Union and Intersection of Two Lists

Input:Two lists, L1 and L2
Output:The union and intersection of L1 and L2
1. Start
2. Read the number of elements in L1 and L2.
3. Read the elements in L1 and L2.
4. For union:
- For each element in L1, if the element is not in the union list, append it.
- For each element in L2, if the element is not in the union list, append it.
5. For intersection:
- For each element in L1, if the element is also in L2, append it to the intersection list.
6. Print "After Union:" and display the union list.
7. Print "After Intersection:" and display the intersection list.

k. Means
1)start
2) Import Libraries
3) from sklearn.cluster import K-Means.
4) model= kmeans (n cluster = 3).
5) model. fit (x)
6) kmeans (n.. cluster = 3)
(7) Calculate Euclidean distance b/w each data point and all centroide
7.1D Assign each datapoints to cluster with nearest centroid
8) Recalculate centroids by taking mean of all points in each cluster
(9) cluster labels for each data points
10) Plot data using cluster labels.
11) Plot final centroid
12) stop

Occurrences(string)
1) Start
2) Read the sentence in `st` and split it into words.
3) Initialize a list `l` with the words from `st`.
4) For each word `i` in `l`, do:
- Print "count of `i` is", `st.count(i)`
5) Stop

ID3
1)start
2) Calculate the entropy for one entire table
3) Calculate the information gain for each attributes
4) set one node as one with maximumm inform gain
5) Calculate the new entropy
6) Repeat the process complete classification been done
7Print the deusion true
8) stop
Result

Single Variable Linear Regression

Input: A CSV file with training data (contains one independent variable and one dependent variable).
Output: Trained linear regression model and accuracy score.
1. Start
2. Import necessary libraries: pandas, numpy, and sklearn.
3. Load the CSV file data into a DataFrame using pandas.
4. Convert the DataFrame columns to numpy arrays for the features (X) and target (y) variables.
5. Perform train-test split to create X_train, X_test, y_train, and y_test using train_test_split from sklearn.
6. Create a linear regression model object.
7. Fit the model using model.fit(X_train, y_train).
8. Calculate the accuracy of the model on the test data using model.score(X_test, y_test).
9. Stop

Matrix Multiplication
Input: Two matrices
Output: Their product matrix
1. Start
2. Enter two matrices, say mat1 and mat2.
3. Initialize an empty array, say mat3.
4. Compute the matrix multiplication of mat1 and mat2. Store the result in mat3.
5. Display mat3.
6. End

Frequently Occurring Word
Input: Any file
Output: The most frequently occurring word
1. Start
2. Get file name from user and open the file.
3. Initialize man_val = 0 and man_word = "".
4. Read content of the file and store it in data.
5. For each word i in data, do:
6. If data.count(i) > man_val, then
- Set man_val = data.count(i)
- Set man_word = i
6. Display man_word and man_val (the most frequent word and its frequency).
7. End

MULTI VARIABLE REGRESSION
1)start
2) import pandas, numpy and sklearn
3) df = pandas. read _csv (csv_file)
4) Convert df into numpy array.
5) Using train_test_split, split the dataset
6)create linear regression object as model
7) model. fit()
8) The accuracy wrt train and test data can be viewed. using test model.score()
9) Stop

Polynomial Regression

1)Start
2) import pandas, numpy, matplotlib and sklearn
3)df =pandas. read_csv(csv_file)
4) Convert the dataframe Into numpy array
5) Use train_test_split to create dataset.
(6) Poly_feature = sklearn. preprocessing. Polynomial Features (degree)
7) X_Poly = poly_features. fit _ transform (x).
8) model = Linear Regression ( )
9) poly_model = make _ pipeline (poly_features, model)
10) poly-model. fit (X_poly, y)
11) The accuracy can be estimated using poly_model.score( )
12) Stop

ANN
1) Start
2) Import dataset into data.
3) Import necessary libraries.
4) Load data columns to X.
5) Load target column to y.
6) Split dataset into X_train, X_test, y_train, y_test.
7) Initialize learning rate, iterations, hidden layer, input layer, and output layer.
8) Initialize weights of hidden layers (e.g., w1, w2).
9) For i in iterations do:
9.1) Compute dot product of X_train and w1.
9.2) A1 = sigmoid (alpha).
9.3) E1 = A1 - y_train.
9.4) dw1 = E1 * A1 * (1 - A1).
9.5) E2 = dot product of dw1, w2, and y.
9.6) dw2 = E2 * A1 * (1 - A1).
9.7) Update the weights.
10) End for loop.
.

SVM
1) Start
2) Import necessary libraries
3) Import SVC from sklearn.svm
4) Import train_test_split from sklearn.model_selection
5) Split the data into training and testing sets using train_test_split
6) Create an SVM classifier using sklearn.svm.SVC()
7) Fit the SVM classifier using svm.fit(X_train, y_train)
8) Predict labels for the test data
9) Evaluate the accuracy of the classifier
10) Stop

NB
1) Start
2) Import required libraries: pandas, numpy, and sklearn
3) Read the dataset from CSV using **`pd.read_csv()`** from pandas
4) Convert the DataFrame to a numpy array
5) Use train_test_split to split the dataset into training and testing sets
6) Initialize the Naive Bayes model using sklearn.naive_bayes.GaussianNB()
7) Fit the model using model.fit(X_train, y_train)
8) To see the accuracy, use model.score(X_test, y_test) on the test data. Also, calculate and print precision and recall
9) Stop

TEMP
1) Start
2) Read the temperature in Celsius (C)
3) Calculate Fahrenheit using the formula: F = (9/5) * C + 32
4) Display the temperature in Fahrenheit (F)
5) Stop

Reversing the Digits of a Number
1) Start
2) Read a number in n
3) Initialize reversed number as rev = 0
4) While n is not equal to 0, do:
- Extract the last digit using digit = n % 10
- Update rev = rev * 10 + digit
- Remove the last digit from n using n = n // 10
5) Print rev
6) Stop

Sum of digits is even
1)Start
2)For each number `i` from 100 to 200:

- Calculate the sum of digits of `i`.
- If the sum is even, print the number `i`.

3)Stop

Calculator
1)Start
2)Display Menu
3)Perform the selected operation:

- If Addition, print the sum.
- If Subtraction, print the difference.
- If Multiplication, print the product.
- If Division, check for zero and print the quotient or an error message.

4)Repeat from step 2 until the user chooses to exit.
5)Stop.

Reversing a Int

1) Start
2) Read the number `x`.
3) Initialize `b = 0`.
4) While `x` is not 0:
- Get the last digit `c = x % 10`.
- Update `x = x // 10`.
- Update `b = b * 10 + c`.
5) Print the reversed number `b`.
6) Stop

Fibonacci Sequence

1) Start
2) Read the number n.
3) Initialize two variables: a = 1 and b = 2.
4) For i from 1 to n:
- Print the value of a.
- Update the values: a = b and b = a + b.
5) Stop

Geometric mean
1) Start
2) Read the number `x`.
3) Define a function `geoMean(x)` to calculate the geometric mean:
- Initialize `y = 1`.
- For each number `i` from 1 to `x`, multiply `y` by `i`.
- After the loop, calculate the geometric mean using `v = y^(1/x)` (where `v` is the result).
4) Print the geometric mean of the first `x` numbers.
5) Stop

Greatest of Three Numbers
1) Start
2) Read three numbers as a,b,c
3) Compare the numbers:
- If `a` is greater than both `b` and `c`, print `a`.
- If `b` is greater than both `a` and `c`, print `b`.
- Else, print `c`.
4) Stop

Multiplication Table**

1) Start
2) Read the number `x` from the user.
3) For `i` from 1 to 12, do the following:
- Print the multiplication of `x` and `i` (i.e., `x * i`).
4) Stop

Palindrome
1) Start
2) Read the number `x`.
3) Reverse the digits of `x`.
4) If the original number is equal to the reversed number, print "The number is a palindrome."
5) Otherwise, print "The number is not a palindrome."
6) Stop

Divisibility by 5 and 7
1) Start
2) Read the number `x`.
3) If `x` is divisible by both 5 and 7, print "The number is divisible by both 5 and 7".
4) Else if `x` is divisible by 5, print "The number is divisible by 5".
5) Else if `x` is divisible by 7, print "The number is divisible by 7".
6) Otherwise, print "The number is not divisible by both 5 and 7".
7) Stop