INDEX

Practical No. 1

Aim: Implement Bayes Theorem

Q: Past data reveals that 10% of the patients entering a particular clinic have liver disease. Also 5% of the patients are alcoholic. Among the patients diagnosed with liver disease 7% are alcoholics. Find out the probability that the patients have liver disease if they are alcoholic.

A: A = Patients have liver disease; B = Patients are alcoholic

Code 1:-

```
P_A = float(input("Enter the percentage of patients having liver disease : "))
P_B = float(input("Enter the percentage of patients who are alcoholic : "))
P_B_Given_A = float(input("Enter the percentage of patients who are alcoholic if they have liv
er disease : "))

P_A_Given_B = (P_B_Given_A * P_A) / P_B

print("There are %.2f%% chances that patients have liver disease if they are alcoholic" % P_A_
Given_B )
```

Output:-

Past data reveals that 10% of the patients entering a particular clinic have liver disease. Also 5% of the patients are alcoholic. Among the patients diagnosed with liver disease 7% are alcoholics.Find out the probability that the patients have liver disease if they are alcoholic.

A = Patients have liver disease B = Patients are alcoholic

```
P_A = float(input("Enter the percentage of patients having liver disease : "))
P_B = float(input("Enter the percentage of patients who are alcoholic : "))
P_B_Given_A = float(input("Enter the percentage of patients who are alcoholic if they have liver disease : "))


P_A_Given_B = (P_B_Given_A * P_A) / P_B

print("There are %.2f%% chances that patients have liver disease if they are alcoholic" % P_A_Given_B )
```

```
Enter the percentage of patients having liver disease : 10
Enter the percentage of patients who are alcoholic : 5
Enter the percentage of patients who are alcoholic if they have liver disease : 7
There are 14.00% chances that patients have liver disease if they are alcoholic
```

Q: Given that in a particular sample space 1% of the patients have a certain genetic defect. 90% of the test for the gene detect the defect i.e. they are true positives. 9.6% of the tests are false positives. If a person gets a positive test result, what are the chances that they are actually have the genetic defect?

A: A = Patient has genetic defect; B = Patient has positive test result

Code 2:-

```
def Bayes_Theorem(P_A, P_B_Given_A, P_B_Given_Not_A):
  P_Not_A = 1 - (P_A / 100)
  P_Not_A = P_Not_A * 100
  temp1 = P_B_Given_A * P_A
  temp2 = P_B_Given_Not_A * P_Not_A
  P_A_Given_B = temp1 / (temp1 + temp2)
  return P_A_Given_B

P_A = float(input("Enter the percentage of patients having genetic defect : "))
P_B_Given_A = float(input("Enter the percentage of positive test results if the patients have the
 genetic defect : "))
P_B_Given_Not_A = float(input("Enter the percentage of positive test results if the patients do
not have the genetic defect : "))

Result = Bayes_Theorem(P_A, P_B_Given_A, P_B_Given_Not_A)

print("There are %.3f%% chances that the patient has genetic defect if they have a positive test
result" % Result)
```

Output:-

```
def Bayes_Theorem(P_A, P_B_Given_A, P_B_Given_Not_A):
  P_Not_A = 1 - (P_A / 100)
  P_Not_A = P_Not_A * 100
  temp1 = P_B_Given_A * P_A
  temp2 = P_B_Given_Not_A * P_Not_A
  P_A_Given_B = temp1 / (temp1 + temp2)
  return P_A_Given_B

P_A = float(input("Enter the percentage of patients having genetic defect : "))
P_B_Given_A = float(input("Enter the percentage of positive test results if the patients have the genetic defect : "))
P_B_Given_Not_A = float(input("Enter the percentage of positive test results if the patients do not have the genetic defect : "))

Result = Bayes_Theorem(P_A, P_B_Given_A, P_B_Given_Not_A)

print("There are %.3f%% chances that the patient has genetic defect if they have a positive test result" % Result)
```

```
Enter the percentage of patients having genetic defect : 1
Enter the percentage of positive test results if the patients have the genetic defect : 90
Enter the percentage of positive test results if the patients do not have the genetic defect : 9.6
There are 0.087% chances that the patient has genetic defect if they have a positive test result
```

✓ 14s completed at 7:08 AM

Practical No. 2

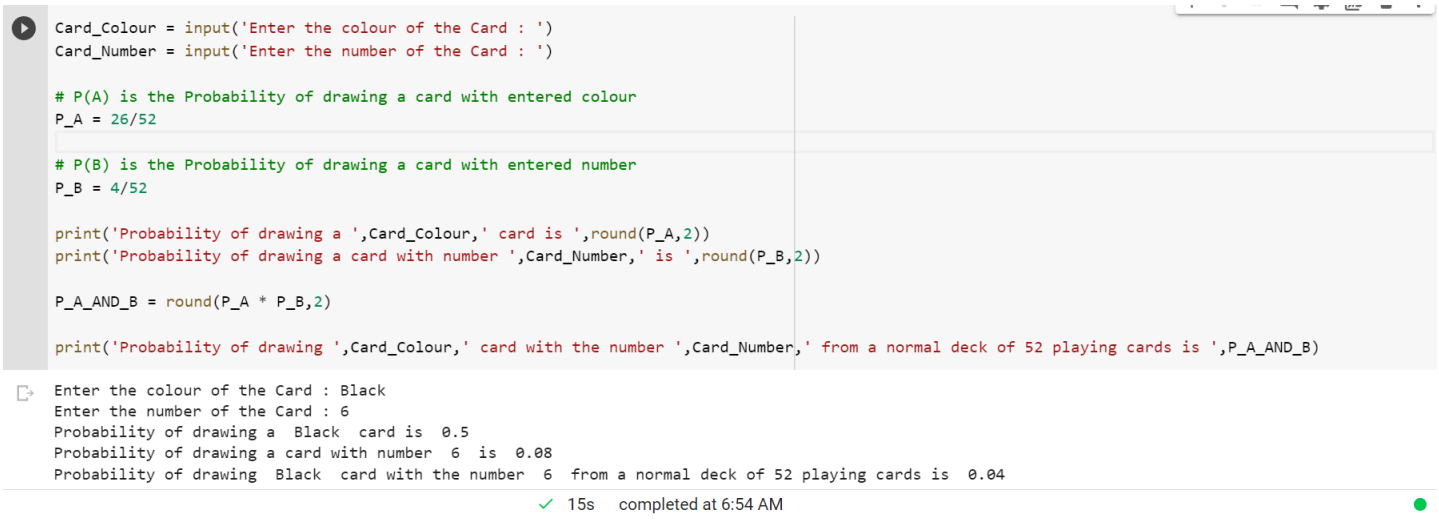
Aim: Implement Joint probability & Conditional probability

A) Joint probability

Code:-

```
Card_Colour = input('Enter the colour of the Card : ')
Card_Number = input('Enter the number of the Card : ')

# P(A) is the Probability of drawing a card with entered colour
P_A = 26/52

# P(B) is the Probability of drawing a card with entered number
P_B = 4/52

print('Probability of drawing a ',Card_Colour,' card is ',round(P_A,2))
print('Probability of drawing a card with number ',Card_Number,' is ',round(P_B,2))

P_A_AND_B = round(P_A * P_B,2)

print('Probability of drawing ',Card_Colour,' card with the number ',Card_Number,' from a nor
mal deck of 52 playing cards is ',P_A_AND_B)
```

Output:-



B) Conditional Probability

Code:-

```
import pandas as pd
import numpy as np
import io

from google.colab import files
uploaded = files.upload()

df=pd.read_csv(io.BytesIO(uploaded['student_data.csv']))
df

df['G']= round((df['G1'] + df['G2'] + df['G3'])/3)
df
```

```
df['G']= round((df['G1'] + df['G2'] + df['G3'])/3)
df
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	G
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1	1	3	6	5	6	6	6.0
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1	1	3	4	5	5	6	5.0
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2	3	3	10	7	8	10	8.0
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	2	1	1	5	2	15	14	15	15.0
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1	2	5	4	6	10	10	9.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	4	4	5	4	11	9	9	9	9.0
391	MS	M	17	U	LE3	T	3	1	services	services	...	4	5	3	4	2	3	14	16	16	15.0
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	3	3	3	3	3	10	8	7	8.0
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	1	3	4	5	0	11	12	10	11.0
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	2	3	3	3	5	5	8	9	9	9.0

395 rows × 34 columns

```
df['Percentage'] = df['G'] * 5
df['Grade_O'] = np.where( df['Percentage'] >=80,1,0)
df.head(10)
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	Dalc	Walc	health	absences	G1	G2	G3	G	Percentage	Grade_O
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	1	1	3	6	5	6	6	6.0	30.0	0
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	1	1	3	4	5	5	6	5.0	25.0	0
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	2	3	3	10	7	8	10	8.0	40.0	0
3	GP	F	15	U	GT3	T	4	2	health	services	...	1	1	5	2	15	14	15	15.0	75.0	0
4	GP	F	16	U	GT3	T	3	3	other	other	...	1	2	5	4	6	10	10	9.0	45.0	0
5	GP	M	16	U	LE3	T	4	3	services	other	...	1	2	5	10	15	15	15	15.0	75.0	0
6	GP	M	16	U	LE3	T	2	2	other	other	...	1	1	3	0	12	12	11	12.0	60.0	0
7	GP	F	17	U	GT3	A	4	4	other	teacher	...	1	1	1	6	6	5	6	6.0	30.0	0
8	GP	M	15	U	LE3	A	3	2	services	other	...	1	1	1	0	16	18	19	18.0	90.0	1
9	GP	M	15	U	GT3	T	3	4	other	other	...	1	1	5	0	14	15	15	15.0	75.0	0

```
df['High_Absentees'] = np.where( df['absences'] >=10, 1, 0)
df.head(10)
```

Fjob	...	Walc	health	absences	G1	G2	G3	G	Percentage	Grade_O	High_Absentees
teacher	...	1	3	6	5	6	6	6.0	30.0	0	0
other	...	1	3	4	5	5	6	5.0	25.0	0	0
other	...	3	3	10	7	8	10	8.0	40.0	0	1
services	...	1	5	2	15	14	15	15.0	75.0	0	0
other	...	2	5	4	6	10	10	9.0	45.0	0	0
other	...	2	5	10	15	15	15	15.0	75.0	0	1
other	...	1	3	0	12	12	11	12.0	60.0	0	0
teacher	...	1	1	6	6	5	6	6.0	30.0	0	0
other	...	1	1	0	16	18	19	18.0	90.0	1	0
other	...	1	5	0	14	15	15	15.0	75.0	0	0

```
df['Count'] = 1
df.head(10)
```

```
df = df[['Grade_O', 'High_Absentees', 'Count']]
df.head(5)
```

```
df = df[['Grade_O', 'High_Absentees', 'Count']]
df.head(5)
```

	Grade_O	High_Absentees	Count
0	0	0	1
1	0	0	1
2	0	1	1
3	0	0	1
4	0	0	1

```
pd.pivot_table(df, values='Count', index='Grade_O', columns='High_Absentees', aggfunc=np.size,fill_value=0)
```

High_Absentees	0	1
Grade_O		
0	283	78
1	29	5

```
Total = 283 + 78 + 29 + 5
#P(A) probability of getting grade of 80% or more
P_A = (29+5)/Total
print(P_A)

#P(A) probability of missing 10 lectures or more
P_B = (78+5)/Total
print(P_B)

# P(A_Intersection_B) is the probability of getting grade of 80% or more and missing 10 lecture
s or more
P_A_Intersection_B =5/Total
print(P_A_Intersection_B)

P_A_Given_B = P_A_Intersection_B / P_B
print(P_A_Given_B)

print('probability of students getting atleast 80% grade given they have missed 10 lectures or m
ore is ', round(P_A_Given_B,2))
```

Output:-

```
[2] Total = 283 + 78 + 29 + 5
    #P(A) probability of getting grade of 80% or more
    P_A = (29+5)/Total
    print(P_A)

    0.08607594936708861

[3] #P(A) probability of missing 10 lectures or more
    P_B = (78+5)/Total
    print(P_B)

    0.21012658227848102

[4] # P(A_Intersection_B) is the probability of getting grade of 80% or more and missing 10 lectures or more
    P_A_Intersection_B =5/Total
    print(P_A_Intersection_B)

    0.012658227848101266

[5] P_A_Given_B = P_A_Intersection_B / P_B
    print(P_A_Given_B)

    0.060240963855421686

    print('probability of students getting atleast 80% grade given they have missed 10 lectures or more is ', round(P_A_Given_B,2))

    probability of students getting atleast 80% grade given they have missed 10 lectures or more is  0.06
```

Practical No. 3

Aim: Write a program to implement Rule based matching

Code:-

```
import spacy
from spacy.matcher import Matcher
nlp=spacy.load('en_core_web_sm')
matcher=Matcher(nlp.vocab)

doc=nlp("2022 Fifa world cup : ITALY Wins")
pattern=[{'IS_DIGIT' :True} ,{'LOWER':'fifa'}, {'LOWER':'world'}, {'LOWER':'cup'}, {'IS_PUNCT':True}]
matcher.add('FIFA_PATTERN', [pattern])
matches=matcher(doc)

for match_id, start, end in matches:
  matched_span=doc[start:end]
  print(matched_span.text)
```

Output:-

```
2022 Fifa world cup :
```

```
doc=nlp("I loved dogs but now i love cats more")
pattern=[{'LEMMA':'love'} ,{'POS':'NOUN'}]
matcher.add('DOG_PATTERN',[pattern])
matches=matcher(doc)

for match_id, start, end in matches:
  matched_span=doc[start:end]
  print(matched_span.text)
```

Output:-

```
loved dogs
love cats
```

```
doc=nlp("I bought smartphone and now I am buying another smartphone")
pattern=[{'LEMMA':'buy'} ,{"POS":"DET", "OP" :"?"}, {'POS':'NOUN'}]
matcher.add('BUY_PATTERN',[pattern])
matches=matcher(doc)

for match_id, start, end in matches:
  matched_span=doc[start:end]
  print(matched_span.text)
```

Output:-

```
bought smartphone
buying another smartphone
```

Practical No. 5

Aim: Design a Fuzzy based application.

Code:-

```
# Union
A = dict()
B = dict()
C = dict()

A = {"a" : 0.3 , "b" : 0.4 , "c" : 0.6 , "d" : 0.9 , "e" : 0.2}
B = {"a" : 0.2 , "b" : 0.9 , "c" : 0.5 , "d" : 0.7 , "e" : 0.1}

print('Fuzzy Set A : ', A)
print('Fuzzy Set B : ', B)

for A_Key, B_Key in zip(A,B):
  A_Value = A[A_Key]
  B_Value = B[B_Key]

  if A_Value > B_Value:
    C[A_Key] = A_Value
  else:
    C[A_Key] = B_Value

print('Union Operation')
print('Fuzzy Set C : ', C)
```

Output:-

```
Fuzzy Set A :  {'a': 0.3, 'b': 0.4, 'c': 0.6, 'd': 0.9, 'e': 0.2}
Fuzzy Set B :  {'a': 0.2, 'b': 0.9, 'c': 0.5, 'd': 0.7, 'e': 0.1}
Union Operation
Fuzzy Set C :  {'a': 0.3, 'b': 0.9, 'c': 0.6, 'd': 0.9, 'e': 0.2}
```

```
# Intersection
A = dict()
B = dict()
C = dict()
```

```
A = {"a" : 0.3 , "b" : 0.4 , "c" : 0.6 , "d" : 0.9 , "e" : 0.2}
B = {"a" : 0.2 , "b" : 0.9 , "c" : 0.5 , "d" : 0.7 , "e" : 0.1}

print('Fuzzy Set A : ', A)
print('Fuzzy Set B : ', B)

for A_Key, B_Key in zip(A,B):
  A_Value = A[A_Key]
  B_Value = B[B_Key]

  if A_Value < B_Value:
    C[A_Key] = A_Value
  else:
    C[A_Key] = B_Value

print('Intersection Operation')
print('Fuzzy Set C : ', C)
```

Output:-

```
Fuzzy Set A :  {'a': 0.3, 'b': 0.4, 'c': 0.6, 'd': 0.9, 'e': 0.2}
Fuzzy Set B :  {'a': 0.2, 'b': 0.9, 'c': 0.5, 'd': 0.7, 'e': 0.1}
Intersection Operation
Fuzzy Set C :  {'a': 0.2, 'b': 0.4, 'c': 0.5, 'd': 0.7, 'e': 0.1}
```

```
# Difference
A = dict()
B = dict()
C = dict()

A = {"a" : 0.3 , "b" : 0.4 , "c" : 0.6 , "d" : 0.9 , "e" : 0.2}
B = {"a" : 0.2 , "b" : 0.9 , "c" : 0.5 , "d" : 0.7 , "e" : 0.1}

print('Fuzzy Set A : ', A)
print('Fuzzy Set B : ', B)

for A_Key, B_Key in zip(A,B):
  A_Value = A[A_Key]
  B_Value = B[B_Key]
```

```
  B_Value = 1 - B_Value

  if A_Value < B_Value:
    C[A_Key] = A_Value
  else:
    C[A_Key] = B_Value

print('Difference Operation')
print('Fuzzy Set C : ', C)
```

Output:-

```
Fuzzy Set A :  {'a': 0.3, 'b': 0.4, 'c': 0.6, 'd': 0.9, 'e': 0.2}
Fuzzy Set B :  {'a': 0.2, 'b': 0.9, 'c': 0.5, 'd': 0.7, 'e': 0.1}
Difference Operation
Fuzzy Set C :  {'a': 0.3, 'b': 0.09999999999999998, 'c': 0.5, 'd': 0.30000000000000004, 'e': 0.2}
```

```
# Complement
A = dict()
C = dict()

A = {"a" : 0.3 , "b" : 0.4 , "c" : 0.6 , "d" : 0.9 , "e" : 0.2}

print('Fuzzy Set A : ', A)

for A_Key in A:
   C[A_Key] = 1 - A[A_Key]

print('Complement Operation')
print('Fuzzy Set C : ', C)
```

Output:-

```
Fuzzy Set A :  {'a': 0.3, 'b': 0.4, 'c': 0.6, 'd': 0.9, 'e': 0.2}
Complement Operation
Fuzzy Set C :  {'a': 0.7, 'b': 0.6, 'c': 0.4, 'd': 0.09999999999999998, 'e': 0.8}
```

Practical No. 6

Aim: Write an application to implement supervised and unsupervised learning model.

Code:-

```
#Supervised Learning
from sklearn.linear_model import LinearRegression
import random

feature_set = []
target_set = []

rows = 200
limit = 2000

for i in range(0, rows):
  x = random.randint(0, limit)
  y = random.randint(0, limit)
  z = random.randint(0, limit)
  feature_set.append([x,y,z])
  function = (10*x) + (2*y) + (0*z)
  target_set.append(function)

model = LinearRegression()
model.fit(feature_set, target_set)

test_set=[[8,10,0]]
prediction= model.predict(test_set)
print('Prediction : ' ,prediction)
```

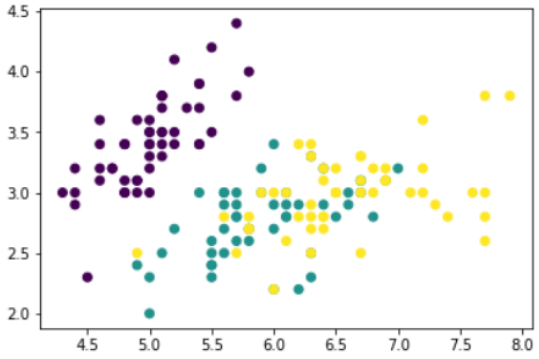
Output:-

```
test_set=[[8,10,0]]
prediction= model.predict(test_set)
print('Prediction :  ' ,prediction)

Prediction :  [100.]
```

```
#Unsupervised Learning
from sklearn import datasets

import matplotlib.pyplot as plt
iris_df = datasets.load_iris()
print(dir(iris_df))
print(iris_df.feature_names)
print(iris_df.target)
print(iris_df.target_names)
x_axis = iris_df.data[:,0]
y_axis = iris_df.data[:,1]
plt.scatter(x_axis, y_axis, c=iris_df.target)
plt.show()
```

Output:-

```
['DESCR', 'data', 'data_module', 'feature_names', 'filename', 'frame', 'target', 'target_names']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
['setosa' 'versicolor' 'virginica']
```



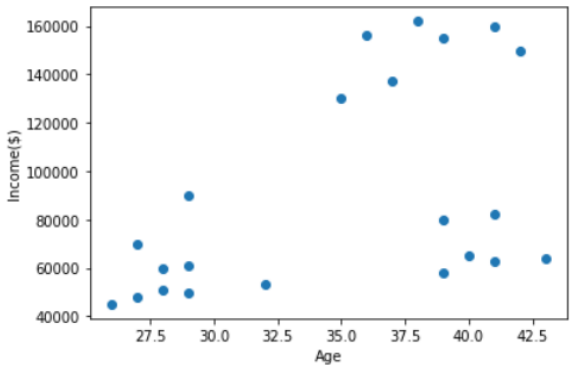
Practical No. 7

Aim: Implementation of Clustering Algorithm (K-means)

Code:-

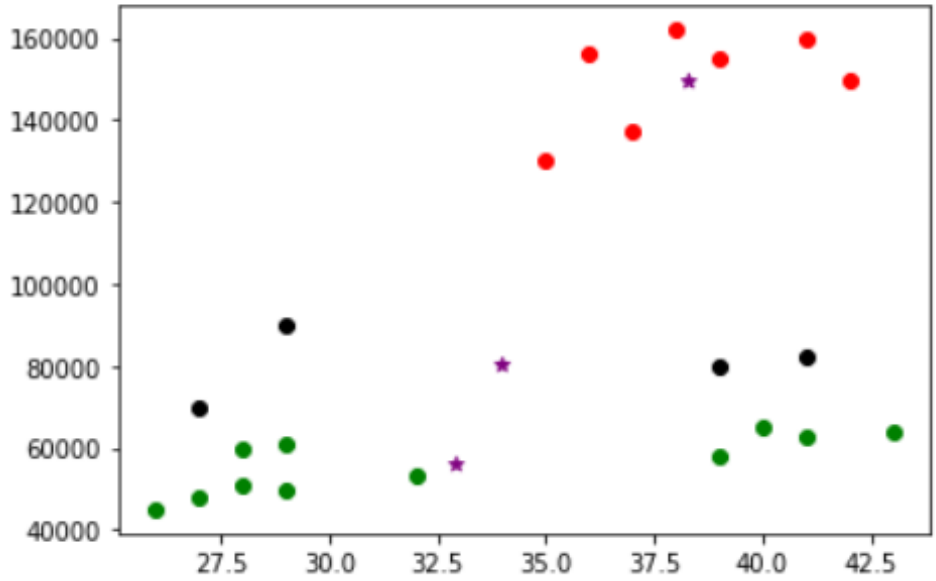
```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt

df=pd.read_csv('/content/sample_data/Income.csv')
df.head()

plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')

km = KMeans(n_clusters=3)
predicted = km.fit_predict(df[['Age','Income($)']])
df['cluster']=predicted
df.head()
```

	Name	Age	Income($)	cluster
0	Rob	27	70000	2
1	Michael	29	90000	2
2	Mohan	29	61000	0
3	Ismail	28	60000	0
4	Kory	42	150000	1



```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]

plt.scatter(df1['Age'],df1['Income($)'],color='green')
plt.scatter(df2['Age'],df2['Income($)'],color='red')
plt.scatter(df3['Age'],df3['Income($)'],color='black')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='Centroid')
```

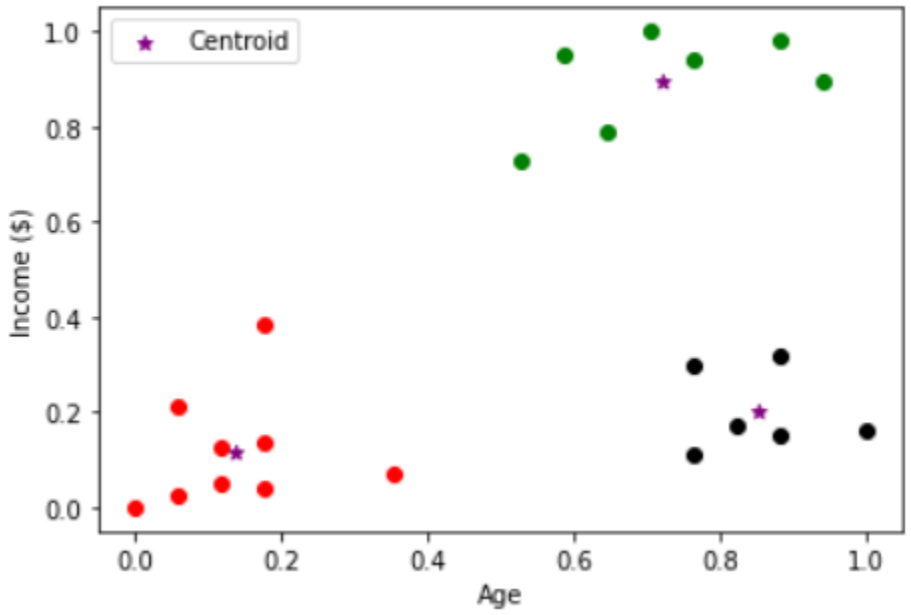
```
<matplotlib.collections.PathCollection at 0x7f2ca99df910>
```




```
km = KMeans(n_clusters=3)
predicted = km.fit_predict(df[['Age','Income($)']])
df['cluster']=predicted
df.head()

df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]

plt.scatter(df1['Age'],df1['Income($)'],color='green')
plt.scatter(df2['Age'],df2['Income($)'],color='red')
plt.scatter(df3['Age'],df3['Income($)'],color='black')
```

```
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='Centroid')

plt.xlabel('Age')
plt.ylabel('Income ($)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f2ca98afbd0>
```



Practical No. 8

Aim: Write an application to implement support vector machine algorithm.

Code:-

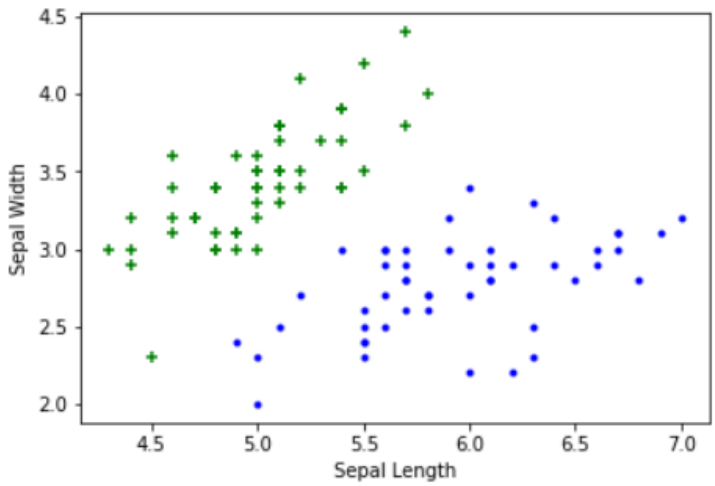
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC


iris = load_iris()
print(iris.feature_names)

print(iris.target_names)

df = pd.DataFrame(iris.data,columns=iris.feature_names)
print(df)

df['target'] = iris.target
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
print(df)

df0 = df[:50]          # setosa
df1 = df[50:100]       # versicolor
df2 = df[100:]         # virginica
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

     target flower_name
0         0      setosa
1         0      setosa
2         0      setosa
3         0      setosa
4         0      setosa
..      ...         ...
145       2   virginica
146       2   virginica
147       2   virginica
148       2   virginica
149       2   virginica

[150 rows x 6 columns]
```

```
# Sepal length vs Sepal Width
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

Output:-

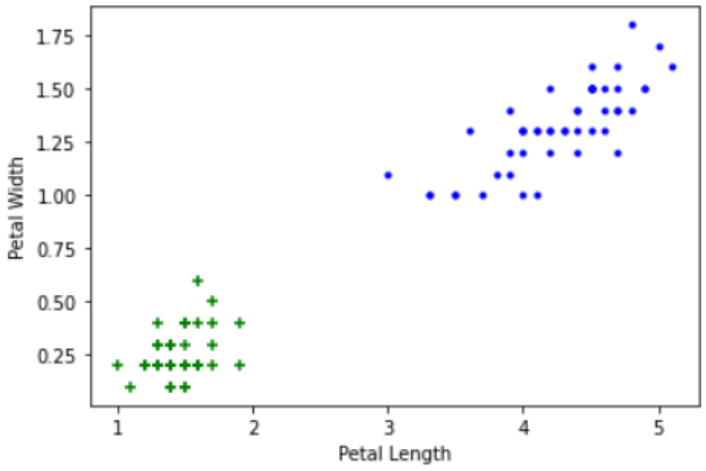
```
<matplotlib.collections.PathCollection at 0x7f213f3d1910>
```




```
# Petal length vs Pepal Width
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

Output:-

```
<matplotlib.collections.PathCollection at 0x7f213f3d1f90>
```



```
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)

model = SVC()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Output:-

```
1.0
```

Practical No. 9

Aim: Design a bot using AIML.

Code:

Pip install aiml
Pip install python-aiml
Pip3 install python-aiml

****Start.py****

```
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
while True:
input_text = input(">Human: ")
response = kernel.respond(input_text)
print(">Bot: "+response)
```

****std-startup.xml****

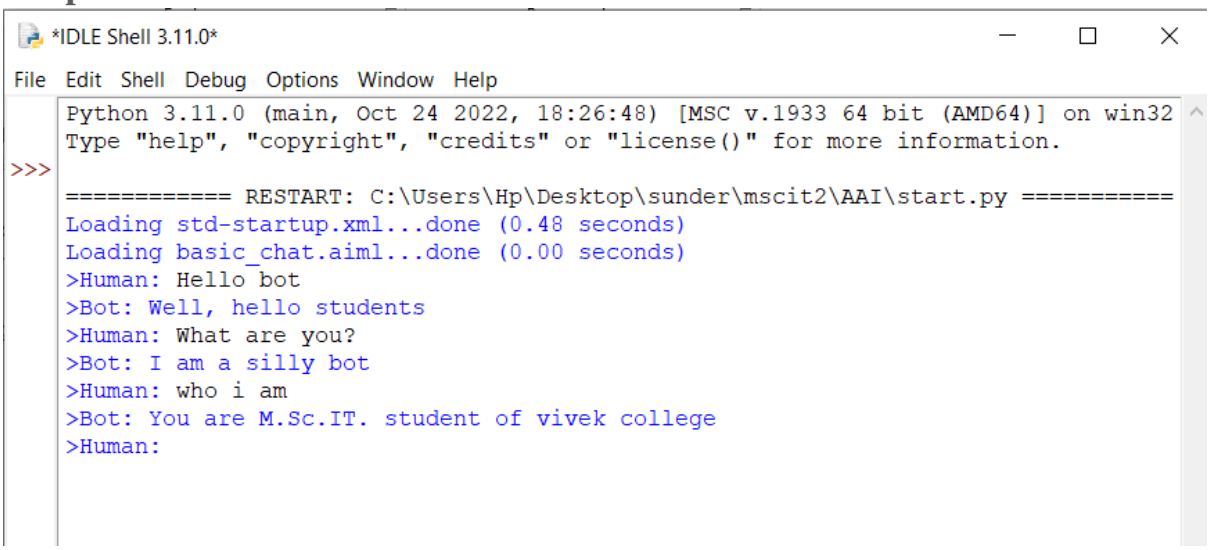
```
<aiml version="1.0.1" encoding="UFT-8">
<category>
<pattern>LOAD AIML B</pattern>
<template>
<learn>basic_chat.aiml</learn>
</template>
</category>
</aiml>
```

****Basic_chat.aiml****

```
<aiml version="1.0.1" encoding="UTF-8">
<category>
<pattern>HELLO *</pattern>
<template>
Well, hello students
</template>
</category>
<category>
<pattern>WHAT ARE YOU</pattern>
<template>
I am a silly bot
```

```
</template>
</category>
<category>
<pattern>WHAT DO YOU DO</pattern>
<template>
I am here to annoy you!
</template>
</category>
<category>
<pattern>WHO I AM</pattern>
<template>
You are M.Sc.IT. student of vivek college
</template>
</category>
</aiml>
```

Output:



Practical No. 10

Aim: Design an Expert System using AIML.

Code:

****Doctor_chat.aiml****

```
<aiml version="1.0.1" encoding="UTF-8">
<category>
<pattern>HELLO *</pattern>
<template>
Well, hello patient
</template>
</category>
<category>
<pattern>WHO ARE YOU</pattern>
<template>
I am a Doctor bot
</template>
</category>
<category>
<pattern>WHAT DO YOU DO</pattern>
<template>
I am here to treat you!
</template>
</category>
<category>
<pattern>WHO I AM</pattern>
<template>
You are my patient
</template>
</category>
</aiml>
```

****Std_startup.xml****

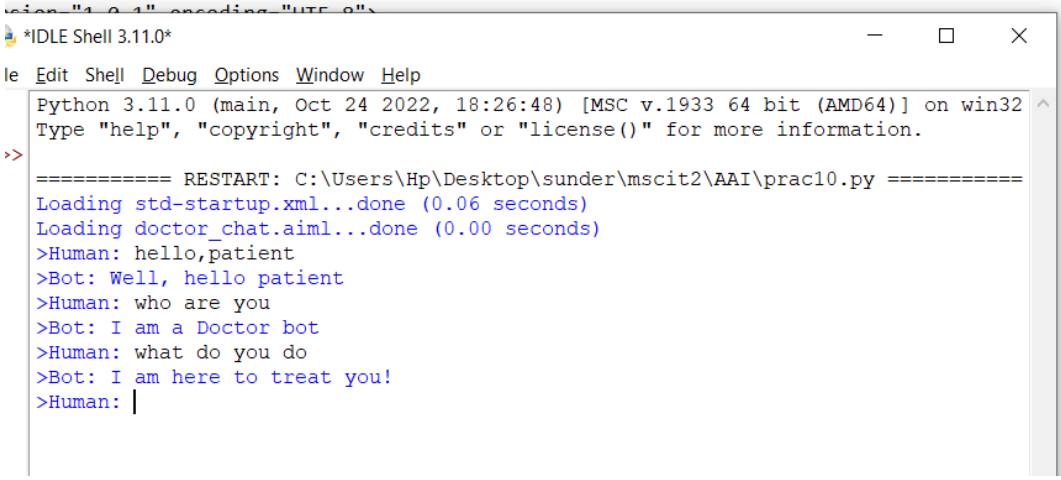
```
<aiml version="1.0.1" encoding="UFT-8">
<category>
<pattern>LOAD AIML B</pattern>
<template>
<learn>doctor_chat.aiml</learn>
</template>
</category>
```

```
</aiml>
```

****Practical10.py****

```
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
while True:
input_text = input(">Human: ")
response = kernel.respond(input_text)
print(">Bot: "+response)
```

Output:



Practical No. 11

Aim: Design an application to simulate Semantic Web.

Code:

Pip install rdflib

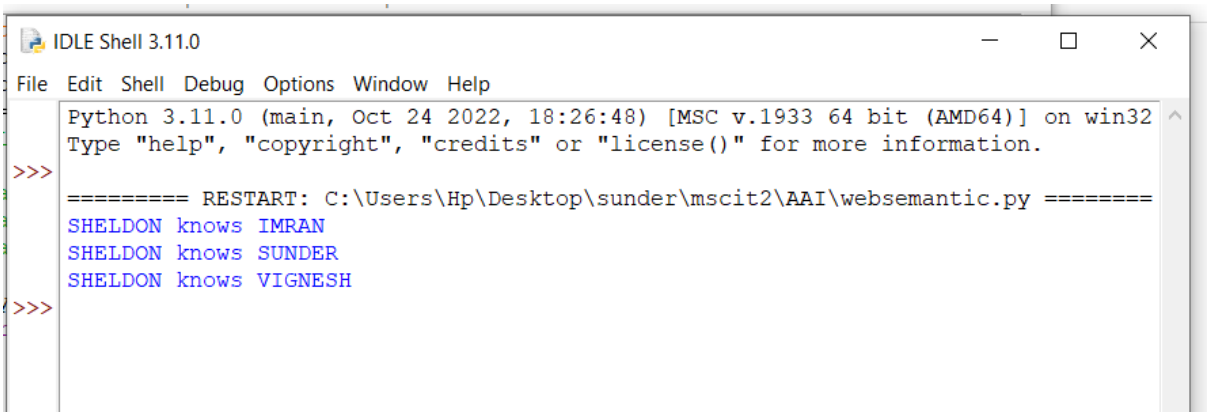
****Websemantic.py****

```
import rdflib
mygraph = rdflib.Graph()
mygraph.parse("myfoaf.rdf")
qres = mygraph.query(
"""SELECT DISTINCT ?fname ?lname
WHERE {
?a foaf:knows ?b .
?a foaf:name ?fname .
?b foaf:name ?lname .
}""")
for myrow in qres:
   print("%s knows %s" % myrow)
```

****roaf.rdflib****

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:foaf="http://xmlns.com/foaf/0.1/"
     xmlns:admin="http://webns.net/mvcb/">

<foaf:Person rdf:nodeID="me">
  <foaf:name>SHELDON</foaf:name>
  <foaf:knows>
    <foaf:Person>
      <foaf:name>IMRAN</foaf:name>
    </foaf:Person>
  </foaf:knows>
  <foaf:knows>
    <foaf:Person>
      <foaf:name>SUNDER</foaf:name>
    </foaf:Person>
  </foaf:knows>
  <foaf:knows>
```

```
    <foaf:Person>
      <foaf:name>VIGNESH</foaf:name>
    </foaf:Person>
  </foaf:knows>
</foaf:Person>

</rdf:RDF>
```



Practical No. 12

Aim: Design an Artificial Intelligence application to implement Intelligent Agent.

Code:

```
import random

def display(room):
    print(room)

# 1 means dirty location
# 0 means clean location

room = [
        [1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1],
      ]

print("All the locations in the room are dirty")
display(room)

x=0 # rows
y=0 # cols
while x < 4:
    while y < 4:
        room[x][y] = random.choice([0,1])
        y+=1
    x+=1
    y=0

print("Before cleaning the room the Vacuum cleaner detects all the random dirts in the followin
g locations")
display(room)

x=0
y=0
z=0 #number of rooms cleaned

#Agent code
```

```
while x < 4:
    while y < 4:
        if room[x][y] == 1:
            print("Vacuum cleaner is in this location now : ", x, y)
            room[x][y] = 0
            print("Location cleaned : ", x, y)
            z+=1
        y+=1
    x+=1
    y=0

print("Number of locations cleaned = ", z)

Performance=(100-((z/16)*100))
print("Room is clean now")
display(room)
print("Cleaning Performance = ", Performance,"%")
```

Output:

```
All the locations in the room are dirty
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
```

```
Before cleaning the room the Vacuum cleaner detects all the random dirts in the following locations
[[0, 1, 1, 0], [1, 0, 0, 1], [1, 1, 1, 1], [1, 0, 0, 0]]
```

```
Vacuum cleaner is in this location now :  0 1
Location cleaned :  0 1
Vacuum cleaner is in this location now :  0 2
Location cleaned :  0 2
Vacuum cleaner is in this location now :  1 0
Location cleaned :  1 0
Vacuum cleaner is in this location now :  1 3
Location cleaned :  1 3
Vacuum cleaner is in this location now :  2 0
Location cleaned :  2 0
Vacuum cleaner is in this location now :  2 1
Location cleaned :  2 1
Vacuum cleaner is in this location now :  2 2
Location cleaned :  2 2
Vacuum cleaner is in this location now :  2 3
Location cleaned :  2 3
Vacuum cleaner is in this location now :  3 0
Location cleaned :  3 0
Number of locations cleaned =  9
```

```
Room is clean now
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
Cleaning Performance =  43.75 %
```