**: Add a penalty for large coefficients in the model, or randomly drop units (along with their connections) during training to prevent the neural network from becoming overly dependent on any particular feature.
3. **Cross-Validation**: Use techniques like k-fold cross-validation to ensure the model's performance is consistent across different subsets of the data.
4. **Data Augmentation**: Increase the size of your training set by adding transformed versions of your existing data (e.g., rotating, flipping, and scaling images in image classification tasks).
5. **Pruning**: In decision tree models, reduce the size of the tree by removing sections that provide little power in predicting target variables.
6. **Early Stopping**: Stop training when the performance on a validation set starts to degrade, which can indicate the model is beginning to overfit.
7. **Increase Training Data**: More data can help the model generalize better, provided it is representative of the data the model will encounter in practice.
8. **Ensemble Methods**: Combine predictions from multiple models to reduce overfitting. Techniques like bagging, boosting, or stacking can be effective.

**50.8 Explain the concept of cross-validation.**

Cross-validation is a statistical method used to evaluate the performance and generalizability of a machine-learning model. It involves partitioning the original dataset into a training set to train the model and a test set to evaluate it. It helps assess how well a model will generalize to an independent dataset. The key idea is to ensure that every data point has a chance to be in the training and test sets, which helps in understanding how the model will perform on unseen data. By using different subsets of the data for training and validation, cross-validation helps detect and prevent overfitting. It provides a reliable metric for model selection and hyperparameter tuning, allowing for comparisons between different models or configurations.

**Common Types of Cross-Validation:**

1. **K-Fold Cross-Validation**: The dataset is randomly partitioned into k equal-sized folds. The model is trained k times, each time using a different fold as the validation set and the remaining k-1 folds as the training set. The final evaluation metric is the average of the metrics obtained from each fold. Common choices for k are 5, 10.
2. **Leave-One-Out Cross-Validation**: A special case of k-fold cross-validation where k equals the number of observations in the dataset. Each observation is used as a validation set once, and the model is trained on the remaining observations. It is computationally expensive but useful for small datasets.
3. **Stratified K-Fold Cross-Validation**: A variation of k-fold cross-validation that maintains the percentage of samples for each class across all folds. Especially useful for imbalanced datasets.
4. **Time Series Cross-Validation**: For time series data, it’s crucial to preserve the temporal order. Therefore, folds are created in such a way that past data points are used to predict future data points. Techniques like **Rolling Cross-Validation** or **Walk-Forward Validation** are commonly used.
5. **Repeated K-Fold Cross-Validation**: The k-fold cross-validation process is repeated multiple times with different random splits of the data. It helps reduce the variability in the model's performance estimates.

**Benefits of Cross-Validation:**

* Provides a more reliable estimate of model performance than a single train-test split.
* Maximizes the use of the dataset, particularly valuable when data is limited.
* By averaging the results over multiple splits, cross-validation reduces the variability associated with a single train-test split.

**50.9 What is the difference between a classification and a regression problem?**

|  |  |
| --- | --- |
| **Classification** | **Regression** |
| It involves predicting a categorical label or class for a given input. The goal is to categorize inputs into one of several predefined classes. | It involves predicting a continuous numerical value based on input data. The goal is to model the relationship between the input variables and the output continuous variable. |
| The output of a classification model is discrete. For example, it could be a binary outcome (e.g., "spam" or "not spam" in email classification) | The output of a regression model is continuous. It can take any real-number value, and there are no predefined categories. |
| Example Algorithms: Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, Neural Networks (for classification tasks), k-Nearest Neighbours, Naive Bayes. | Example Algorithms: Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression, Support Vector Regression, Neural Networks (for regression tasks), and k-Nearest Neighbours (for regression). |
| Models interpret the probability of each class and often involve decision boundaries. | Models interpret the relationship between independent variables and a dependent continuous variable. |
| Used in scenarios where the decision is categorical, such as medical diagnosis (disease present or not), fraud detection, sentiment analysis. | Used where the prediction involves numerical values, such as stock price prediction, sales forecasting, or any type of trend prediction. |
| It typically uses accuracy, precision, recall, F1 score, etc., which are based on correctly identifying the categories. | It uses metrics like MSE, MAE, RMSE, which measure the difference between predicted and actual continuous values. |
| Example: Spam Detection, Image Classification | Example: House Price Prediction, Temperature Forecasting |

**50.10 Explain the concept of ensemble learning.**

Ensemble learning is a machine learning paradigm where multiple models, often referred to as "weak learners" or "base models," are trained and combined to solve the same problem. The main idea behind ensemble learning is that by aggregating the predictions of multiple models, the ensemble model can achieve better performance, generalize better, and be more robust than any single model alone. The individual models in an ensemble should ideally make different errors on the test data. The greater the diversity among the models, the more likely the ensemble is to perform better, as different models may compensate for each other's weaknesses.

Some of the aggregation methods include:

* **Averaging**: For regression tasks, the predictions of the individual models are averaged.
* **Voting**: For classification tasks, the class predicted by the majority of the models is chosen (majority voting). It can be weighted, where some models have more influence.
* **Stacking**: Uses another model (meta-model) to learn how to best combine the predictions of the base models.

**Common Ensemble Learning Techniques**

1. **Bagging (Bootstrap Aggregating)**:

* In bagging, multiple instances of the same learning algorithm are trained on different subsets of the training data, which are generated by random sampling with replacement (bootstrapping).
* The predictions of these models are then aggregated, typically by averaging (regression) or majority voting (classification).
* **Example**: Random Forest, which combines multiple decision trees.

1. **Boosting**:

* Boosting focuses on training models sequentially, where each subsequent model tries to correct the errors of the previous ones.
* It assigns higher weights to the data points that were misclassified or poorly predicted by previous models, forcing the model to focus on these harder cases.
* **Examples**: AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost.

1. **Stacking**:

* In stacking, multiple models (level-0 models) are trained on the training data, and a meta-model (level-1 model) is trained on the outputs of the level-0 models to make the final prediction.
* This method allows leveraging the strengths of different models by learning how to best combine them.

1. **Voting**:

* Voting ensembles are used primarily in classification. They combine predictions from multiple models by taking a majority vote (for discrete classes) or averaging probabilities (for probabilistic outputs).
* There are two main types: **Hard Voting** (majority voting) and **Soft Voting** (averaging probabilities).

**Benefits of Ensemble Learning:**

* Ensemble methods often achieve higher accuracy than individual models because they combine the strengths of multiple models and mitigate their weaknesses.
* By averaging the predictions of several models, ensembles can reduce the risk of overfitting, especially when using techniques like bagging.
* Ensembles can be more robust to outliers and noisy data, as they aggregate the decisions of multiple models.
* Different types of models can be combined in an ensemble, allowing for creative and powerful model architectures.

**Drawbacks of Ensemble Learning:**

* Ensembles can be more complex and harder to interpret as compared to individual models.
* Training multiple models can require more computational resources and time.
* Properly tuning and combining multiple models can be challenging.

**50.11 What is gradient descent and how does it work?**

Gradient Descent is an optimisation algorithm used in machine learning and deep learning to minimise the cost function, which measures how far off a model's predictions are from the actual values. It's a method of finding the best parameters (like weights in a neural network) that reduce the error in predictions.

**Gradient Descent Working:**

**1. Initializing Parameters**: Parameters (weights and biases) of the model are initialised randomly or with some heuristics.

**2. Cost Function computation**: The cost function quantifies the difference between the predicted values and the actual values. Common cost functions include Mean Squared Error (MSE) for regression problems and Cross-Entropy Loss for classification.

**3. Gradient computation**: The gradient of the cost function with respect to the parameters is computed. The gradient is a vector of partial derivatives, which tells us the direction in which the cost function increases the most. The idea is to update the parameters in the opposite direction of the gradient to reduce the cost.

**4. Updating Parameters**: Parameters are updated by moving in the opposite direction of the gradient by a certain amount. This amount is controlled by a factor called the learning rate () .

Here, represents the parameters (weights and biases), is the learning rate, is the gradient of the cost function.

**5. Repetition**: The process is repeated iteratively—compute the cost, calculate the gradient, and update the parameters—until the cost function is minimized to a satisfactory level or until the changes in the cost function are below a threshold.

**Types of Gradient Descent:**

**1. Batch Gradient Descent:** Uses the entire dataset to compute the gradient at every step. It can be slow for large datasets.

**2. Stochastic Gradient Descent (SGD):** Uses a single data point (or a small batch) to compute the gradient, which makes it faster and allows for more frequent updates. However, it introduces more noise in the parameter updates.

**3. Mini-Batch Gradient Descent:** Uses compromise between batch and stochastic gradient descent. It uses a small batch of data to compute the gradient, balancing efficiency and stability.

**Challenges with Gradient Descent:**

1. **Choosing the Right Learning Rate**: A learning rate that is too high can cause the algorithm to overshoot the minimum, while a learning rate that is too low can make convergence very slow.

**2. Local Minima**: For complex functions, the algorithm might get stuck in local minima or saddle points, which are not the global minimum.

**3. Feature Scaling**: Gradient descent can converge faster if features are on similar scales, which often requires preprocessing steps like normalization or standardization.

**50.12 Describe the difference Between Batch Gradient Descent and Stochastic Gradient Descent.**

|  |  |
| --- | --- |
| **Batch Gradient Descent** | **Stochastic Gradient Descent (SGD)** |
| Uses the entire dataset to compute the gradient of the cost function in each iteration. | Uses the entire dataset to compute the gradient of the cost function in each iteration. |
| Parameters are updated only once per iteration after the gradient for the entire dataset is calculated. | Parameters are updated after computing the gradient from just one data point, leading to more frequent updates. |
| More stable and typically converges smoothly towards the minimum but can be slow, especially with large datasets. | Faster updates but introduces more noise in the gradient calculation, which can cause the optimization path to fluctuate, potentially leading to faster but less stable convergence. |
| Requires significant memory to compute gradients because it processes the whole dataset at once. | More memory-efficient since it processes one or a few data points at a time, making it suitable for large datasets. |

**50.13 What is the curse of dimensionality in machine learning?**

The curse of dimensionality refers to the various phenomena that arise when analysing and organizing data in high-dimensional spaces that do not occur in lower-dimensional spaces. It highlights the challenges that arise as the number of features (dimensions) increases.

**Key Aspects of the Curse of Dimensionality:**

1. **Increased Computational Complexity**: As the number of dimensions increases, the amount of data needed to generalize accurately grows exponentially. This means more computational resources are required for tasks like distance calculations, model training, and optimization.
2. **Sparsity of Data**: In high-dimensional spaces, data points tend to become sparse. Most of the data resides far from each other, making it harder to detect patterns, clusters, or correlations. This sparsity can lead to poor model performance because the model struggles to find meaningful relationships in the data.
3. **Overfitting**: With more dimensions, models tend to have more parameters. While this can allow the model to capture more complex patterns, it also increases the risk of overfitting, where the model learns noise in the training data rather than the actual underlying distribution.
4. **Distance Measures Become Less Informative**: Many machine learning algorithms rely on distance measures (e.g., Euclidean distance) to evaluate similarity between data points. In high-dimensional spaces, the difference between the nearest and farthest points decreases, making these measures less reliable.
5. **Visualization Challenges**: Visualizing data and understanding relationships between features becomes extremely difficult as dimensions increase beyond 3D or 4D, limiting the ability to gain insights from the data directly.

**Mitigating the Curse of Dimensionality:**

* **Dimensionality Reduction**: Techniques like Principal Component Analysis (PCA), t-SNE, and Linear Discriminant Analysis (LDA) can be used to reduce the number of features while retaining as much of the data's variance as possible.
* **Feature Selection**: Choosing the most relevant features based on statistical methods, domain knowledge, or regularization techniques can help reduce the dimensionality and focus on the most important aspects of the data.
* **Regularization**: Adding regularization terms to machine learning models helps prevent overfitting by penalizing complexity, making the models more robust in high-dimensional spaces.

**50.14 Explain the difference between L1 and L2 regularization.**

L1 and L2 regularization are techniques used to prevent overfitting in machine learning models by adding a penalty to the cost function based on the magnitude of the model parameters (weights).

|  |  |
| --- | --- |
| **L1 Regularization (Lasso Regularization)** | **L2 Regularization (Ridge Regularization)** |
| L1 regularization adds the sum of the absolute values of the coefficients (weights) to the loss function.  The modified cost function with L1 regularization is:  Here, λ is the regularization parameter that controls the strength of the penalty, and wi represents the weights. | L2 regularization adds the sum of the squared values of the coefficients (weights) to the loss function.  The modified cost function with L2 regularization is:  Here, λ is the regularization parameter, and wi represents the weights. |
| L1 regularization tends to drive some weights to exactly zero, effectively performing feature selection. This means it can create sparse models where only a subset of features has a non-zero coefficient. | L2 regularization tends to shrink the weights but does not typically drive them to exactly zero. Instead, it makes all weights smaller and more evenly distributed. It results in non-sparse models. |
| L1 regularization can automatically perform feature selection by driving irrelevant feature weights to zero. It is is used when you want a simpler model that potentially ignores some features. | L2 regularization reduces the impact of irrelevant features but typically does not eliminate them entirely. It is used when you want to keep all features but ensure that none of them have disproportionately large weights. |
| L1 regularization involves optimization that can be more complex due to the non-differentiability at zero, but it leads to feature selection. | L2 regularization is easier to optimize because the penalty term is differentiable, making gradient-based methods more straightforward. |

**50.15 What is a confusion matrix and how is it used?**

A **confusion matrix** is a table used to evaluate the performance of a classification algorithm. It helps to understand how well the model is performing by comparing the predicted classes with the actual classes. The matrix is particularly useful in binary classification, but it can also be extended to multi-class classification problems.

**Structure of a Confusion Matrix**

In a binary classification scenario, the confusion matrix is a 2x2 table that looks like this:

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

* **True Positive (TP):** The number of cases where the model correctly predicted the positive class.
* **False Positive (FP):** The number of cases where the model incorrectly predicted the positive class (Type I error).
* **True Negative (TN):** The number of cases where the model correctly predicted the negative class.
* **False Negative (FN):** The number of cases where the model incorrectly predicted the negative class (Type II error).

**Key Metrics Derived from a Confusion Matrix**

1. **Accuracy:** The proportion of correctly classified instances (both positives and negatives).
2. **Precision:** The proportion of correctly predicted positive observations to the total predicted positives.
3. **Recall (Sensitivity or True Positive Rate):** The proportion of correctly predicted positive observations to all observations in the actual positive class.
4. **F1 Score:** The harmonic mean of Precision and Recall, providing a balance between them.
5. **Specificity (True Negative Rate):** The proportion of correctly predicted negative observations to all observations in the actual negative class.

**Use of a Confusion Matrix**

* **Identify Misclassifications:** By analysing the FP and FN counts, you can determine where the model is making errors.
* **Evaluate Model Performance:** It allows you to assess whether the model is biased towards a particular class (e.g., high precision but low recall).
* **Compare Models:** By calculating metrics like accuracy, precision, and recall from the confusion matrix, you can compare the performance of different models.

**50.16 Define AUC-ROC curve.**

The AUC-ROC (Area Under the Curve-Receiver Operating Characteristic) curve is a performance measurement for classification models at various threshold settings.

**Components of AUC-ROC**

1. **ROC Curve:**

* The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
* It plots **TPR** against **FPR** at different threshold levels.

1. **TPR**: The proportion of actual positives correctly identified by the model.
2. **FPR**: The proportion of actual negatives that are incorrectly identified as positives by the model.

* The ROC curve starts at (0, 0) and ends at (1, 1). The point (0, 1) represents a perfect classifier, meaning TPR is 1 (all positives are correctly identified) and FPR is 0 (no negatives are incorrectly identified).

1. **AUC (Area Under the Curve):**

* The AUC represents the area under the ROC curve. It quantifies the overall ability of the model to discriminate between positive and negative classes.
* The AUC value ranges between 0 and 1:

1. **AUC = 1:** Perfect classifier.
2. **AUC = 0.5:** The model has no discriminative power, equivalent to random guessing.
3. **AUC < 0.5:** The model is performing worse than random guessing, meaning it is consistently misclassifying the positive and negative classes.

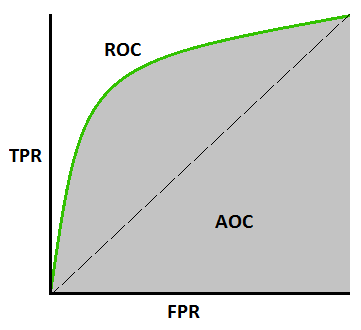
**Interpretation of AUC-ROC:**

* **High AUC (> 0.9):** Excellent model performance.
* **Moderate AUC (0.7 to 0.9):** Good to moderate model performance.
* **Low AUC (0.5 to 0.7):** Poor model performance.

**Use of AUC-ROC Curve:**

The AUC-ROC curve is particularly useful in situations where:

* You need to compare the performance of multiple models.
* You need to assess the trade-off between sensitivity (TPR) and specificity (1 - FPR) across different threshold values.
* The classes are imbalanced, and accuracy alone may be misleading.



**50.17 Explain the k-nearest neighbours algorithm.**

* The k-Nearest Neighbours algorithm is a non-parametric and instance-based machine learning method, meaning it makes decisions based on the closest data points in the training set without making any assumptions about the underlying data distribution. k-NN is commonly used in areas such as Pattern recognition, Recommendation systems, Anomaly detection, and Image and text classification.
* The choice of k is crucial to the performance of the k-NN algorithm. For small k, the model may be sensitive to noise and may lead to overfitting. For large k, the model may become too smooth and may underfit, losing the ability to capture the local structure of the data. A common approach is to use cross-validation to find the optimal value of k.
* k-NN is a lazy learner, meaning it doesn’t build an explicit model during the training phase. Instead, it stores the training data and makes decisions during the prediction phase. Since it doesn’t require training, k-NN can be directly applied to the data, but it can be computationally expensive for large datasets. The algorithm is sensitive to the scale of the data. Feature scaling (normalization or standardization) is important to ensure that all features contribute equally to the distance calculation.

**How k-Nearest Neighbours Works:**

1. Each data point in the dataset is represented as a point in an n-dimensional space, where n is the number of features.
2. When a new data point (test instance) needs to be classified or predicted, the algorithm calculates the distance between this point and all the points in the training dataset. Common distance metrics include Euclidean distance, Manhattan distance and Minkowski distance.
3. The algorithm then identifies the k closest data points (neighbours) to the new data point based on the calculated distances.
4. Averaging is done after neighbour identification. In case of classification, the class label of the new data point is determined by a majority vote among the k nearest neighbours. The most common class among the neighbours is assigned as the prediction. In case of regression, the prediction is made by averaging the values of the k nearest neighbours.
5. The final output is either the class label (for classification) or a predicted value (for regression).

**50.18 Explain the basic concept of a Support Vector Machine (SVM).**

A Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm used primarily for classification tasks, though it can also be adapted for regression. The key idea behind SVM is to find the optimal boundary (or hyperplane) that best separates data points of different classes.

**Basic Concepts of SVM:**

1. **Hyperplane:** In an SVM, a hyperplane is a decision boundary that separates the data into different classes. In two-dimensional space, the hyperplane is simply a line. In higher dimensions, it becomes a plane or hyperplane. The goal of the SVM algorithm is to find the hyperplane that maximizes the margin between the two classes.
2. **Margin:** The margin is the distance between the hyperplane and the nearest data points from each class. These nearest points are known as support vectors.The optimal hyperplane is the one that has the largest margin, meaning it is as far away as possible from the nearest data points of any class. This maximizes the classifier's ability to generalize to unseen data.
3. **Support Vectors:** Support vectors are the data points that lie closest to the hyperplane and directly influence its position and orientation. These points are crucial because they define the margin, and the optimal hyperplane is determined based on them.
4. **Linear SVM:** When the data is linearly separable, SVM finds a straight line (in 2D) or hyperplane (in higher dimensions) that separates the classes with the maximum margin.
5. **Non-Linear SVM:** When the data is not linearly separable, SVM uses a technique called the kernel trick. The kernel trick maps the original data into a higher-dimensional space where a linear hyperplane can be used to separate the classes. Common kernels include Linear Kernel, Polynomial Kernel, Radial Basis Function Kernel, Sigmoid Kernel

**How SVM Works**

1. **Training Phase:** SVM takes the labelled training data and finds the optimal hyperplane that maximizes the margin between the classes. If the data is not linearly separable, the kernel trick is applied to map the data into a higher-dimensional space where it becomes separable.
2. **Prediction Phase:** Once the optimal hyperplane is found, the SVM can predict the class of new, unseen data points by determining on which side of the hyperplane they fall.

**50.19 How does the kernel trick work in SVM?**

The kernel trick is a technique used in Support Vector Machines (SVM) to handle non-linearly separable data by implicitly mapping the input features into a higher-dimensional space. In this higher-dimensional space, the data can often become linearly separable, allowing the SVM to find an optimal hyperplane for classification. Here's how it works:

**1. Feature Mapping:**

* The kernel trick avoids the direct computation of the mapping from the original feature space X to a higher-dimensional space H, which could be computationally expensive or even infeasible.
* Instead of explicitly calculating this mapping, the kernel function k(x,x′) computes the dot product between the images of two points x and 'x′ in the higher-dimensional space H

, where ϕ(x) is the mapping from the original space X to the higher-dimensional space H.

**2. Linear Classification in Higher-Dimensional Space:**

* In SVM, the goal is to find a hyperplane that maximizes the margin between two classes. For non-linear data in the original space, finding such a hyperplane is difficult or impossible.
* However, by using the kernel function, the SVM can operate as if it were in the higher-dimensional space H, even though the data remains in the original space X.
* The SVM solves the optimization problem using the kernel function to compute the inner products without ever needing to explicitly compute the coordinates of the points in H.

**4. Advantages of the Kernel Trick:**

* **Efficiency**: The kernel trick allows SVMs to work efficiently in very high-dimensional spaces without needing to explicitly compute the high-dimensional mapping.
* **Flexibility**: Different kernel functions can be chosen based on the problem at hand, allowing SVMs to handle a variety of data types and structures.

**50.20 What are the different types of kernels used in SVM and when would you use each?**

In Support Vector Machines (SVM), different types of kernel functions are used depending on the nature of the data and the problem at hand. Here’s a breakdown of the most common kernels and when they might be used:

**1. Linear Kernel**

* **Function**:
* **Usage**:
  + **When data is linearly separable**: If the data is linearly separable in the original feature space or nearly so, the linear kernel is often the best choice.
  + **High-dimensional data**: For high-dimensional data where the number of features is greater than the number of samples, the linear kernel is often effective.
  + **Interpretability**: The linear kernel is also preferred when interpretability of the model is important because it produces a linear decision boundary.
* **Example Applications**: Text classification (e.g., spam detection), document categorization, and scenarios where the data is sparse and high-dimensional.

**2. Polynomial Kernel**

* **Function**: ,

where c is a constant, and d is the degree of the polynomial.

* **Usage**:
  + **Non-linear data**: When the relationship between classes is non-linear, and you want to capture interactions between features, a polynomial kernel might be appropriate.
  + **Degree d**: The degree d controls the flexibility of the decision boundary; a higher degree allows more complex boundaries, but may also increase the risk of overfitting.
  + **Feature interactions**: Useful when you believe there are interactions between features that are important for classification.
* **Example Applications**: Image processing, where pixel interactions are important, or in problems where feature interactions are key to capturing the relationship between data points.

**3. Radial Basis Function (RBF) Kernel / Gaussian Kernel**

* **Function**:

Where σ controls the width of the Gaussian function.

* **Usage**:
  + **Complex non-linear relationships**: The RBF kernel is a popular choice when there is no prior knowledge about the data, as it can model highly non-linear relationships.
  + **Default kernel**: Often used as the default kernel in SVMs because it can handle various data distributions and is less sensitive to the scale of the data.
  + **Data not linearly separable**: Particularly effective when the data is not linearly separable in the original feature space.
  + **Parameter tuning**: Requires careful tuning of the σ parameter (also referred to as the kernel width or gamma), as it controls the decision boundary's smoothness.
* **Example Applications**: Handwriting recognition, image classification, and other cases where the decision boundary is not linear.

**4. Sigmoid Kernel**

* **Function**:

Where α and c are kernel parameters.

* **Usage**:
  + **Neural network-like behavior**: The sigmoid kernel mimics the behavior of a two-layer, perceptron-based neural network, which makes it suitable for certain types of pattern recognition tasks.
  + **Binary classification**: Sometimes used in binary classification tasks where the data might be mapped in a way that resembles neural network classification.
  + **Not widely used**: Less common than the linear, polynomial, or RBF kernels because it can behave unpredictably depending on the parameters and data.
* **Example Applications**: Some specific use cases in binary classification where non-linear relationships are suspected, but the problem space isn't well understood.

**5. Custom Kernels**

* **Function**: Any kernel function that satisfies Mercer’s theorem can be used, allowing for domain-specific custom kernels.
* **Usage**:
  + **Domain-specific problems**: When the problem at hand has unique characteristics that standard kernels cannot capture, a custom kernel can be designed.
  + **Prior knowledge**: Custom kernels are particularly useful when there is strong domain knowledge about the data's structure.
* **Example Applications**: Bioinformatics (e.g., string kernels for DNA sequences), graph kernels in network analysis, and other specialized domains.

**Choosing the Right Kernel**

* **Start Simple**: Often, you start with a linear kernel because it is simple and fast. If it performs well, there is no need for a more complex kernel.
* **Experimentation**: If the linear kernel does not perform well, try the RBF kernel as it is flexible and widely applicable.
* **Model Complexity vs. Overfitting**: Be cautious with polynomial and RBF kernels as they introduce more complexity, which could lead to overfitting, especially with small datasets.
* **Cross-validation**: Always use cross-validation to tune kernel parameters and evaluate different kernels to find the best fit for your data.

**50.21 What is the hyperplane in SVM and how is it determined?**

In SVM, the hyperplane is the decision boundary that separates different classes in the feature space. It’s a flat affine subspace that divides the data points into two categories. In an nnn-dimensional space, the hyperplane is an n−1n-1n−1 dimensional entity (e.g., a line in 2D, a plane in 3D).

The optimal hyperplane is determined by maximizing the margin, which is the distance between the hyperplane and the closest data points from each class, known as support vectors. SVM finds this hyperplane by solving an optimization problem that minimizes the norm of the weight vector while ensuring that all data points are correctly classified with the maximum margin. If the data isn’t linearly separable, SVM can use the kernel trick to map it into a higher-dimensional space where a linear hyperplane can effectively separate the classes.

**50.22 What are the pros and cons of using a Support Vector Machine (SVM)?**

**Pros of SVM**:

* **Effective in High-Dimensional Spaces**: SVM is particularly powerful when dealing with high-dimensional data and can be effective even when the number of dimensions exceeds the number of samples.
* **Versatile with Kernels**: SVM can handle non-linear data by using the kernel trick, allowing it to find complex decision boundaries. This makes it adaptable to various types of data.
* **Robustness**: SVMs are relatively resistant to overfitting, especially when the correct regularization parameters are used, and they focus on the most relevant data points (support vectors).

**Cons of SVM**:

* **Computationally Intensive**: SVMs can be slow to train, especially with large datasets, because the algorithm requires solving a complex quadratic programming problem.
* **Sensitive to Parameter Tuning**: SVM performance heavily depends on the choice of kernel, regularization parameter CCC, and kernel-specific parameters like γ\gammaγ for the RBF kernel, making it tricky to optimize.
* **Less Interpretable**: The resulting model, especially with non-linear kernels, can be less interpretable compared to simpler models like linear regression.

**50.23 Explain the difference between a hard margin and a soft margin SVM.**

The difference between hard margin and soft margin SVMs lies in how they handle data that is not perfectly separable.

**Hard Margin SVM**:

* **Strict Separation**: A hard margin SVM seeks a hyperplane that perfectly separates the data points of different classes without allowing any misclassification.
* **Applicability**: This approach works well when the data is linearly separable and there are no outliers.
* **Limitations**: It’s very sensitive to outliers and noise. Even a single outlier can make it impossible to find a hyperplane that perfectly separates the classes.

**Soft Margin SVM**:

* **Relaxed Separation**: A soft margin SVM allows for some misclassification by introducing a margin of tolerance. It aims to find a balance between maximizing the margin and minimizing classification errors.
* **Applicability**: This approach is used when the data is not perfectly separable or when some noise or outliers are present. It’s more flexible and robust compared to the hard margin SVM.
* **Parameter CCC**: The regularization parameter CCC controls the trade-off between a larger margin and fewer misclassifications. A smaller CCC allows a wider margin but more misclassifications, while a larger CCC enforces stricter classification at the expense of a narrower margin.

**50.24 Describe the process of constructing a decision tree.**

Constructing a decision tree involves recursively splitting the dataset into subsets based on feature values to create a tree-like model of decisions. Here’s the process:

1. **Start with the Entire Dataset**:
   * Begin with the full dataset as the root node of the tree. Each node in the tree represents a subset of the data.
2. **Select the Best Feature to Split**:
   * At each node, select the feature that best separates the data into different classes. The selection is based on criteria like **Gini impurity**, **information gain** (using entropy), or **variance reduction** for regression tasks.
   * The best feature is the one that maximizes the separation of classes in the resulting subsets.
3. **Split the Data**:
   * Divide the dataset into subsets based on the selected feature’s values. For categorical features, this could mean splitting on each category; for continuous features, it might involve choosing a threshold.
4. **Create Child Nodes**:
   * Each subset forms a new child node of the tree, and the process is recursively repeated for each child node.
5. **Repeat Until Stopping Criteria are Met**:
   * The process continues until a stopping criterion is met, such as a maximum tree depth, a minimum number of samples per leaf, or no further improvement in classification.
   * Leaves are created when no further splits are possible or necessary, and they represent the final decision (classification or regression output).

**50.25 Describe the working principle of a decision tree.**

A decision tree works by recursively dividing the dataset into subsets based on feature values, with the goal of creating a model that makes predictions or classifications by following a series of decisions. Here’s the working principle:

1. **Feature Selection**:
   * At each node, the decision tree selects the feature that best separates the data according to a chosen criterion, such as **Gini impurity**, **information gain**, or **variance reduction**. This determines how the data will be split.
2. **Data Splitting**:
   * Based on the selected feature, the data is split into subsets. For categorical features, each category forms a branch; for continuous features, the split is based on a threshold value. This creates child nodes from the current node.
3. **Recursive Splitting**:
   * The process is applied recursively to each child node. Each subset of data is used to create new nodes and further splits until a stopping criterion is met, such as reaching a maximum tree depth, having a minimum number of samples per node, or achieving pure nodes where all data points belong to the same class.
4. **Decision Making**:
   * Once the tree is fully grown, making a prediction involves traversing the tree from the root to a leaf. Each internal node represents a decision based on feature values, and each leaf node represents the final predicted outcome or class label.

The resulting tree is a model that can be used for classification or regression tasks, where the path from the root to a leaf defines the decision rules based on the input features.

**50.26 What is information gain and how is it used in decision trees?**

Information gain is a metric used in decision trees to determine the best feature to split the data at each node. It measures how much uncertainty in the dataset is reduced by splitting based on a particular feature. Here’s how it works:

1. **Entropy Calculation**:
   * Entropy quantifies the amount of uncertainty or disorder in a dataset. For a given set of data, entropy is calculated as:

where pi is the proportion of instances belonging to class i in set S, and k is the number of classes.

1. **Information Gain Computation**:
   * **Information Gain** is calculated as the difference between the entropy of the original dataset and the weighted average entropy of the subsets created by splitting on a feature. It is given by:

where A is the feature being evaluated, Values(A) are the possible values of feature A, Sv is the subset of data where feature A has value v, and ∣Sv∣/∣S∣ is the proportion of instances in Sv.

1. **Feature Selection**:
   * The feature with the highest information gain is chosen for the split because it provides the most significant reduction in uncertainty about the class labels. This means it creates subsets that are more homogeneous with respect to the target variable.

**50.27 Explain Gini impurity and its role in decision trees.**

Gini impurity is a metric used in decision trees to evaluate the quality of a split at each node. It measures the degree of impurity or disorder in a dataset and helps in selecting the best feature for splitting. Here’s how it works:

1. **Calculation of Gini Impurity**:
   * The Gini impurity of a dataset is calculated as:

where pi is the proportion of instances belonging to class i in the set S, and k is the number of classes. The value ranges from 0 (perfectly pure) to 0.5 (completely impure for binary classification).

1. **Impurity Reduction**:
   * To determine the best feature to split on, the decision tree calculates the Gini impurity for each possible split. For a feature that splits the data into subsets, the Gini impurity of each subset is weighted by its proportion of the total dataset:

Where A is the feature being evaluated, Sy is the subset where feature A has value v, and ∣Sv∣/∣S∣ is the proportion of instances in Sv​.

1. **Feature Selection**:
   * The feature with the lowest Gini impurity after the split is chosen because it results in the greatest reduction in impurity, leading to subsets that are more homogenous with respect to the target variable.