**Q.FIFO Page Replacement Algorithm...**

#include <stdio.h>

#define TMAX\_FRATIES 10

#define TMAX\_PAGES 10

int main()

{

int pages [MAX\_PAGES];

int frames [MAX\_ FRAMES ];

int n, f, i, j,k, Page\_faults=0;

int current\_frame = 0;

Printf("enter the number of pages: ");

Scanf("%d", &n);

Printf("enter the page reference sequences: ");

For(i=o; i<n; i++)

{

Scanf("%d", & pages[i]);

}

printf("enter the number of frames: ");

Scanf("%d", &f);

for (i=o; i<f; i++);

{

frames[i]=-1;

}

for (i=0; i<n; i++)

{

int found = 0;

for(j=0; j<f;j++)

{

if (frames [i] == pages[i]){

found = 1;

break;

}

}

if (!found){

frames [current\_frame] = pages[i];

Current\_frame = (current\_frame +1) %f;

Page\_faults ++;

}

printf("\nframes:");"

for(k=0; K<f; k++)

{

if (frames [k]==-1)

printf("-");

else

printf("%d", frames [K]);"

}

}

printf("\n in Total page fauts : %d\n", page faults);

return 0;

}

**2)Accept Input at compile time for FIFO page replacement Algorithm.**

#include <stdio.h>

Void FPR(int pages [], int n, int frame\_size)

{

int frames [ frame\_size];

int page\_faults = 0;

int i, j;

for (i=0; i<frame\_size; j++)

{

if (frames[1] = = pages[i])

{

found = 1;

break;

}

}

If(found==0)

{

Page\_fault++;

For(j=0;j<frames\_size -1;j++)

{

Frames[j]=frames[j+1];

}

Frames[j]==frames[j+1];

}

Frames[frame\_size-1]=pages[i];

}}

printf("page faults: %d \n", page faults);

int main()

{

int pages[] = {1, 2, 3, 4, 3, 2, 1,5,6,7,9};

int n = sizeof(pages) / sizeof(pages[0]);

int frame size = 4;

printf("page replacement Algorithm fifo\n”);

printf("pages");

for (int i=0; i<n; i++)

{

printf("%d", pages[i]);

}

Printt (in frame size %d in frame size);

FPR( pages, n, frame\_size);

return 0;

}

**3) Optimal (page Replacement Algorithm)**

#include <stdio.h>

#include <limits.h>

int findOptimalPage (int pages[], int n, int frames[], int FrameSize, int index){

int forthest = index;

int replaceIndex=-1;

for (int i=0; i<FrameSize;i++)

{

int i;

for (j=index;j<n; j++) {

if (Frames[i]== pages[j]) {

if (j>farthest) {

farthest = j;

replaceīndex=i;

}

break;

}

}

if (j==n){

retun i;

}

}

retun(replaceIndex==-1)?0: replaceIndex;

}

int main() {

int n,frameSize;

printf("Enter the number of pages: ");

scanf("%d",&n);

int pages[n];

printf("Enter the page reference sequence:");

for (int i=0; i<n;i++) {

scanf("%d", & pages[i]);

}

Printf("Enter the number of frames: ");

scanf("%d", & frameSize);

int frames[frameSize];

int pagefaults = o, count = 0;

for(int i=0; i<frameSize; i++)

{

frames[i]=-1;

frames

}

for (int i=0; i<n; i++) {

int found = 0;

for (int j = 0; j<frameSize; i++) {

if (frames[j] == pages[i]) {

found = 1;

break;

}

}

if(!found){

if (count<frameSize){

frames[count++] = pages[i];

}else{

int replaceIndex = findOptimalPage(pages, n, frames, framesize, i+1);

frames[replaceIndex] = page [i];

}

PageFaults++;

}

printf(" frames after processing page %d: ", pages[i]);

for(int j=0; j < framesize; i++){

if(frames[j] != -1){

printf( "%d", frames[j]);

}else{

printf("-");

}

}

printf("\n");

}

printf("\n Total page Fault's: %d \n", pageFaults);

return 0;

}

**4) Program for accept Input at compile time for Optimal page replacement. algo.**

#include <stdio.h>

void optimal\_page\_replacement (int pages[], int n ,int frame\_size){

int frames [frame\_size);

int page\_Faults = 0;

int i, j ,k;

for (i=0; i<frame\_size; i++) {

frames[i]=- 1;

}

for (i=0;i<n; i++) {

int found = 0;

for(j=0; j<frame\_size;i++) {

if (frames[j] = = pages[i]){

found = 1;

break;

} }

if (found==0)

page\_faults++;

int max\_distance = 0;

int replace\_index = 0;

for (J=0; j<frame\_size; j++) {

int distance= o;

For (k=i+1; k<n; k++) {

if (frames [i]== pages [K]){

distance=k-i;

break;

}}

if (distance > max\_distance) {

max-distance = distance;

replace\_index = j;

} }

Frames[replace\_index] = pages[i];

} }

printf ("page faults: %d \n", page\_fault);

}

int main() {

int pages[] = {1,2,3,4,2, 1, 5, 6,2,1,2,3,7,4,5};

int n=sizeof(pages) / sizeof(pages[0]);

int frame\_size =3;

printf("\n page replacement Algorithm: Optimal \n");

printf("pages");

for (int i=0; i<n;i++){

printf("%d", pages[i]);

}

printf("\n frame size: %d\n", frame\_size);

Optimal-page-replacement( pages, n, frame\_size);

return 0;

}

**5) LRU Page replacement Algorithm.**

#include <stdio.h>

#includes limits.h>

Void findLRU(int time[], int n,int \*LruIndex) {

Int i, min=time[0];

\* lruindex = 0;

For(i=1; i<n; i++)

if (time [i] < min)

{

Min=time[i];

\*Lruindex = i;

}

}

}

Void LRUPageReplacement (int pages[], int n, int frames) {

int frame[frames], time [frames);

int i, j,k, LruIndex, pageFaults = 0;

for(i=0; i<frames; i++) {

frame[i]=-1;

time[i]=0;

}

for(i=0; i<n; i++){

int Found=0;

for(j=0; j<frames; i++) {

if (frame [j] == pages[i]) {

Found=1;

Time[j]=i;

break;

}

}

if (!found) {

if(i<frames){

frame[i] = page [i];

time[i]=i;

}else{

findLRU (time, frames, & LruIndex);

frame[LruIndex] = pages[i];

time[LruIndex] = i;

}

pageFaults++;

}

printf(" step%d ", i+1);

for(k=0;k< frames; k++) {

if (frame [K]!=-1)

printf("%d ", frame [k]);

else

printf("-");

}

Printf("\n");

}

printf(" \n Total Page faults: %d \n", Pagefaults);

}

int main() {

int n, frames;

printf("Enter number of pages: ");

scanf("%d",&n);

int pages[n];

printf("Enter the page sequence: ");

for(int P=0; i<n; i++){

scanf("%d", &pages[i]);

}

printf("Enter Number of frames”);

scanf("%d", & frames);

LRUPageReplacement (pages, in, frames);

return 0;

}

**6) Implement c program and accept Input at compile time for LRU Page replacement algo**

#include <stdio.h>

#include<stdlib.h>

#define NUM\_PAGES 3

#define PAGE\_REFERENCES 10

int page\_refs[PAGE\_REFERENCES]= {1,2,3, 4,2, 1, 5, 2, 3,4 };

int main()

{

int pages[NUM\_PAGES];

int Page\_faults=0;

for(int i<o; i<NUM\_PAGES; i++)

{

Pages [i]=-1;

}

for (int i = 0; i<PAGE\_REFERENCES; i++)

int page = page\_refs[i];

int found = 0;

for(int j=0; j<NUM\_PAGES;j++)

{

if (Pages[j] = = page)

{

found =1 ;

break;

}

}

if (!found){

page\_faults++;

int Lru\_page=0;

for(int j=1;j<NUM\_PAGES; j++)

{

if (pages[j]<pages [Lru\_page])

{

Lru\_page=j;

}

}

pages [Lru\_page] = page;

}

}

printf ("page faults: %d\n" %d\n", page\_faults );

return 0;

}

**Association Rules**

**1.Consider following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse the performance of the model (Use sklearn package)**

import matplotlib.pyplot as plt

from scipy import stats

import numpy as np

x = np.array([1,2,3,4,5,6,7,8])

y = np.array([7,14,15,18,19,21,26,23])

slope, intercept, r, p, std\_err = stats.linregress(x, y)

def myfunc(x):

return slope \* x + intercept

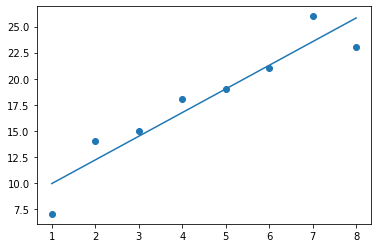
mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

**#output**



**2.Write python program to read “studentPerformance.csv” file. solve the following.**

**-display shape of dataset**

**-display top rows of dataset**

import numpy as np

import pandas as pd

import os

dataset=pd.read\_csv("StudentsPerformance.csv")

#To display the shape of dataset.

dataset

#To display the top rows of the dataset with their columns.

dataset.head()

#To display the number of rows randomly.

dataset.sample(5)

#To display the number of columns and names of the columns.

rows = len(dataset.axes[0])

cols = len(dataset.axes[1])

# Print the number of rows and columns

print("Number of Rows: " + str(rows))

print("Number of Columns: " + str(cols))

#name of columns

for col in dataset.columns:

print(col)

**#output**

**3.write a python program for aprior algorithm using ARM and print the rule ----**

import pandas as pd

import os

dataset=pd.read\_csv("groceries2.csv")

print(dataset.head(10))

transactions = []

# Add all the items from each row in one list( Neglect the 1st columns where all the items are in number (0-9))

for i in range(0, 100):

transactions.append([str(dataset.values[i,u]) for u in range(10, 32)])

from apyori import apriori

rules = apriori(transactions, min\_support=0.0040, min\_confidence=0.2, min\_lift=3, min\_length = 2)

results = list(rules)

# See the items that were bought together with their support

results\_list = []

for i in range(0, len(results)):

results\_list.append('RULE:\t' + str(results[i][0]) + '\nSUPPORT:\t' + str(results[i][1]))

print(dataset.head())

**Clustering**

1. **Write python program to read “car.csv” file.**

import pandas

from sklearn import linear\_model

df = pandas.read\_csv("cars.csv")

X = df[['Weight', 'Volume']]

y = df['CO2']

regr = linear\_model.LinearRegression()

regr.fit(X, y)

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)

**#output**

**2.linear regression**

import pandas as pd

import matplotlib.pyplot as plt

Stock\_Market = {'Year':[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],

'Interest\_Rate':[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'Unemployment\_Rate':[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'Stock\_Index\_Price':[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}df = pd.DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

plt.scatter(df['Interest\_Rate'], df['Stock\_Index\_Price'], color='red')

plt.title('Stock Index Price Vs Interest Rate', fontsize=14)

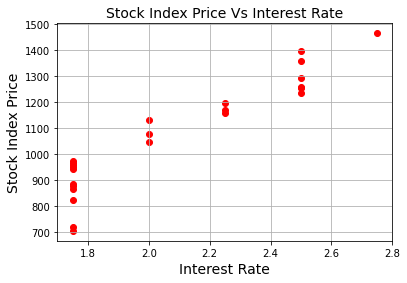
plt.xlabel('Interest Rate', fontsize=14)

plt.ylabel('Stock Index Price', fontsize=14)

plt.grid(True)

plt.show()

**#output**



**Assignment 3**

**1.K-mean algorithm**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler, normalize

from sklearn.decomposition import PCA

from sklearn.metrics import silhouette\_score

raw\_df = pd.read\_csv("GENERAL.csv")

raw\_df = raw\_df.drop('CUST\_ID', axis = 1)

raw\_df.fillna(method ='ffill', inplace = True)

raw\_df.head(2)

**#output**

ModuleNotFoundError: No module named 'sklearn'