CURNEU MEDTECH INNOVATIONS PRIVATE LIMITED

SD03Q18

-GOBIKRISHNAN S.K(1832020)

PROBLEM STATEMENT-1

HEART FAILURE DEATH_EVENT PREDICTION USING KNN

ABSTRACT:

This document deals with the prediction to assess the death by heart failure using classification model. The model here is implemented by using KNN classifier algorithm .For the given dataset, we plot different scatter plots for different parameter combinations ,do feature scaling, split data into train and test sets, define a KNN classifier. The tool used here is Python.

INTRODUCTION:

Likelihood of Heart Failure death event can be predicted based on whether patient has anaemia, diabetes, high blood pressure, smoking habit etc….. .For a machine to assess the death event, we must provide basic information about the patients like the prior mentioned attributes for which we can develop a model and implement and see as to how accurate the machine is.

ABOUT DATASET:

age: age of the patient (years)

anaemia: decrease of red blood cells or hemoglobin (boolean)

high blood pressure: if the patient has hypertension (boolean)

creatinine phosphokinase : level of the CPK enzyme in the blood

diabetes: if the patient has diabetes (boolean)

ejection fraction: percentage of blood leaving the heart at each contraction (percentage)

platelets: platelets in the blood

sex: woman or man

serum creatinine: level of serum creatinine in the blood

serum sodium: level of serum sodium in the blood

smoking: if the patient smokes or not

time: follow-up period

[target] death event: if the patient deceased during the follow-up period (boolean)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	anaemia	creatinine_	diabetes	ejection_fra	high_blood_	platelets	serum_crea	serum_sodi	sex	smoking	time	DEATH_EVENT	
2	75	0	582	0	20	1	265000	1.9	130	1	0	4	1	
3	55	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1	
4	65	0	146	0	20	0	162000	1.3	129	1	1	7	1	
5	50	1	111	0	20	0	210000	1.9	137	1	0	7	1	
6	65	1	160	1	20	0	327000	2.7	116	0	0	8	1	
7	90	1	47	0	40	1	204000	2.1	132	1	1	8	1	
8	75	1	246	0	15	0	127000	1.2	137	1	0	10	1	
9	60	1	315	1	60	0	454000	1.1	131	1	1	10	1	
10	65	0	157	0	65	0	263358.03	1.5	138	0	0	10	1	
11	80	1	123	0	35	1	388000	9.4	133	1	1	10	1	
12	75	1	81	0	38	1	368000	4	131	1	1	10	1	
13	62	0	231	0	25	1	253000	0.9	140	1	1	10	1	
14	45	1	981	0	30	0	136000	1.1	137	1	0	11	1	
15	50	1	168	0	38	1	276000	1.1	137	1	0	11	1	
16	49	1	80	0	30	1	427000	1	138	0	0	12	0	
17	82	1	379	0	50	0	47000	1.3	136	1	0	13	1	
18	87	1	149	0	38	0	262000	0.9	140	1	0	14	1	
19	45	0	582	0	14	0	166000	0.8	127	1	0	14	1	
20	70	1	125	0	25	1	237000	1	140	0	0	15	1	
21	48	1	582	1	55	0	87000	1.9	121	0	0	15	1	
22	65	1	52	0	25	1	276000	1.3	137	0	0	16	0	
23	65	1	128	1	30	1	297000	1.6	136	0	0	20	1	
24	68	1	220	0	35	1	289000	0.9	140	1	1	20	1	
25	53	0	63	1	60	0	368000	0.8	135	1	0	22	0	
26	75	0	582	1	30	1	263358.03	1.83	134	0	0	23	1	
27	80	0	148	1	38	0	149000	1.9	144	1	1	23	1	
28	95	1	112	0	40	1	196000	1	138	0	0	24	1	
29	70	0	122	1	45	1	284000	1.3	136	1	1	26	1	
30	58	1	60	0	38	0	153000	5.8	134	1	0	26	1	
31	82	0	70	1	30	0	200000	1.2	132	1	1	26	1	
32	94	0	582	1	38	1	263358.03	1.83	134	1	0	27	1	

CODE EXPLANATION:

Importing Required Libraries:

- First, we import the required libraries for the code to run effectively

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
from sklearn.model_selection import train_test_split
import seaborn as sns
```

Importing heart failure clinical records dataset:

- Given heart failure clinical records is imported

```
df=pd.read_csv(r'C:\Users\GOBIKRISHNAN\heart_failure_clinical_records_dataset.csv')

X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Training and testing datas

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=42, stratify=y)
```

Droping Null values

```
df.dropna()
```

Plots comparing different attributes

- To visually see the how each parameter effects the other parameters, we can use scatter plots, line, histogram, boxplot

```
sns.boxplot(x = df['age'])
plt.title("age BOX_PLOT")
plt.show()
```

#Hist for age

```
df['age'].hist()
plt.title('age')
plt.show()
```

#Hist for Death_event

```
df['DEATH_EVENT'].hist()
plt.title('DEATH_EVENT ')
plt.show()
```

#COUNTPLOT

```
df['DEATH_EVENT'].value_counts().plot.pie(autopct = '%1.1f%%', shadow= True, figsize = (6, 6))
plt.title("DEATH_EVENT distrubion")
plt.show()
```

#COUNTPLOT for diabetes

```
sns.countplot('diabetes',data = df )
plt.title("diabetes")
plt.show()
```

#COUNTPLOT for anaemia

```
sns.countplot('anaemia',data = df )
plt.title("anaemia")
plt.show()
```

#COUNTPLOT for anaemia

```
sns.countplot('high_blood_pressure' ,data = df )
plt.title("high_blood_pressure")
plt.show()
```

#COUNTPLOT for smoking

```
sns.countplot('smoking' ,data = df )
plt.title("smoking")
plt.show()
```

#Scatterplot age vs creatinine_phosphokinase

```
sns.scatterplot(data = df,x = 'age',y = 'creatinine_phosphokinase',hue = 'anaemia')
plt.title("age vs creatinine_phosphokinase ")
plt.show()
```

#Scatterplot age vs ejection_fraction

```
sns.scatterplot(data = df,x = 'age',y = 'ejection_fraction',hue = 'diabetes')
plt.title("age vs ejection_fraction ")
plt.show()
```

#Scatterplot age vs serum_sodium

```
sns.scatterplot(data = df,x = 'age',y = 'serum_sodium',hue = 'serum_creatinine')
plt.title("age vs serum_sodium")
plt.show()
```

#CORRELATION

```
sns.heatmap(df.corr(), annot=True)
plt.title("correlation heatmap")
plt.show()
```

#FEATURE SCALING

- Feature scaling is done to normalize the data range or feature

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

#EUCLIDEAN DISTANCE

```
def euclid_dist(v1,v2):
dist = np.sqrt(np.sum((v1-v2)**2))
return dist
```

#KNN PREDICTION

```
def knn_predict(X_train, X_test, y_train, y_test, k):

# Counter to help with label voting
from collections import Counter
```

```
    # Make predictions on the test data
    # Need output of 1 prediction per test data point
    y_hat_test = []

    for test_point in X_test:
        distances = []

        for train_point in X_train:
            distance = euclid_dist(test_point, train_point)
            distances.append(distance)

        # Storing distances in a dataframe
        df_dists = pd.DataFrame(data=distances, columns=['dist'],
                                index=y_train)

        # Sort distances and considering the k closest points
        df_nn = df_dists.sort_values(by=['dist'], axis=0)[:k]

        # Create counter object to track the labels of k closest neighbors
        counter = Counter(y_train[df_nn.index])

        # Get most common label of all the nearest neighbors
        prediction = counter.most_common()[0][0]

        # Append prediction to output list
        y_hat_test.append(prediction)

    return y_hat_test
y_hat_test=knn_predict(X_train, X_test, y_train, y_test, 3)

#ACCURRACY SCORE
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(y_test,y_hat_test))
```

#KNN CLASSIFIER FUNCTION

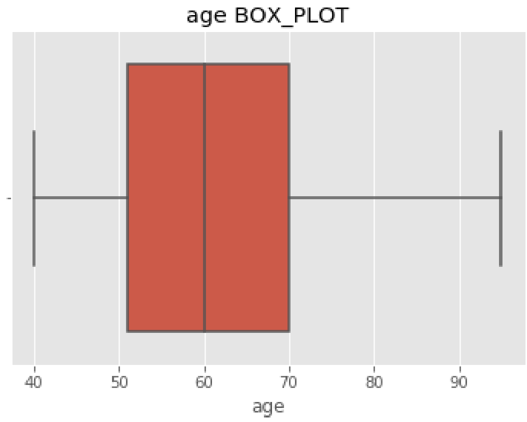
It is a simple classification algorithm. its purpose is to use a database in which the data points are separated into several classes

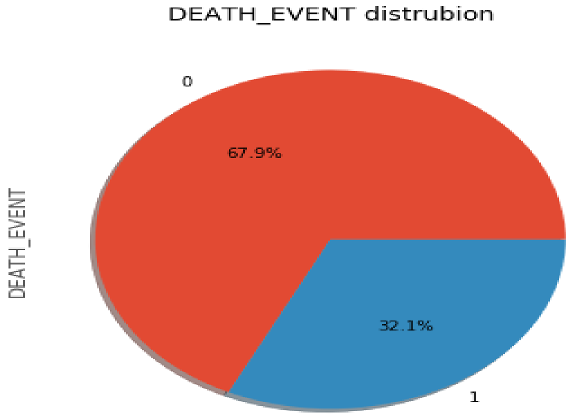
```
from sklearn.neighbors import KNeighborsClassifier
neighbors = np.arange(1,9)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
for i,k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(X_train, y_train)
train_accuracy[i] = knn.score(X_train, y_train)
test_accuracy[i] = knn.score(X_test, y_test)
plt.title('KNN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
```

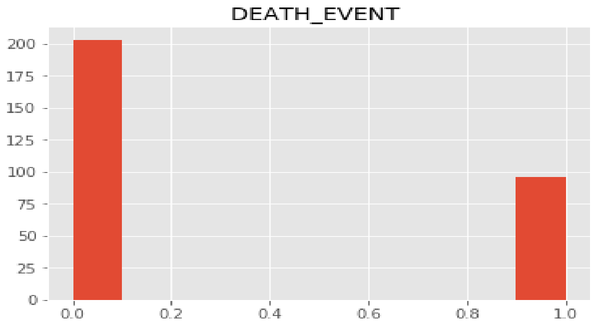
#ACCURRACY AND CONFUSION MATRIX

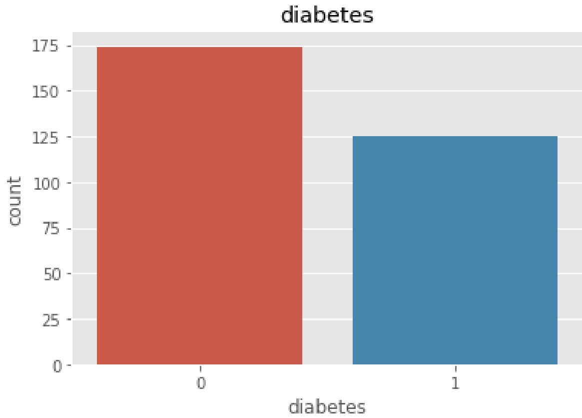
```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,y_train)
print("Accuracy", knn.score(X_test,y_test)*100)
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
```

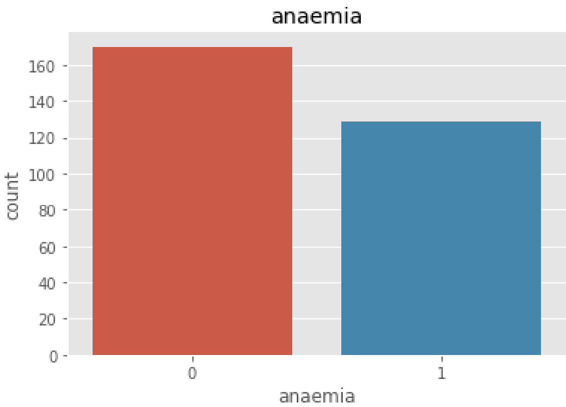
OUTPUT:

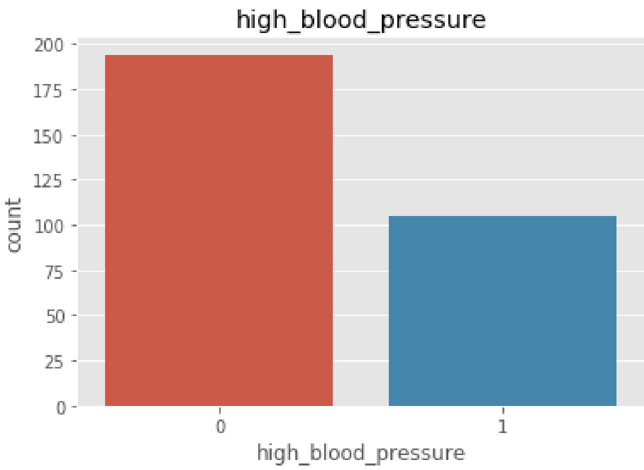

age BOX_PLOT
40
50
60
70
80
90
age

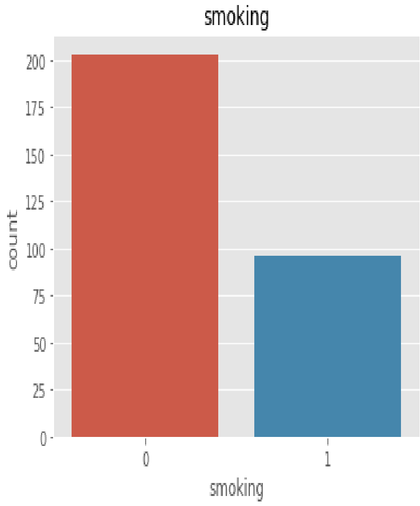

DEATH_EVENT distrubion
DEATH_EVENT
0
67.9%
32.1%
1

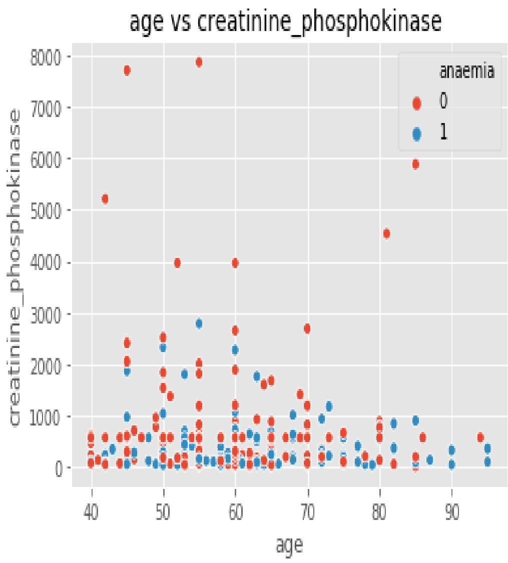

DEATH_EVENT
200
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0

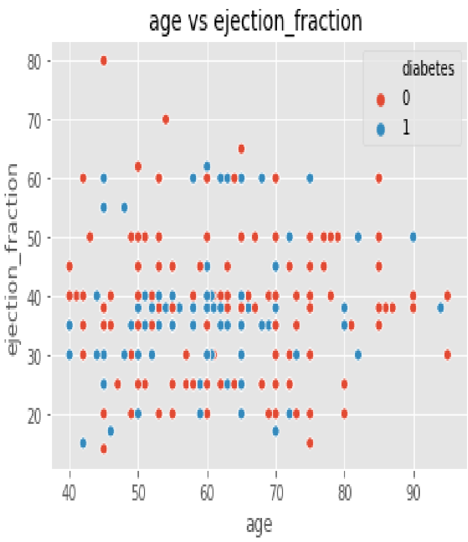

diabetes
175
150
125
100
75
50
25
0
count
0
1
diabetes

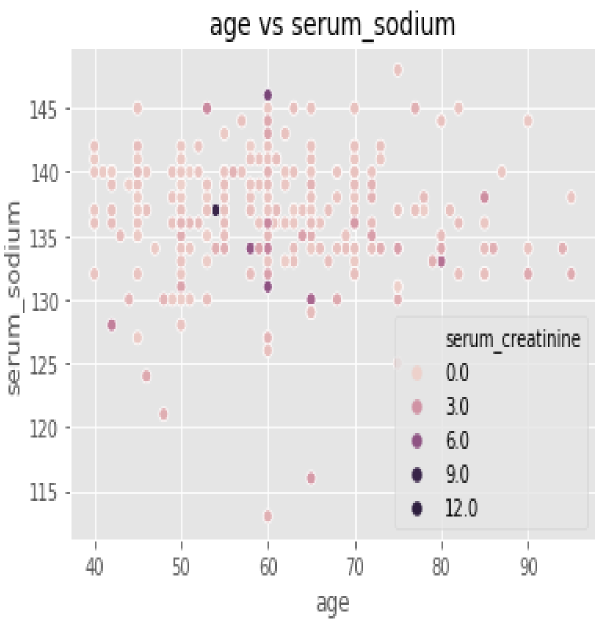

anaemia
160
140
120
100
80
60
40
20
0
count
0
1
anaemia

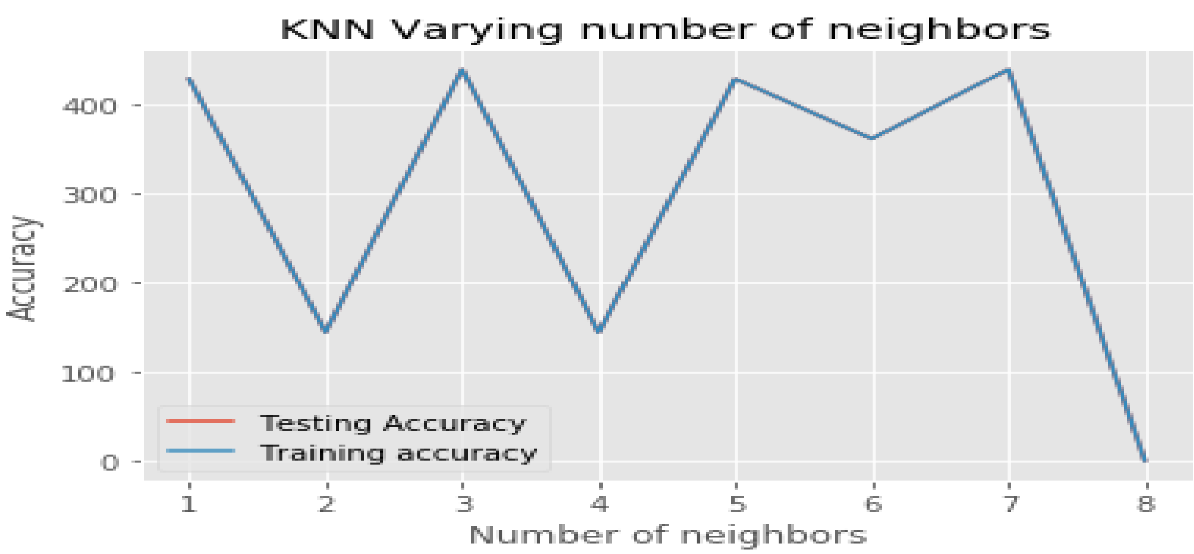

high_blood_pressure
200
175
150
125
100
75
50
25
0
count
0
1
high_blood_pressure


smoking
count
0 25 50 75 100 125 150 175 200
0
1
smoking


age vs creatinine_phosphokinase
creatinine_phosphokinase
0 1000 2000 3000 4000 5000 6000 7000 8000
40 50 60 70 80 90
age
anaemia
0
1


age vs ejection_fraction
ejection_fraction
20 30 40 50 60 70 80
40 50 60 70 80 90
age
diabetes
0
1


age vs serum_sodium
serum_sodium
115 120 125 130 135 140 145
40 50 60 70 80 90
age
serum_creatinine
0.0
3.0
6.0
9.0
12.0


KNN Varying number of neighbors
Accuracy
0 100 200 300 400
1 2 3 4 5 6 7 8
Number of neighbors
Testing Accuracy
Training accuracy

```
Accuracy 76.0
```

Finally we have achieved the given output plot for KNN and the with the accuracy of 76%

We have used a built in code to check if the scratch code is working properly

CONCLUSION:

The model can predict the death event in the given solution but with the margin of 20%.Thus this model help us to predict future cardiovascular deaths beforehand so that we can take appropriate measures to reduce the death rate caused by this since it is the major contributor of death world wide