**Code**

**Hotspot Analysis**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import seaborn as sns; sns.set()

import csv

import math

df = pd.read\_excel (r'C:\Users\Siddhartha Kaushik\OneDrive\Desktop\book2.xlsx')

print (df)

df.dropna(axis=0,how='any',subset=['latitude','longitude'],inplace=True)

X=df.loc[:,['accident\_severity','latitude','longitude']]

X.head(10)

K\_clusters = range(1,10)

kmeans = [KMeans(n\_clusters=i) for i in K\_clusters]

Y\_axis = df[['latitude']]

X\_axis = df[['longitude']]

score = [kmeans[i].fit(Y\_axis).score(Y\_axis) for i in range(len(kmeans))]

plt.plot(K\_clusters, score)

plt.xlabel('Number of Clusters')

plt.ylabel('Score')

plt.title('Elbow Curve')

plt.show()

kmeans = KMeans(n\_clusters = 3, init ='k-means++')

kmeans.fit(X[X.columns[1:3]])

X['cluster\_label'] = kmeans.fit\_predict(X[X.columns[1:3]])

centers = kmeans.cluster\_centers\_

labels = kmeans.predict(X[X.columns[1:3]])

data\_with\_clusters = X.copy()

data\_with\_clusters['Clusters'] = X['cluster\_label']

plt.scatter(data\_with\_clusters['longitude'],data\_with\_clusters['latitude'],c=data\_with\_clusters['cluster\_label'],cmap='rainbow')

print (X)

print (data\_with\_clusters)

array=data\_with\_clusters[["accident\_severity","Clusters"]].to\_numpy()

print (array)

n=len(array)

p1=0

p2=0

p3=0

q1=0

q2=0

q3=0

r1=0

r2=0

r3=0

for i in range(n):

x=array[i,0]

y=array[i,1]

if (y==1):

if (x==1):

p1+=1

elif (x==2):

q1+=1

elif (x==3):

r1+=1

elif (y==2):

if (x==1):

p2+=1

elif (x==2):

q2+=1

elif (x==3):

r2+=1

elif (y==2):

if (x==1):

p3+=1

elif (x==2):

q3+=1

elif (x==3):

r3+=1

si=[]

si.append(3\*p1+1.8\*q1+1.3\*r1)

si.append(3\*p2+1.8\*q2+1.3\*r2)

si.append(3\*p3+1.8\*q3+1.3\*r3)

print (si)

n=len(data\_with\_clusters.Clusters)

print (n)

print (si)

point\_0= (51.462262,-0.254001 )

point\_1= (51.470327, -0.139253)

point\_2= (51.529614, -0.178719)

point\_3= (51.541210 ,-0.001683 )

point\_4= (51.515704 ,-0.137592)

point\_5= (51.476278 ,-0.025880)

point\_6= (51.494780 ,0.011959)

point\_7= (51.523195 ,-0.039390)

point\_8= (51.605653 ,-0.111500 )

point\_9= (51.614971 , -0.127840)

def naive\_euclidian\_distance(point1, point2):

differences = [point1[x] - point2[x] for x in range(len(point1))]

differences\_squared = [difference \*\* 2 for difference in differences]

sum\_of\_squares = sum(differences\_squared)

return sum\_of\_squares \*\* 0.5

s0=naive\_euclidian\_distance(point\_3, point\_5)+naive\_euclidian\_distance(point\_3, point\_6)+naive\_euclidian\_distance(point\_3, point\_7)+naive\_euclidian\_distance(point\_5, point\_6)+naive\_euclidian\_distance(point\_5, point\_7)+naive\_euclidian\_distance(point\_6, point\_7)

print (s0)

s1=naive\_euclidian\_distance(point\_0, point\_1)+naive\_euclidian\_distance(point\_0, point\_2)+naive\_euclidian\_distance(point\_0, point\_4)+naive\_euclidian\_distance(point\_1, point\_2)+naive\_euclidian\_distance(point\_2, point\_4)

print (s1)

s2=naive\_euclidian\_distance(point\_8, point\_9)

print (s2)

g0=(s0\*0.30910647736006613)/10

print (g0)

g1=(s1\*0.4587717352787306)/12

print (g1)

g2=(s2\*0.018810122912942578)/6

print (g2)

import statistics

import numpy as np

def expected\_value(values, weights):

values = np.asarray(values)

weights = np.asarray(weights)

return (values \* weights).sum() / weights.sum()

dataset=[0.009554681434594908,0.01753929209088814, 5.8970120666667894e-05]

output=statistics.variance(dataset)

var=math.sqrt(output)

print (var)

dataset=[0.009554681434594908,0.01753929209088814, 5.8970120666667894e-05]

probs=[0.4,0.4,0.2]

expected= expected\_value(dataset,probs)

print (expected)

z0=(g0-expected)/var

print (z0)

z1=(g1-expected)/var

print (z1)

z2=(g2-expected)/var

print (z2)

z=[-0.14794835965560732, 0.7644701345864018, -1.2330435498615893]

assignments = vq( z, accident\_severity )[0]

clusters = [[] for i in range( len( assignments ) )

for item,clustNum in zip( dataset, assignments):

clusters[clustNum].append( item )

**Decision Tree**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import seaborn as sns; sns.set()

import csv

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import metrics

inputs=df.drop('accident\_severity', axis='columns')

target = df['accident\_severity']

inputs

from sklearn.preprocessing import LabelEncoder

le\_vehiclno = LabelEncoder()

le\_casualtyno = LabelEncoder()

le\_day= LabelEncoder()

le\_road\_type= LabelEncoder()

le\_speed\_limit= LabelEncoder()

le\_lights= LabelEncoder()

le\_weather= LabelEncoder()

le\_road= LabelEncoder()

le\_areas= LabelEncoder()

inputs['vehiclno\_n']= le\_vehiclno.fit\_transform(inputs['vehiclno'])

inputs['casualtyno\_n']= le\_casualtyno.fit\_transform(inputs['casualtyno'])

inputs['day\_n']= le\_day.fit\_transform(inputs['day'])

inputs['road\_type\_n']= le\_road\_type.fit\_transform(inputs['road\_type'])

inputs['speed\_limit\_n']= le\_speed\_limit.fit\_transform(inputs['speed\_limit'])

inputs['lights\_n']= le\_lights.fit\_transform(inputs['lights'])

inputs['weather\_n']= le\_weather.fit\_transform(inputs['weather'])

inputs['road\_n']= le\_road.fit\_transform(inputs['road'])

inputs['area\_n']= le\_areas.fit\_transform(inputs['area'])

inputs.head()

inputs\_n = inputs.drop(['vehiclno','casualtyno','day','road\_type','speed\_limit','lights','weather','road','area'], axis= 'columns')

inputs\_n

from sklearn import tree

model= tree.DecisionTreeClassifier()

model.fit(inputs\_n,target)

model.score(inputs\_n,target)

model.predict([[0,0,0,0,0,0,0,5,1]])

pima = pd.read\_excel (r'C:\Users\Siddhartha Kaushik\OneDrive\Desktop\Book1.xlsx')

pima.head()

feature\_cols = ['vehiclno','casualtyno','day','road\_type','speed\_limit','lights','weather','road','area']

X = pima[feature\_cols]

y = pima.accident\_severity

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1) # 70% training and 30% test

clf = DecisionTreeClassifier()

clf = clf.fit(X\_train,y\_train)

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

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

pip install graphviz

pip install pydotplus

pip install six

clf = DecisionTreeClassifier()

clf = clf.fit(X\_train,y\_train)

clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)

clf = clf.fit(X\_train,y\_train)

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

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

import graphviz

dot\_data = tree.export\_graphviz(clf, out\_file=None,

feature\_names=inputs\_n,

class\_names=df.accident\_severity,

filled=True)

graph = graphviz.Source(dot\_data, format="png")

graph