- Name : P.Sai Venkatesh

- course : python
- project : Developing a machine learning model and performing Feature Detection task by handling the realtime dataset to bring best possible accuracy outcome.

Feature Detection, Classification and Identification

-- Diabetes dataset --

Performing the accuracy task using

decision tree classifier

In [33]:

```
# Import the libraries
import pandas as pd
```

In [34]:

```
# Import the dataset
dataset = pd.read_csv(r"C:\Project-Dataset.csv")
```

In [35]:

```
# checking the dataset whether the dataset is empty or not
dataset.isnull().any()
```

Out[35]:

```
Age                         False
BMI                         False
Pregnancies                 False
BloodPressure               False
Glucose                     False
Insulin                     False
SkinThickness               False
Alzheimer                   False
Vision                      False
DiabetesPedigreeFunction    False
Alcoholic/Smoker            False
Outcome                     False
dtype: bool
```

In [4]:

```
#checking weather the dataset is numerical or not
dataset
```

Out[4]:

	Age	BMI	Pregnancies	BloodPressure	Glucose	Insulin	SkinThickness	Alzheimer	Vision
0	50	33.6	6	72	148	0	35	Yes	1.2
1	31	26.6	1	66	85	0	29	No	0.7
2	32	23.3	8	64	183	0	0	Yes	2.7
3	21	28.1	1	66	89	94	23	No	1.4
4	33	43.1	0	40	137	168	35	Yes	3.2
5	30	25.6	5	74	116	0	0	No	0.4
6	26	31.0	3	50	78	88	32	Yes	3.5
7	29	35.3	10	0	115	0	0	No	1.4
8	53	30.5	2	70	197	543	45	No	3.2
9	54	0.0	8	96	125	0	0	Yes	2.8
10	30	37.6	4	92	110	0	0	Yes	0.8
11	34	38.0	10	74	168	0	0	No	1.9
12	57	27.1	10	80	139	0	0	Yes	1.1
13	59	30.1	1	60	189	846	23	Yes	4.6
14	51	25.8	5	72	166	175	19	Yes	4.1
15	32	30.0	7	0	100	0	0	No	3.7
16	31	45.8	0	84	118	230	47	Yes	3.2
17	31	29.6	7	74	107	0	0	No	4.8
18	33	43.3	1	30	103	83	38	No	2.0

In [36]:

```
# Describimg the dataset
dataset.describe()
```

Out[36]:

	Age	BMI	Pregnancies	BloodPressure	Glucose	Insulin	SkinThicknes
count	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000	19.00000
mean	37.736842	30.757895	4.684211	61.263158	130.157895	117.210526	17.1578
std	11.836113	9.879626	3.559848	26.801414	36.155933	221.704904	17.9791
min	21.000000	0.000000	0.000000	0.000000	78.000000	0.000000	0.00000
25%	30.500000	26.850000	1.000000	55.000000	105.000000	0.000000	0.00000
50%	32.000000	30.100000	5.000000	70.000000	118.000000	0.000000	19.00000
75%	50.500000	36.450000	7.500000	74.000000	157.000000	131.000000	33.50000
max	59.000000	45.800000	10.000000	96.000000	197.000000	846.000000	47.00000

To find accuracy of the dataset the values of dataset must be in numerical form.

In [5]:

```
# convering charectors into numeric

m={'Yes': 1, 'No': 0}
dataset["Alcoholic/Smoker"]=dataset["Alcoholic/Smoker"].map(m)

n={'Yes': 1, 'No': 0}
dataset["Alzheimer"]=dataset["Alzheimer"].map(n)
```

In [16]:

```
# spliting the dataset into dependent and independent values by using  "iloc" Function

x=dataset.iloc[:, 0:10].values
y=dataset.iloc[:,10].values
```

In [18]:

```
print("x values")
print(" ")
print(x)
print(" ")
print("y values")
print(" ")
print(y)
```

```
x values

[[5.000e+01 3.360e+01 6.000e+00 7.200e+01 1.480e+02 0.000e+00 3.500e+01
  1.000e+00 1.200e+00 6.270e-01]
 [3.100e+01 2.660e+01 1.000e+00 6.600e+01 8.500e+01 0.000e+00 2.900e+01
  0.000e+00 7.000e-01 3.510e-01]
 [3.200e+01 2.330e+01 8.000e+00 6.400e+01 1.830e+02 0.000e+00 0.000e+00
  1.000e+00 2.700e+00 6.720e-01]
 [2.100e+01 2.810e+01 1.000e+00 6.600e+01 8.900e+01 9.400e+01 2.300e+01
  0.000e+00 1.400e+00 1.670e-01]
 [3.300e+01 4.310e+01 0.000e+00 4.000e+01 1.370e+02 1.680e+02 3.500e+01
  1.000e+00 3.200e+00 2.288e+00]
 [3.000e+01 2.560e+01 5.000e+00 7.400e+01 1.160e+02 0.000e+00 0.000e+00
  0.000e+00 4.000e-01 2.010e-01]
 [2.600e+01 3.100e+01 3.000e+00 5.000e+01 7.800e+01 8.800e+01 3.200e+01
  1.000e+00 3.500e+00 2.480e-01]
 [2.900e+01 3.530e+01 1.000e+01 0.000e+00 1.150e+02 0.000e+00 0.000e+00
  0.000e+00 1.400e+00 1.340e-01]
 [5.300e+01 3.050e+01 2.000e+00 7.000e+01 1.970e+02 5.430e+02 4.500e+01
  0.000e+00 3.200e+00 1.580e-01]
 [5.400e+01 0.000e+00 8.000e+00 9.600e+01 1.250e+02 0.000e+00 0.000e+00
  1.000e+00 2.800e+00 2.320e-01]
 [3.000e+01 3.760e+01 4.000e+00 9.200e+01 1.100e+02 0.000e+00 0.000e+00
  1.000e+00 8.000e-01 1.910e-01]
 [3.400e+01 3.800e+01 1.000e+01 7.400e+01 1.680e+02 0.000e+00 0.000e+00
  0.000e+00 1.900e+00 5.370e-01]
 [5.700e+01 2.710e+01 1.000e+01 8.000e+01 1.390e+02 0.000e+00 0.000e+00
  1.000e+00 1.100e+00 1.441e+00]
 [5.900e+01 3.010e+01 1.000e+00 6.000e+01 1.890e+02 8.460e+02 2.300e+01
  1.000e+00 4.600e+00 3.980e-01]
 [5.100e+01 2.580e+01 5.000e+00 7.200e+01 1.660e+02 1.750e+02 1.900e+01
  1.000e+00 4.100e+00 5.870e-01]
 [3.200e+01 3.000e+01 7.000e+00 0.000e+00 1.000e+02 0.000e+00 0.000e+00
  0.000e+00 3.700e+00 4.840e-01]
 [3.100e+01 4.580e+01 0.000e+00 8.400e+01 1.180e+02 2.300e+02 4.700e+01
  1.000e+00 3.200e+00 5.510e-01]
 [3.100e+01 2.960e+01 7.000e+00 7.400e+01 1.070e+02 0.000e+00 0.000e+00
  0.000e+00 4.800e+00 2.540e-01]
 [3.300e+01 4.330e+01 1.000e+00 3.000e+01 1.030e+02 8.300e+01 3.800e+01
  0.000e+00 2.000e+00 1.830e-01]]

y values

[1 0 1 1 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1]
```

In [15]:

```
# rechecking the dataset weather it is numeric or not

dataset
```

Out[15]:

	Age	BMI	Pregnancies	BloodPressure	Glucose	Insulin	SkinThickness	Alzheimer	Vision
0	50	33.6	6	72	148	0	35	1	1.2
1	31	26.6	1	66	85	0	29	0	0.7
2	32	23.3	8	64	183	0	0	1	2.7
3	21	28.1	1	66	89	94	23	0	1.4
4	33	43.1	0	40	137	168	35	1	3.2
5	30	25.6	5	74	116	0	0	0	0.4
6	26	31.0	3	50	78	88	32	1	3.5
7	29	35.3	10	0	115	0	0	0	1.4
8	53	30.5	2	70	197	543	45	0	3.2
9	54	0.0	8	96	125	0	0	1	2.8
10	30	37.6	4	92	110	0	0	1	0.8
11	34	38.0	10	74	168	0	0	0	1.9
12	57	27.1	10	80	139	0	0	1	1.1
13	59	30.1	1	60	189	846	23	1	4.6
14	51	25.8	5	72	166	175	19	1	4.1
15	32	30.0	7	0	100	0	0	0	3.7
16	31	45.8	0	84	118	230	47	1	3.2
17	31	29.6	7	74	107	0	0	0	4.8
18	33	43.3	1	30	103	83	38	0	2.0

In [23]:

```
# normalising the values of dataset.

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

# Fitting the dataset

x=sc.fit_transform(x)
x
```

Out[23]:

```
array([[ 1.06447085,  0.29555628,  0.37974805,  0.41158488,  0.50699897,
        -0.54316511,  1.01956847,  0.9486833 , -0.94751788,  0.22633617],
       [-0.58477368, -0.43238788, -1.06329455,  0.18158156, -1.28320093,
        -0.54316511,  0.67670474, -1.05409255, -1.32414633, -0.31096889],
       [-0.49797134, -0.77556156,  0.95696509,  0.10491379,  1.50155447,
        -0.54316511, -0.98046997,  0.9486833 ,  0.18236746,  0.31394026],
       [-1.45279712, -0.27639985, -1.06329455,  0.18158156, -1.16953745,
        -0.10755986,  0.333841  , -1.05409255, -0.79686651, -0.66917226],
       [-0.411169  ,  1.28348049, -1.35190307, -0.81509946,  0.19442438,
         0.23536342,  1.01956847,  0.9486833 ,  0.55899591,  3.45990032],
       [-0.67157603, -0.53637991,  0.09113953,  0.48825265, -0.40230892,
        -0.54316511, -0.98046997, -1.05409255, -1.5501234 , -0.60298251],
       [-1.0187854 ,  0.02517702, -0.48607751, -0.43176061, -1.48211203,
        -0.13536445,  0.8481366 ,  0.9486833 ,  0.78497298, -0.51148491],
       [-0.75837837,  0.47234271,  1.53418213, -2.34845488, -0.43072479,
        -0.54316511, -0.98046997, -1.05409255, -0.79686651, -0.73341526],
       [ 1.32487788, -0.026819  , -0.77468603,  0.33491711,  1.89937667,
         1.97315031,  1.59100802, -1.05409255,  0.55899591, -0.68669308],
       [ 1.41168022, -3.19857569,  0.95696509,  1.33159813, -0.14656607,
        -0.54316511, -0.98046997,  0.9486833 ,  0.25769315, -0.54263303],
       [-0.67157603,  0.71152437, -0.19746899,  1.17826259, -0.57280415,
        -0.54316511, -0.98046997,  0.9486833 , -1.24882064, -0.62245008],
       [-0.32436665,  0.75312118,  1.53418213,  0.48825265,  1.0753164 ,
        -0.54316511, -0.98046997, -1.05409255, -0.42023806,  0.051128  ],
       [ 1.67208725, -0.38039187,  1.53418213,  0.71825596,  0.25125613,
        -0.54316511, -0.98046997,  0.9486833 , -1.02284357,  1.81099675],
       [ 1.84569194, -0.0684158 , -1.06329455, -0.04842175,  1.6720497 ,
         3.37728212,  0.333841  ,  0.9486833 ,  1.61355556, -0.21947129],
       [ 1.15127319, -0.5155815 ,  0.09113953,  0.41158488,  1.01848466,
         0.26780211,  0.10526518,  0.9486833 ,  1.23692711,  0.14846587],
       [-0.49797134, -0.07881501,  0.66835657, -2.34845488, -0.85696286,
        -0.54316511, -0.98046997, -1.05409255,  0.93562435, -0.05205015],
       [-0.58477368,  1.56425896, -1.35190307,  0.8715915 , -0.34547717,
         0.52267752,  1.70529594,  0.9486833 ,  0.55899591,  0.0783826 ],
       [-0.58477368, -0.12041182,  0.66835657,  0.48825265, -0.65805176,
        -0.54316511, -0.98046997, -1.05409255,  1.76420694, -0.49980436],
       [-0.411169  ,  1.3042789 , -1.06329455, -1.19843832, -0.77171525,
        -0.15853494,  1.19100034, -1.05409255, -0.34491237, -0.63802414]])
```

In [10]:

```
# training and testing the dataset using "train-test-split" function.

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)
```

In [26]:

```
x_train
```

Out[26]:

```
array([[-0.58477368,  1.56425896, -1.35190307,  0.8715915 , -0.34547717,
         0.52267752,  1.70529594,  0.9486833 ,  0.55899591,  0.0783826 ],
       [-1.0187854 ,  0.02517702, -0.48607751, -0.43176061, -1.48211203,
        -0.13536445,  0.8481366 ,  0.9486833 ,  0.78497298, -0.51148491],
       [-0.411169  ,  1.28348049, -1.35190307, -0.81509946,  0.19442438,
         0.23536342,  1.01956847,  0.9486833 ,  0.55899591,  3.45990032],
       [-0.49797134, -0.77556156,  0.95696509,  0.10491379,  1.50155447,
        -0.54316511, -0.98046997,  0.9486833 ,  0.18236746,  0.31394026],
       [-0.67157603, -0.53637991,  0.09113953,  0.48825265, -0.40230892,
        -0.54316511, -0.98046997, -1.05409255, -1.5501234 , -0.60298251],
       [ 1.84569194, -0.0684158 , -1.06329455, -0.04842175,  1.6720497 ,
         3.37728212,  0.333841  ,  0.9486833 ,  1.61355556, -0.21947129],
       [ 1.41168022, -3.19857569,  0.95696509,  1.33159813, -0.14656607,
        -0.54316511, -0.98046997,  0.9486833 ,  0.25769315, -0.54263303],
       [-0.75837837,  0.47234271,  1.53418213, -2.34845488, -0.43072479,
        -0.54316511, -0.98046997, -1.05409255, -0.79686651, -0.73341526],
       [-0.58477368, -0.12041182,  0.66835657,  0.48825265, -0.65805176,
        -0.54316511, -0.98046997, -1.05409255,  1.76420694, -0.49980436],
       [-0.32436665,  0.75312118,  1.53418213,  0.48825265,  1.0753164 ,
        -0.54316511, -0.98046997, -1.05409255, -0.42023806,  0.051128  ],
       [-1.45279712, -0.27639985, -1.06329455,  0.18158156, -1.16953745,
        -0.10755986,  0.333841  , -1.05409255, -0.79686651, -0.66917226],
       [ 1.06447085,  0.29555628,  0.37974805,  0.41158488,  0.50699897,
        -0.54316511,  1.01956847,  0.9486833 , -0.94751788,  0.22633617],
       [-0.49797134, -0.07881501,  0.66835657, -2.34845488, -0.85696286,
        -0.54316511, -0.98046997, -1.05409255,  0.93562435, -0.05205015],
       [ 1.67208725, -0.38039187,  1.53418213,  0.71825596,  0.25125613,
        -0.54316511, -0.98046997,  0.9486833 , -1.02284357,  1.81099675]])
```

In [28]:

```
y_train
```

Out[28]:

```
array([0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0], dtype=int64)
```

In [27]:

```
x_test
```

Out[27]:

```
array([[-0.67157603,  0.71152437, -0.19746899,  1.17826259, -0.57280415,
        -0.54316511, -0.98046997,  0.9486833 , -1.24882064, -0.62245008],
       [-0.58477368, -0.43238788, -1.06329455,  0.18158156, -1.28320093,
        -0.54316511,  0.67670474, -1.05409255, -1.32414633, -0.31096889],
       [ 1.32487788, -0.026819  , -0.77468603,  0.33491711,  1.89937667,
         1.97315031,  1.59100802, -1.05409255,  0.55899591, -0.68669308],
       [-0.411169  ,  1.3042789 , -1.06329455, -1.19843832, -0.77171525,
        -0.15853494,  1.19100034, -1.05409255, -0.34491237, -0.63802414],
       [ 1.15127319, -0.5155815 ,  0.09113953,  0.41158488,  1.01848466,
         0.26780211,  0.10526518,  0.9486833 ,  1.23692711,  0.14846587]])
```

In [29]:

```
y_test
```

Out[29]:

```
array([0, 0, 0, 1, 0], dtype=int64)
```

In [88]:

```
# Implementing decision tree classifier

from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier(criterion="entropy",random_state=0)
```

In [89]:

```
# Fitting the model

c.fit(x_train,y_train)
```

Out[89]:

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [90]:

```
# Predicting the outcome

y_pred=c.predict(x_test)
y_pred
```

Out[90]:

```
array([0, 0, 1, 1, 1], dtype=int64)
```

In [91]:

```
# confusion matrix

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print("confusion matrix")
print(cm)
```

```
confusion matrix
[[2 2]
 [0 1]]
```

In [92]:

```
# Testing the accuracy

print("By using decision tree classifier we got,")

from sklearn.metrics import accuracy_score
print("Accuracy = {0} %".format(accuracy_score(y_test,y_pred)*100))
```

```
By using decision tree classifier we got,
Accuracy = 60.0 %
```

In [93]:

```
# Comparision of Real and Predicted Class values
Comparision_table = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
Comparision_table
```

Out[93]:

	Real_class	Predicted_class
0	0	0
1	0	0
2	0	1
3	1	1
4	0	1

We can observe that from comparision table 3 of the rows are matched among 5.Hence,we got an accuracy of 60%.

As we know that if a machine learning model gets accuracy more than 50% then our model is correct. Hence, our machine learning model using decision tree classifier is correct.

Performing the accuracy task using

KNN classifier

In [47]:

```
# Import the libraries
import pandas as pd

# Import the dataset
dataset = pd.read_csv(r"C:\Project-Dataset.csv")

# checking the dataset whether the dataset is empty or not
dataset.isnull().any()

# convering charectors into numeric
m={'Yes': 1, 'No': 0}
dataset["Alcoholic/Smoker"]=dataset["Alcoholic/Smoker"].map(m)

n={'Yes': 1, 'No': 0}
dataset["Alzheimer"]=dataset["Alzheimer"].map(n)


# spliting the dataset into dependent and independent values by using  "iloc" Function
x=dataset.iloc[:, 0:10].values
y=dataset.iloc[:,10].values


# normalising the values of dataset.
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

# Fitting the dataset
x=sc.fit_transform(x)


# training and testing the dataset using "train-test-split" function.
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)


# Implementing KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
c=KNeighborsClassifier(n_neighbors=3)


# Fitting the model
c.fit(x_train,y_train)

# Predicting the outcome
y_pred=c.predict(x_test)

# confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print("confusion matrix")
print(cm)
print("")

print("By using KNeighborsClassifier we got,")

# Testing the accuracy
from sklearn.metrics import accuracy_score
print("Accuracy = {0} %".format(accuracy_score(y_test,y_pred)*100))
```

```
confusion matrix
[[0 4]
 [0 1]]

By using KNeighborsClassifier we got,
Accuracy = 20.0 %
```

In [39]:

```
# Comparision of Real and Predicted Class values
Comparision_table = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
Comparision_table
```

Out[39]:

	Real_class	Predicted_class
0	0	1
1	0	1
2	0	1
3	1	1
4	0	1

We can observe that from comparision table 1 of the rows are matched among 5.Hence,we got an accuracy of 20%.

As we know that if a machine learning model gets accuracy more than 50% then our model is correct. Hence, our machine learning model using KNN classifier is not correct.

Performing the accuracy task using

SVM(support vector machine) classifier

In [84]:

```
# Import the libraries
import pandas as pd

# Import the dataset
dataset = pd.read_csv(r"C:\Project-Dataset.csv")

# checking the dataset whether the dataset is empty or not
dataset.isnull().any()

# convering charectors into numeric
m={'Yes': 1, 'No': 0}
dataset["Alcoholic/Smoker"]=dataset["Alcoholic/Smoker"].map(m)

n={'Yes': 1, 'No': 0}
dataset["Alzheimer"]=dataset["Alzheimer"].map(n)


# spliting the dataset into dependent and independent values by using  "iloc" Function
x=dataset.iloc[:, 0:10].values
y=dataset.iloc[:,10].values


# normalising the values of dataset.
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

# Fitting the dataset
x=sc.fit_transform(x)


# training and testing the dataset using "train-test-split" function.
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)


# Implementing KNeighborsClassifier
from sklearn.svm import SVC
c=SVC(kernel="linear",random_state=0)


# Fitting the model
c.fit(x_train,y_train)

# Predicting the outcome
y_pred=c.predict(x_test)

# confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print("confusion matrix")
print(cm)
print("")

print("By using SVM(support vector machine) we got,")

# Testing the accuracy
from sklearn.metrics import accuracy_score
print("Accuracy = {0} %".format(accuracy_score(y_test,y_pred)*100))
```

```
confusion matrix
[[1 3]
 [0 1]]

By using SVM(support vector machine) we got,
Accuracy = 40.0 %
```

In [51]:

```
# Comparision of Real and Predicted Class values
Comparision_table = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
Comparision_table
```

Out[51]:

	Real_class	Predicted_class
0	0	0
1	0	1
2	0	1
3	1	1
4	0	1

We can observe that from comparision table 2 of the rows are matched among 5.Hence,we got an accuracy of 40%.

As we know that if a machine learning model gets accuracy more than 50% then our model is correct. Hence, our machine learning model using SVM(support vector machine) classifier is not correct.

To perform the task of feauture selection we need to calculate the "Correlation coefficient" values of the features in the dataset.

After calculating the Correlation coefficient values the dataset is named as Project-Dataset 1. The new dataset is displayed below.

In [64]:

```
# Import the libraries
import pandas as pd

# Import the dataset
dataset1 = pd.read_csv(r"C:\Project-Dataset 1.csv")

dataset1
```

Out[64]:

	Age	BMI	Pregnancies	BloodPressure	Glucose	Insulin	SkinThickness
0	50.000000	33.600000	6.00000	72.000000	148.000000	0.000000	35.000000
1	31.000000	26.600000	1.00000	66.000000	85.000000	0.000000	29.000000
2	32.000000	23.300000	8.00000	64.000000	183.000000	0.000000	0.000000
3	21.000000	28.100000	1.00000	66.000000	89.000000	94.000000	23.000000
4	33.000000	43.100000	0.00000	40.000000	137.000000	168.000000	35.000000
5	30.000000	25.600000	5.00000	74.000000	116.000000	0.000000	0.000000
6	26.000000	31.000000	3.00000	50.000000	78.000000	88.000000	32.000000
7	29.000000	35.300000	10.00000	0.000000	115.000000	0.000000	0.000000
8	53.000000	30.500000	2.00000	70.000000	197.000000	543.000000	45.000000
9	54.000000	0.000000	8.00000	96.000000	125.000000	0.000000	0.000000
10	30.000000	37.600000	4.00000	92.000000	110.000000	0.000000	0.000000
11	34.000000	38.000000	10.00000	74.000000	168.000000	0.000000	0.000000
12	57.000000	27.100000	10.00000	80.000000	139.000000	0.000000	0.000000
13	59.000000	30.100000	1.00000	60.000000	189.000000	846.000000	23.000000
14	51.000000	25.800000	5.00000	72.000000	166.000000	175.000000	19.000000
15	32.000000	30.000000	7.00000	0.000000	100.000000	0.000000	0.000000
16	31.000000	45.800000	0.00000	84.000000	118.000000	230.000000	47.000000
17	31.000000	29.600000	7.00000	74.000000	107.000000	0.000000	0.000000
18	33.000000	43.300000	1.00000	30.000000	103.000000	83.000000	38.000000
19	0.314036	-0.094116	0.02486	0.087174	0.477792	0.325354	0.187704

From the above dataset we consider first 4 features having highest correlation coefficien values.

New dataset is named as Project-Dataset 2. The new dataset is displayed below.

In [63]:

```
# Import the libraries
import pandas as pd

# Import the dataset
dataset2 = pd.read_csv(r"C:\Project-Dataset 2.csv")

dataset2
```

Out[63]:

	Glucose	Alzheimer	Vision	Alcoholic/Smoker	Outcome
0	148	Yes	1.2	Yes	1
1	85	No	0.7	No	0
2	183	Yes	2.7	Yes	1
3	89	No	1.4	Yes	0
4	137	Yes	3.2	Yes	1
5	116	No	0.4	No	0
6	78	Yes	3.5	Yes	1
7	115	No	1.4	No	0
8	197	No	3.2	No	1
9	125	Yes	2.8	Yes	1
10	110	Yes	0.8	No	0
11	168	No	1.9	Yes	1
12	139	Yes	1.1	No	0
13	189	Yes	4.6	Yes	1
14	166	Yes	4.1	No	1
15	100	No	3.7	Yes	1
16	118	Yes	3.2	No	1
17	107	No	4.8	No	1
18	103	No	2.0	Yes	0

Now we perform accuracy calculation with "dataset2" for which we perform "feature detection" task.

Performing accuracy tasks using decision tree classifier,SVM classifier,KNN classifier

decision tree classifier

In [3]:

```
import pandas as pd
dataset = pd.read_csv(r"C:\Project-Dataset 2.csv")

dataset.isnull().any()

m={'Yes': 1, 'No': 0}
dataset["Alcoholic/Smoker"]=dataset["Alcoholic/Smoker"].map(m)

n={'Yes': 1, 'No': 0}
dataset["Alzheimer"]=dataset["Alzheimer"].map(n)

x=dataset.iloc[:, 0:4].values
y=dataset.iloc[:,4].values

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)

from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier(criterion="entropy",random_state=0)
c.fit(x_train,y_train)
y_pred=c.predict(x_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm=confusion_matrix(y_test,y_pred)
print("Confusion matrix")
print(cm)
print(" ")
print("Decision Tree Classifier  ")
print("Accuracy = {0} %".format(accuracy_score(y_test,y_pred)*100))
```

```
Confusion matrix
[[2 1]
 [0 2]]

Decision Tree Classifier
Accuracy = 80.0 %
```

In [79]:

```
# Comparision of Real and Predicted Class values
Comparision_table1 = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
Comparision_table1
```

Out[79]:

	Real_class	Predicted_class
0	0	0
1	0	0
2	1	1
3	0	1
4	1	1

We can observe that from comparision table1 4 of the rows are matched among 5.Hence,we got an accuracy of 80%. By performing the task of accuracy there is increase in accuracy from 60%-80%

KNN classifier

In [5]:

```
import pandas as pd
dataset = pd.read_csv(r"C:\Project-Dataset 2.csv")

dataset.isnull().any()

m={'Yes': 1, 'No': 0}
dataset["Alcoholic/Smoker"]=dataset["Alcoholic/Smoker"].map(m)

n={'Yes': 1, 'No': 0}
dataset["Alzheimer"]=dataset["Alzheimer"].map(n)

x=dataset.iloc[:, 0:4].values
y=dataset.iloc[:,4].values

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)

from sklearn.neighbors import KNeighborsClassifier
c=KNeighborsClassifier(n_neighbors=3)
c.fit(x_train,y_train)
y_pred=c.predict(x_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm=confusion_matrix(y_test,y_pred)
print("Confusion matrix")
print(cm)
print(" ")
print("KNN Classifier  ")
print("Accuracy = {0} %".format(accuracy_score(y_test,y_pred)*100))
```

```
Confusion matrix
[[2 1]
 [0 2]]

KNN Classifier
Accuracy = 80.0 %
```

In [81]:

```
# Comparision of Real and Predicted Class values
Comparision_table2 = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
Comparision_table2
```

Out[81]:

	Real_class	Predicted_class
0	0	0
1	0	0
2	1	1
3	0	1
4	1	1

We can observe that from comparision table1 4 of the rows are matched among 5.Hence,we got an accuracy of 80%. By performing the task of accuracy there is increase in accuracy from 20%-80%

SVM classifier

In [8]:

```
import pandas as pd
dataset = pd.read_csv(r"C:\Project-Dataset 2.csv")

dataset.isnull().any()

m={'Yes': 1, 'No': 0}
dataset["Alcoholic/Smoker"]=dataset["Alcoholic/Smoker"].map(m)

n={'Yes': 1, 'No': 0}
dataset["Alzheimer"]=dataset["Alzheimer"].map(n)

x=dataset.iloc[:, 0:4].values
y=dataset.iloc[:,4].values

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)

from sklearn.svm import SVC
c=SVC(kernel="linear",random_state=0)
c.fit(x_train,y_train)
y_pred=c.predict(x_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm=confusion_matrix(y_test,y_pred)
print("Confusion matrix")
print(cm)
print(" ")
print("SVM Classifier  ")
print("Accuracy = {0} %".format(accuracy_score(y_test,y_pred)*100))
```

```
Confusion matrix
[[3 0]
 [0 2]]

SVM Classifier
Accuracy = 100.0 %
```

In [83]:

```
# Comparision of Real and Predicted Class values
Comparision_table3 = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
Comparision_table3
```

Out[83]:

	Real_class	Predicted_class
0	0	0
1	0	0
2	1	1
3	0	0
4	1	1

In [ ]:

```
We can observe that from comparision table1 4 of the rows are matched among 5.Hence,we got
By performing the task of accuracy there is increase in accuracy from 40%-80%
```

"By performing feature detection task we can conclude that the accuracy of the dataset is increases"