1. **Finding Factorial of a Number**

n = int(input(“Enter the number: ”))

fact = 1

for i in range(1,n+1):

fact = fact\*i

print(“Factorial of “,n,”is”,fact)

**OUTPUT:-**

Enter the number: 3

Factorial of 3 is 6

1. **To Check Prime Number**

n = int(input(“Enter the number: “))

flag = False

if n>1:

for i in range(2,n):

if(n%i)==0:

flag = True

break

if flag:

print(n,”is not a Prime Number”)

print(i,”times”,n//I,”is”,n)

else:

print(n,”is a Prime Number”)

**OUTPUT:-**

Enter the number: 2

2 is a Prime Number

Enter the number: 6

6 is not a Prime Number

2 times 3 is 6

1. **Write a Python program to get the length of the first two vectors of a given list.**

import numpy as np

list1=[np.array([1,2,3,4]), 'Program', 1.234]

list2=[range(1,5), "R", 12.0]

print(type(list1[0]))

print(list2)

print(type(list2[0]))

**OUTPUT:-**

<class 'numpy.ndarray'>

[range(1, 5), 'R', 12.0]

<class 'range'>

1. **Write a Python program to create a list containing strings, numbers, vectors and a logical values.**

list\_data = [“Python”, “PHP”, [5,7,9,11], True, 125.17, 75.83]

print(“Data of the list: “)

print(list\_data)

**OUTPUT:-**

Data of the list:

['Python', 'PHP', [5, 7, 9, 11], True, 125.17, 75.83]

1. **Write a Python program to merge two given lists into one list**

n1=[1,2,3]

c1 = ["Red", "Green", "Black"]

print("Original lists: ")

print(n1)

print(c1)

print("Merge the said lists: ")

mlist = n1 + c1

print("New merged list: ")

print(mlist)

**OUTPUT:-**

Original lists:

[1, 2, 3]

['Red', 'Green', 'Black']

Merge the said lists:

New merged list:

[1, 2, 3, 'Red', 'Green', 'Black']

**KNN ALGORITHM:-**

import numpy as n

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv(“D:\\ML\\Datasets\\ads.csv")

print(data.head())

print(data.isnull().any())

x=data.iloc[:,1:4].values

y=data.iloc[:,4:5].values

print(x[:5])

print(y[:5])

from sklearn.preprocessing import LabelEncoder

lb= LabelEncoder()

x[:,0]= lb.fit\_transform(x[:,0])

print(x[:5])

from sklearn.model\_selection import train\_test\_split as tts

x\_train,x\_test,y\_train,y\_test = tts(x, y, test\_size = 0.1,random\_state=0)

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.transform(x\_test)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier( n\_neighbors = 5 , p = 2 )

knn.fit(x\_train,y\_train)

y\_pred=knn.predict(x\_test)

print(y\_pred)

print(y\_test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(y\_test,y\_pred))

from sklearn.metrics import confusion\_matrix

import sklearn.metrics as metrics

fpr, tpr ,threshold = metrics.roc\_curve(y\_test,y\_pred)

roc\_auc = metrics.auc(fpr,tpr)

print(roc\_auc)

plt.plot(fpr,tpr)

plt.xlim([0,1])

plt.ylim([0,1])

plt.style.use("dark\_background")

**OUTPUT:-**

User ID Gender Age EstimatedSalary Purchased

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

User ID False

Gender False

Age False

EstimatedSalary False

Purchased False

dtype: bool

[['Male' 19 19000]

['Male' 35 20000]

['Female' 26 43000]

['Female' 27 57000]

['Male' 19 76000]]

[[0]

[0]

[0]

[0]

[0]]

[[1 19 19000]

[1 35 20000]

[0 26 43000]

[0 27 57000]

[1 19 76000]]

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0

0 0 1]

[[0]

[0]

[0]

[0]

[0]

[0]

[0]

[1]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[1]

[0]

[0]

[1]

[0]

[1]

[0]

[1]

[0]

[0]

[0]

[0]

[0]

[1]

[1]

[0]

[0]

[0]

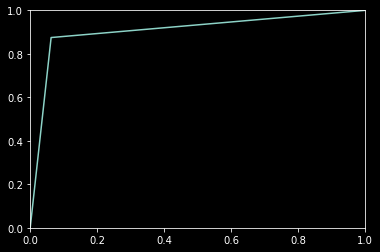
[0]

[0]

[0]

[1]]

0.925  
0.90625



**Classification Using Decision Tree:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv(“D:\\ML\\Datasets\\ads.csv")

print(data.head())

print(data.isnull().any())

x = data.iloc[:,1:4].values

y = data.iloc[:,4:5].values

print(x)

print(y[:10])

from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

x[:,0] = lb.fit\_transform(x[:,0])

print(x)

from sklearn.model\_selection import train\_test\_split as tts

x\_train,x\_test,y\_train,y\_test = tts(x, y, test\_size = 0.1,random\_state=0)

print(x\_train)

print(x\_test[:10])

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='entropy')

print(dt.fit(x\_train,y\_train))

y\_pred = dt.predict(x\_test)

print(y\_pred)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(y\_test,y\_pred))

from sklearn.metrics import confusion\_matrix

print(confusion\_matrix(y\_test,y\_pred))

import sklearn.metrics as metrics

fpr, tpr ,threshold = metrics.roc\_curve(y\_test,y\_pred)

roc\_auc = metrics.auc(fpr,tpr)

print(roc\_auc)

plt.plot(fpr,tpr)

plt.xlim([0,1])

plt.ylim([0,1])

plt.style.use("dark\_background")

from sklearn.tree import export\_graphviz

export\_graphviz(dt, out\_file ='tree.dot',

feature\_names = ["Gender","Age","Salary"], class\_names = ['0','1'],

rounded = True, proportion = False, precision = 2, filled = True)

**OUTPUT:-**

User ID Gender Age EstimatedSalary Purchased

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

User ID False

Gender False

Age False

EstimatedSalary False

Purchased False

dtype: bool

[['Male' 19 19000]

['Male' 35 20000]

['Female' 26 43000]

...

['Female' 50 20000]

['Male' 36 33000]

['Female' 49 36000]]

[[0]

[0]

[0]

[0]

[0]

[0]

[0]

[1]

[0]

[0]]

[[1 19 19000]

[1 35 20000]

[0 26 43000]

...

[0 50 20000]

[1 36 33000]

[0 49 36000]]

[[1 27 88000]

[1 41 52000]

[0 27 84000]

...

[1 36 52000]

[0 27 54000]

[0 26 118000]]

[[1 30 87000]

[0 38 50000]

[1 35 75000]

[0 30 79000]

[0 35 50000]

[1 27 20000]

[0 31 15000]

[1 36 144000]

[0 18 68000]

[1 47 43000]]

DecisionTreeClassifier(criterion='entropy')

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0

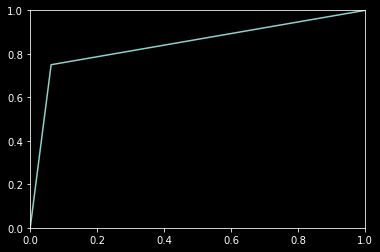
0 0 1]

0.9

[[30 2]

[ 2 6]]

0.84375



**Random Forest Classification:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv(“D:\\ML\\Datasets\\ads.csv")

print(dataset.head())

print(dataset.isnull().any())

x = dataset.iloc[:, 1:4].values

y = dataset.iloc[:, 4].values

print(x[:5])

print(y[:5])

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

x[:,0]=lb.fit\_transform(x[:,0])

print(x[:5])

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.1, random\_state =0)

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n\_estimators=10000,criterion='entropy')

print(rf.fit(x\_train,y\_train))

y\_pred=rf.predict(x\_test)

print(y\_pred)

from sklearn.metrics import accuracy\_score

print("Accuracy Score: ",accuracy\_score(y\_test,y\_pred)\*100,"%")

from sklearn.metrics import confusion\_matrix

print(pd.DataFrame(confusion\_matrix(y\_test,y\_pred),columns=["Prediction -0","Prediction-1"]))

import sklearn.metrics as metrics

fpr, tpr, threshold = metrics.roc\_curve(y\_test, y\_pred)

roc\_auc = metrics.auc(fpr, tpr)

print("AUC:",roc\_auc)

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True PositiveRate')

plt.xlabel('False Positive Rate')

plt.show()

**OUTPUT:-**

User ID Gender Age EstimatedSalary Purchased

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

User ID False

Gender False

Age False

EstimatedSalary False

Purchased False

dtype: bool

[['Male' 19 19000]

['Male' 35 20000]

['Female' 26 43000]

['Female' 27 57000]

['Male' 19 76000]]

[0 0 0 0 0]

[[1 19 19000]

[1 35 20000]

[0 26 43000]

[0 27 57000]

[1 19 76000]]

RandomForestClassifier(criterion='entropy', n\_estimators=10000)

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0

0 0 1]

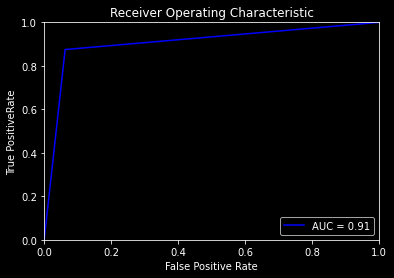
Accuracy Score: 92.5 %

Prediction -0 Prediction-1

0 30 2

1 1 7

AUC: 0.90625



**Decision Tree Based Algorithm For Classification:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv(“D:\\ML\\Datasets\\ads.csv")

print(data.head())

print(data.isnull().any())

x = data.iloc[:,1:4].values

y = data.iloc[:,4:5].values

print(x)

print(y[:10])

from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

x[:,0] = lb.fit\_transform(x[:,0])

print(x)

from sklearn.model\_selection import train\_test\_split as tts

x\_train,x\_test,y\_train,y\_test = tts(x, y, test\_size = 0.1,random\_state=0)

print(x\_train)

print(x\_test[:10])

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='entropy')

print(dt.fit(x\_train,y\_train))

y\_pred = dt.predict(x\_test)

print(y\_pred)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(y\_test,y\_pred))

from sklearn.metrics import confusion\_matrix

print(confusion\_matrix(y\_test,y\_pred))

import sklearn.metrics as metrics

fpr, tpr ,threshold = metrics.roc\_curve(y\_test,y\_pred)

roc\_auc = metrics.auc(fpr,tpr)

print(roc\_auc)

plt.plot(fpr,tpr)

plt.xlim([0,1])

plt.ylim([0,1])

plt.style.use("dark\_background")

from sklearn.tree import export\_graphviz

export\_graphviz(dt, out\_file ='tree.dot',

feature\_names = ["Gender","Age", "Salary"], class\_names = ['0','1'],

rounded = True, proportion = False, precision = 2, filled = True)

**OUTPUT:-**

User ID Gender Age EstimatedSalary Purchased

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

User ID False

Gender False

Age False

EstimatedSalary False

Purchased False

dtype: bool

[['Male' 19 19000]

['Male' 35 20000]

['Female' 26 43000]

...

['Female' 50 20000]

['Male' 36 33000]

['Female' 49 36000]]

[[0]

[0]

[0]

[0]

[0]

[0]

[0]

[1]

[0]

[0]]

[[1 19 19000]

[1 35 20000]

[0 26 43000]

...

[0 50 20000]

[1 36 33000]

[0 49 36000]]

[[1 27 88000]

[1 41 52000]

[0 27 84000]

...

[1 36 52000]

[0 27 54000]

[0 26 118000]]

[[1 30 87000]

[0 38 50000]

[1 35 75000]

[0 30 79000]

[0 35 50000]

[1 27 20000]

[0 31 15000]

[1 36 144000]

[0 18 68000]

[1 47 43000]]

DecisionTreeClassifier(criterion='entropy')

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0

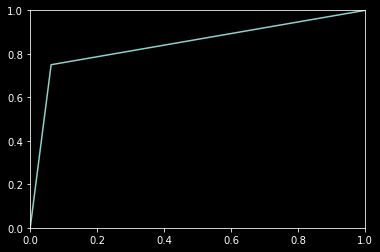
0 0 1]

0.9

[[30 2]

[ 2 6]]

0.84375



**Artificial Neural Network:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('D:\\ML\\Datasets\\Churn\_Modelling.csv')

print(dataset.head())

print(dataset.isnull().any())

X = dataset.iloc[:, 3:13].values

y = dataset.iloc[:, 13].values

print(X[:5])

print(y[:5])

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.compose import ColumnTransformer

labelencoder\_X\_1 =LabelEncoder()

X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])

labelencoder\_X\_2 =LabelEncoder()

X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])

#onehotencoder = OneHotEncoder(categorical\_features = [1])

ct = ColumnTransformer([("Surname", OneHotEncoder(),[1])], remainder = 'passthrough')

X = ct.fit\_transform(X)

X = X[:, 1:]

print("X->{}".format(X))

print('\n')

print("y->{}".format(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)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

X\_train=sc.fit\_transform(X\_train)

X\_test=sc.transform(X\_test)

from keras.models import Sequential

from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(units = 6, kernel\_initializer= "uniform", activation = "relu", input\_dim = 11))

classifier.add(Dense(units = 6, kernel\_initializer = "uniform", activation = "relu"))

classifier.add(Dense(units = 1, kernel\_initializer= "uniform", activation = "sigmoid"))

classifier.compile(optimizer = "adam", loss = "binary\_crossentropy", metrics = ["accuracy"])

classifier.fit(X\_train, y\_train, batch\_size = 32, epochs = 100)

y\_pred = classifier.predict(X\_test)

print(y\_pred)

y\_pred = (y\_pred > 0.5)

print(y\_pred)

from sklearn.metrics import accuracy\_score

print("Accuracy score",accuracy\_score(y\_test,y\_pred)\*100,"%")

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

**OUTPUT:-**

RowNumber CustomerId Surname ... IsActiveMember EstimatedSalary Exited

0 1 15634602 Hargrave ... 1 101348.88 1

1 2 15647311 Hill ... 1 112542.58 0

2 3 15619304 Onio ... 0 113931.57 1

3 4 15701354 Boni ... 0 93826.63 0

4 5 15737888 Mitchell ... 1 79084.10 0

[5 rows x 14 columns]

RowNumber False

CustomerId False

Surname False

CreditScore False

Geography False

Gender False

Age False

Tenure False

Balance False

NumOfProducts False

HasCrCard False

IsActiveMember False

EstimatedSalary False

Exited False

dtype: bool

[[619 'France' 'Female' 42 2 0.0 1 1 1 101348.88]

[608 'Spain' 'Female' 41 1 83807.86 1 0 1 112542.58]

[502 'France' 'Female' 42 8 159660.8 3 1 0 113931.57]

[699 'France' 'Female' 39 1 0.0 2 0 0 93826.63]

[850 'Spain' 'Female' 43 2 125510.82 1 1 1 79084.1]]

[1 0 1 0 0]

[[0.23952079]

[0.3925631 ]

[0.12168884]

...

[0.17557025]

[0.16840512]

[0.1797044 ]]

[[False]

[False]

[False]

...

[False]

[False]

[False]]

Accuracy score 85.75 %

**K-Means Algorithm:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('D:\\ML\\Datasets\\Mall\_Customers.csv')

print(dataset.head())

print(dataset.isnull().any())

x = dataset.iloc[:,[3,4]].values

print(x[:5])

from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', max\_iter=300,

n\_init=10, random\_state=0)

kmeans.fit(x)

wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans = KMeans(n\_clusters=5, init = 'k-means++', max\_iter=300, n\_init = 10, random\_state=0)

y\_kmeans = kmeans.fit\_predict(x)

print(y\_kmeans[:5])

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='yellow',label='cluster 4')

plt.scatter(x[y\_kmeans == 4,0],x[y\_kmeans == 4,1],s=100,c='brown',label='cluster 5')

plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=300,c='black',label='centroids')

plt.title('Clusters of clients')

plt.xlabel("Annual Income in 1000 $")

plt.ylabel("Spending Score (1-1000")

plt.legend()

plt.show()

**OUTPUT:-**

CustomerID Gender Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19 15 39

1 2 Male 21 15 81

2 3 Female 20 16 6

3 4 Female 23 16 77

4 5 Female 31 17 40

CustomerID False

Gender False

Age False

Annual Income (k$) False

Spending Score (1-100) False

dtype: bool

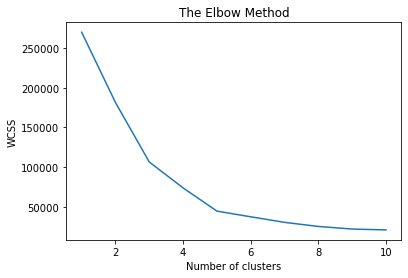
[[15 39]

[15 81]

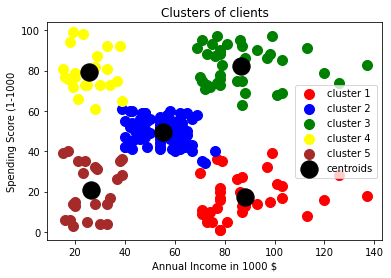
[16 6]

[16 77]

[17 40]]



[4 3 4 3 4]



**Hierarchical Clustering:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('D:\\ML\\Datasets\\Mall\_Customers.csv')

print(dataset.head())

print(dataset.isnull().any())

x = dataset.iloc[:, [3, 4]].values

print(x[:5])

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(x, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(x)

plt.scatter(x[y\_hc==0,0], x[y\_hc==0,1], s=100, color='red', label = 'Cluster 1')

plt.scatter(x[y\_hc==1,0], x[y\_hc==1,1], s=100, color='blue', label = 'Cluster 2')

plt.scatter(x[y\_hc==2,0], x[y\_hc==2,1], s=100, color='green', label = 'Cluster 3')

plt.scatter(x[y\_hc==3,0], x[y\_hc==3,1], s=100, color='cyan', label = 'Cluster 4')

plt.scatter(x[y\_hc==4,0], x[y\_hc==4,1], s=100, color='magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**OUTPUT:-**

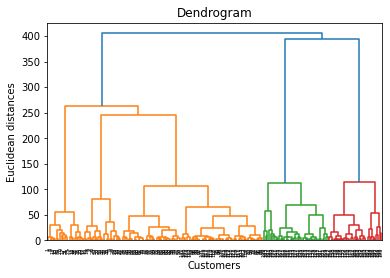
[[15 39]

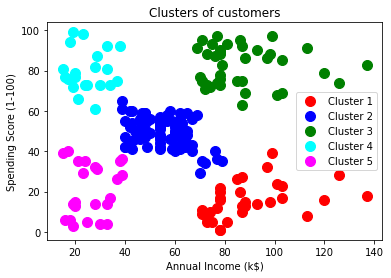
[15 81]

[16 6]

[16 77]

[17 40]]





**Apriori Algorithm:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('D:\\ML\\Datasets\\store\_data.csv',header = None)

print(dataset.head())

x = dataset.iloc[:, 0:].values

print(x)

transactions = []

for i in range(0, 7501):

transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

transactions[:2]

from apyori import apriori

rules = apriori(transactions, min\_support = 0.0045, min\_confidence = 0.2, min\_lift = 3, min\_length=2)

results = list(rules)

print(results)

**OUTPUT:-**

0 1 2 ... 17 18 19

0 shrimp almonds avocado ... frozen smoothie spinach olive oil

1 burgers meatballs eggs ... NaN NaN NaN

2 chutney NaN NaN ... NaN NaN NaN

3 turkey avocado NaN ... NaN NaN NaN

4 mineral water milk energy bar ... NaN NaN NaN

[5 rows x 20 columns]

[['shrimp' 'almonds' 'avocado' ... 'frozen smoothie' 'spinach'

'olive oil']

[RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'light cream'}), items\_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.84395061728395)]), RelationRecord(items=frozenset({'escalope', 'mushroom cream sauce'}), support=0.005732568990801226, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'mushroom cream sauce'}), items\_add=frozenset({'escalope'}), confidence=0.3006993006993007,