EX.NO:01-Simulating a simple calculator

AIM: To generate a simple program for simple calculator without using library function.

Algorithm:

1. Start the program.
2. Get input from the user(Choice corresponding to various operations like addition,multiplication,division).
3. Get two integer inputs from the user.
4. Perform the operations based on the user's choice.
5. Print the output.
6. End the program.

Program:

```
print("Operation: +, -, *, /")
select = input("Select operations: ")

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if select == "+":
print(num1, "+", num2, "=", num1+num2)
elif select == "-":
print(num1, "-", num2, "=", num1-num2)
elif select == "*":
print(num1, "*", num2, "=", num1*num2)
elif select == "-":
print(num1, "/", num2, "=", num1/num2)
else:
print ("Invalid input")
```

Output:

Operation: +, -, *. /
Select Operations: *
Enter First Number: 10
Enter Second Number: 2
10.0*2.0=20.0

Result: Simple calculator program by using python is successfully generated.

EX.NO:02-Armstrong Series

AIM: To generate a program to check whether a number is Armstrong (or) not.

Algorithm:

1. Start the program.
2. Declare variables sum, temp1, num.
3. Read num from users.
4. Initialize variable sum=0 & temp=num.
5. Repeat until>=0
   i) Sum=Sum+Cube of last digit[(num%10)*(num%10)*(num%10)]
   ii)num=num/10
6. If Sum==temp
   print "Armstrong Number"
   else
   print "Not Armstrong number"
7. Stop the program.

Program:

```
lower = int(input("Enter the lower range : "))
upper = int(input("Enter the upper range : "))
for num in range(lower, upper + 1):
order = len(str(num))
sum = 0
temp = num
while temp > 0:
digit = temp%10
sum+=digit**order
temp//=10
if num == sum:
print (num)
```

Output:

```
Enter the Lower range: 100
Enter the upper range: 500
153
370
371
407
```

Result: To generate a program to check weather a number is Armstrong or not is verified successfully.

EX.NO:03-FIBONACCI SERIES

AIM: To generate a program to find the Fibonacci Series.

Algorithm:

1. Start the program.
2. Input the number of values we want to generate the Fabonacci series.
3. Initialize the count=0, n-1=0 & n-2=1.
4. If the n-terms <=0
5. Print “error” as it is not a valid number for series.
6. If n terms=1, it will print n-1 values.
7. While count <n-terms.
8. Print(n-1).
9. nth=n-1+n-2
10. We will update the variable, n-1, n-2, n-3=nth and soon, up to the required term.
11. Stop the Program.

Program:

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
if nterms <= 0:
print("Please enter a positive integer")
elif nterms == 1:
print("Fibonacci sequence upto",nterms,":")
print(n1)
else:
print("Fibonacci sequence:")
while count < nterms:
print(n1)
```

```
nth = n1 + n2
 n1 = n2
n2 = nth
count += 1
```

Output:

```
How many terms? 5
Fibonacci sequence:
0
1
1
2
3
```

Result: To generate a program to find the fibonacci series is implemented successfully.

EX.NO:04-MODULES AND FUNCTIONS

AIM: Creating function and importing those functions as modules.

Algorithm:

1. Start the program.
2. Get input from the user(ns integer).
3. Do the operations like summation,Multiplication, and divide.
4. Then print the output.
5. Stop the program.

Program:

```
def summation(a,b):
return a+b
def multiplication(a,b):
return a*b
def divide(a,b):
return a/b
a = int(input("Enter the first number"))
b = int(input("Enter the second number"))
print("Sum = ",summation(a,b))
print("Product = ",multiplication(a,b))
print("Divisor = ",divide(a,b))
```

Output:

```
Enter the first number 5
Enter the second number 10
Sum = 15
Product = 50
Divisor = 0.5
```

Result: Importing those functions as modules are implemented successfully.

EX.NO:05(A)-WORKING WITH STRINGS

AIM: From the string input count the special character, alphabets, digits, lowercase and uppercase characters.

Algorithm:

1. Start the program.
2. Get the input string from the user.
3. Count the no.of digits no.of alphabets, no.of uppercase given by user.
4. Then print the output.
5. Stop the program.

Program:

```
def Count(str):
alpha,upper,lower,number,special = 0,0,0,0,0
for i in range(len(str)):
if str[i].isalpha():
alpha += 1
if str[i].isupper():
upper += 1
elif str[i].islower():
lower +=1
 elif str[i].isdigit():
number += 1
elif str[i]!=" ":
special += 1
print('Digits:', number)
print('Alphabets:', alpha)
print('Special characters:', special)
print('Lowercase:', lower)
print('Uppercase:', upper)

str = input("Enter a string: ")
Count(str)
```

Output:

```
Enter a string: sathyabama @2023
Digits: 4
Alphabets: 10
Special characters: 1
Lowercase: 10
Uppercase: 0
```

Result: The above program is executed and verified successfully.

EX.NO:05(B)- Print the String "Welcome". Matrix size must be N X M. (N is an odd natural number, and M is 3 times N.). The design should have 'WELCOME' written in the center. The design pattern should only use |, .and - characters.

AIM: Print the String "Welcome". Matrix size must be N X M. (N is an odd natural number, and M is 3 times N.). The design should have 'WELCOME' written in the center. The design pattern should only use |, .and - characters.

Algorithm:

1. Start the program.
2. Size of math must be N*M (W is an odd natural number and M is 3time N)
3. Design pattern should only use ‘1’, ’0’, ’-‘ characters.
4. Get the N amd M values from the user.
5. Print the output.
6. Stop the proram.

Program:

```
import math
N, M = map(int, input("Enter N and M: ").split())
for i in range(0,math.floor(N/2)):
  s= '.|.'*i
  print (s.rjust(math.floor((M-2)/2),'-')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
print ('WELCOME'.center(M,'-'))
for i in reversed(range(0,math.floor(N/2))):
```

```
 s = '.|.'*i
 print (s.rjust(math.floor((M-2)/2),'-')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
```

Output:

```
Enter N and M: 9 27
------------.|.------------
---------.|..|..|.---------
------.|..|..|..|..|.------
---.|..|..|..|..|..|..|.---
----------WELCOME----------
---.|..|..|..|..|..|..|.---
------.|..|..|..|..|.------
---------.|..|..|.---------
------------.|.------------
```

Result: the above program is executed and verified successfully.

EX.NO:06-DATA PREPROCESSING: BUILDING GOOD TRAINING SETS

AIM: To write a python program to implement data processing for building good training sets of data.

Algorithm:

1. Start the program.
2. Import Libraries and dataset.
3. Find the description of data in the data frame count the number of rows that are having no values from each column being describe().
4. Print the no.of columns, columns lables,column, data types from the data frame using info()
5. Replace the value 0 with NAN with replace()
6. Input the missing data with mean values.
7. Assign the values to x excepting the last column and assign values to y using loc[]
8. Split the dataset into training, testing(80:20).

Program:

```
import pandas as pd
df = pd.read_csv("/Heart.csv")
df

df.describe()

df.info()

df.replace(0,'NAN')

df.dropna()

df.fillna(df.mean())

x=df.iloc[:,0:14].values
x

y=df.iloc[:,14].values
y
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2, random_state=0)
print (x_train.shape)
print (x_test.shape)
```

Output:

```
(242, 14)
(61, 14)
```

Result: The program verified and executed successfully.

EX.NO:07- MANIPULATE THE TWITTER DATASET

Aim: To create a program for manipulating the twitter dataset.

Algorithm:

1. Import the required libraries (pandas, numpy).
2. Load the datset using pandas “read.csv” function A store it in a variable (ds) “re”.
3. Get summary of the data using “data” , and remove the pattern in the dataset by declaring or using the det “remove pattern”.
4. Replace the new “data” in the dataset available using [new]
5. Replace the text by using str replace () and print the data.
6. Split the data and print to Henized tweet using “to henized” tweet head() function.
7. Using import porter stemmer remove the common & inflexional endings from words in English and print the tokenized tweet.

Program:

```
import pandas as pd
import numpy as np
import re

data = pd.read_csv("/content/tweets1.csv")
data

def remove_pattern(input_txt, pattern):
 r = re.findall(pattern,input_txt)
 for i in r:
   input_txt = re.sub(i,'',input_txt)
 return input_txt
print(data)

data['new'] = np.vectorize(remove_pattern)(data ['text'],"@[\w]*")
print(data)

data['new'] = data['new'].str.replace("[^a-zA-Z#]"," ")
print(data)

data['new'] = data['new'].apply(lambda x:' '.join([w for w in x.split() if len(w) > 3]))
print (data)

tokenized_tweet = data['new'].apply (lambda x:x.split())
print (tokenized_tweet.head())

from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda x:[stemmer.stem(i) for i in x])
print (tokenized_tweet.head())
```

Output:

```
0                [robot, spare, human, http, jujqwfcv]
1    [exactli, tesla, absurdli, overvalu, base, pas...
2                                               [walt]
3                          [stormi, weather, shortvil]
4                      [coal, die, frack, basic, dead]
Name: new, dtype: object
```

Result: The program to manipulate the twitter dataset using python is manipulated successfully and verified.

EX.NO:08- EVALUATING THE RESULTS OF MACHINE LEARNING

Aim: To create a program for evaluating the result of machine learning.

Algorithm:

1.Read actual values vs predicted values
2.Compute the following:
3. Compute the Confusion Matrix
4. Compute the Accuracy
5. Compute the Specificity
6. Compute the Sensitivity
7. Compute the Precision
8. Compute the Recall
9. Compute the Misclassification Error

Program:

```
y=['0','1','0','1','1','1','0','1','0','1','0','0','0','1','1','1','0','1','1','0']
y_pred = ['0','0','0','0','1','0','1','1','1','1','0','0','0','0','0','1','0','1','1','0']
print (y)
print(y_pred)

j=0
TP,TN,FP,FN = 0,0,0,0
for i in y:
 if i == '1' and y_pred[j] =='1':
   TP +=1
 elif i == '0' and y_pred[j] =='0':
   TN +=1
 elif i == '1' and y_pred[j] =='0':
   FP +=1
 elif i == '0' and y_pred[j] =='1':
   FN+=1
 j+=1
confusion_matrix = [TP,TN,FP,FN]
print ("Confusion Matrix : ", confusion_matrix)
ACC = (TP+TN) / (TP+FP+TN+FN)
print ("ACCURACY : ", ACC)
PREC = TP / (TP+FP)
print ("PRECISION : ", PREC)
REC = TP / (TP+FN)
print ("RECALL : ", REC)
SN = TP/ (TP+FN)
print ("SENSITIVITY : ", SN)
SP = TN/ (TN+FP)
print ("SPECIFICITY : ", SP)
MCE = 1-ACC
print ("MISCLASSIFICATION ERROR : ", MCE)
```

Output:

```
Confusion Matrix :  [6, 7, 5, 2]
ACCURACY :  0.65
PRECISION :  0.5454545454545454
RECALL :  0.75
SENSITIVITY :  0.75
SPECIFICITY :  0.5833333333333334
MISCLASSIFICATION ERROR :  0.35
```

Result: The program to evaluating the result of machine learning is successfully evaluated and verified.

EX.NO:09- IMPLEMENT CORRELATION AND REGRESSION TECHNIQUES

AIM: Write a program to implement correlation and regression techniques by using python.

Algorithm:

1. **Import the libraries.**
2. **Read the csv file and let the variables**

(x,y).3.Find the sum of the means (x,y).

4. **Find zip-li, val, b&bo**
5. **Assign the test and train dataset split.**
6. **Create the machine Learning linear regression model using the train**

dataset.7.plotting classification data in matplotlib.

Program:

```
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r"Salary_Data.csv")
x=list(df["YearsExperience"])
y=list(df["Salary"])
df
def LinearRegressor(x,y):
  sumX=sum(x)
  sumY=sum(y)
  xMean=sumX/len(x)
  yMean=sumY/len(y)
  x_minus_xmean=[val-xMean for val in x]
  y_minus_ymean=[val-yMean for val in y]
  zip_li=zip(x_minus_xmean,y_minus_ymean)
  val=[x*y for x,y in zip_li]
  b1=sum(val)/sum([x**2 for x in x_minus_xmean])
  b0=yMean-b1*xMean
  return b0,b1
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/2,shuffle=True)
b=LinearRegressor(x_train,y_train)
y_pred=[b[0]+b[1]*val for val in x_test]
r2_score(y_test,y_pred)
plt.plot(x_test,y_pred)
plt.scatter(x_test, y_test,c="k")
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import numpy as np
print( "RMSE: ",np.sqrt( mean_squared_error( y_test, y_pred ) ))
#R-squared value
print( "R-squared: ",r2_score( y_test, y_pred ) )
```

Result: The program is executed and the output is verified.

EX.NO:10-IMPLEMENTING CLASSIFICATION ALGORITHM

AIM: Write program to implement classification Algorithm by using python.

Algorithm:

1.Import the libraries.
2.fetch data
3.Determine the target variables
4.creation of predictors variables
5.test and train dataset split
6.create the machine learning classification model using the train dataset
7.prediction
8.plotting classification data in matplotlib.

Program:

```
import pandas as pd
data=pd.read_csv("Iris.csv")
X =data.iloc[:,[1,2,3,4]].values
y =data.iloc[:,5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred=gnb.predict(X_test)
from sklearn import metrics
print("Classification Accuracy:", metrics.accuracy_score(y_test, y_pred)*100)
cm=metrics.confusion_matrix(y_test,y_pred)
print(cm)
import seaborn as sn
from matplotlib import pyplot as plt
plt.figure(figsize=(5,4))
plt.xlabel('Predicted value')
plt.ylabel('Actual value')
plt.show()
sn.heatmap(cm,annot=True)
```

Output:

```
[[11 0 0]
 [0 12 1]
 [0 0 6]]
```

Result: To implement classification algorithm by using python has been executed successfully and verified output.

EX.NO:11-IMPLEMENTING CLUSTERING USING K-MEANS CLUSTERING ALGORITHM

AIM: To implement clustering using K-means clustering Algorithm by using python.

ALGORITHM:

1.select the number k to decide the no.of cluster.
2.select random k points or centriods.
3.Assign each from the data point to their cloest centroid, which will from the predefine k clusters.
4.Calculate the varience and place a new centriod of each other.
5.Repeat the first 3rd to redesign the datapoint.
6.The requirements occure else go to finish
7.Stop the program

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df= pd.read_csv('C:/Users/chill/Downloads/Mall_Customers.csv')
df.head(3)
len(df)
X = df.iloc[:, [3,4]].values
X[0:5]
# KMeans class from the sklearn library.
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=10, random_state=0)
```

```
kmeans.n_clusters
y_kmeans = kmeans.fit_predict(X)
df['cluster'] = y_kmeans
print(y_kmeans.shape)
# Visualising the clusters
plt.scatter(X[y_kmeans==0, 0], X[y_kmeans==0, 1], s=100, c='red', label ='Cluster 1')
plt.scatter(X[y_kmeans==1, 0], X[y_kmeans==1, 1], s=100, c='blue', label ='Cluster 2')
plt.scatter(X[y_kmeans==2, 0], X[y_kmeans==2, 1], s=100, c='green', label ='Cluster 3')
plt.scatter(X[y_kmeans==3, 0], X[y_kmeans==3, 1], s=100, c='cyan', label ='Cluster 4')
plt.scatter(X[y_kmeans==4, 0], X[y_kmeans==4, 1], s=100, c='magenta', label ='Cluster 5')
#Plot the centroid.
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s=300, c='yellow', label = 'Centroids')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income(k$)')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

RESULT: Implementing clustering using k-means clustering algorithm has been implemented successfully.