CURNEW MEDTECH INNOVATIONS PRIVATE LIMITED

SD03Q01

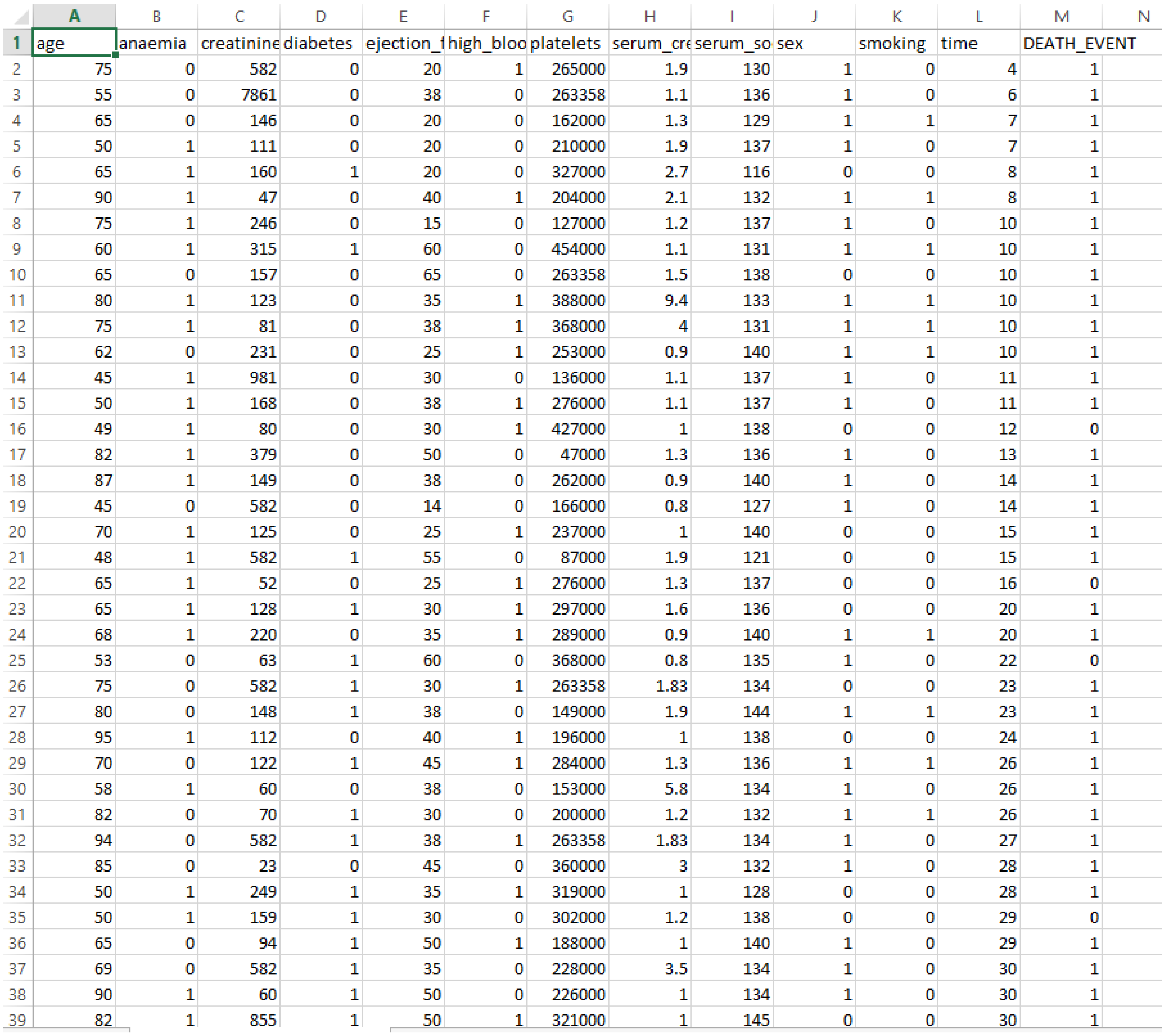
-Sanjay P S (1832044)

Abstract:

The given dataset contains data regarding the factors affecting the Cardiovascular diseases, for which a model is built to assess the likelihood of a death by heart failure event. The tool used is python and the model is built from scratch. The model implemented is KNN classifier algorithm. For the given dataset, we first explore the data and analyze each and every variable, then split data into train and test sets, define a KNN classifier from scratch and achieve the final output.

Dataset:

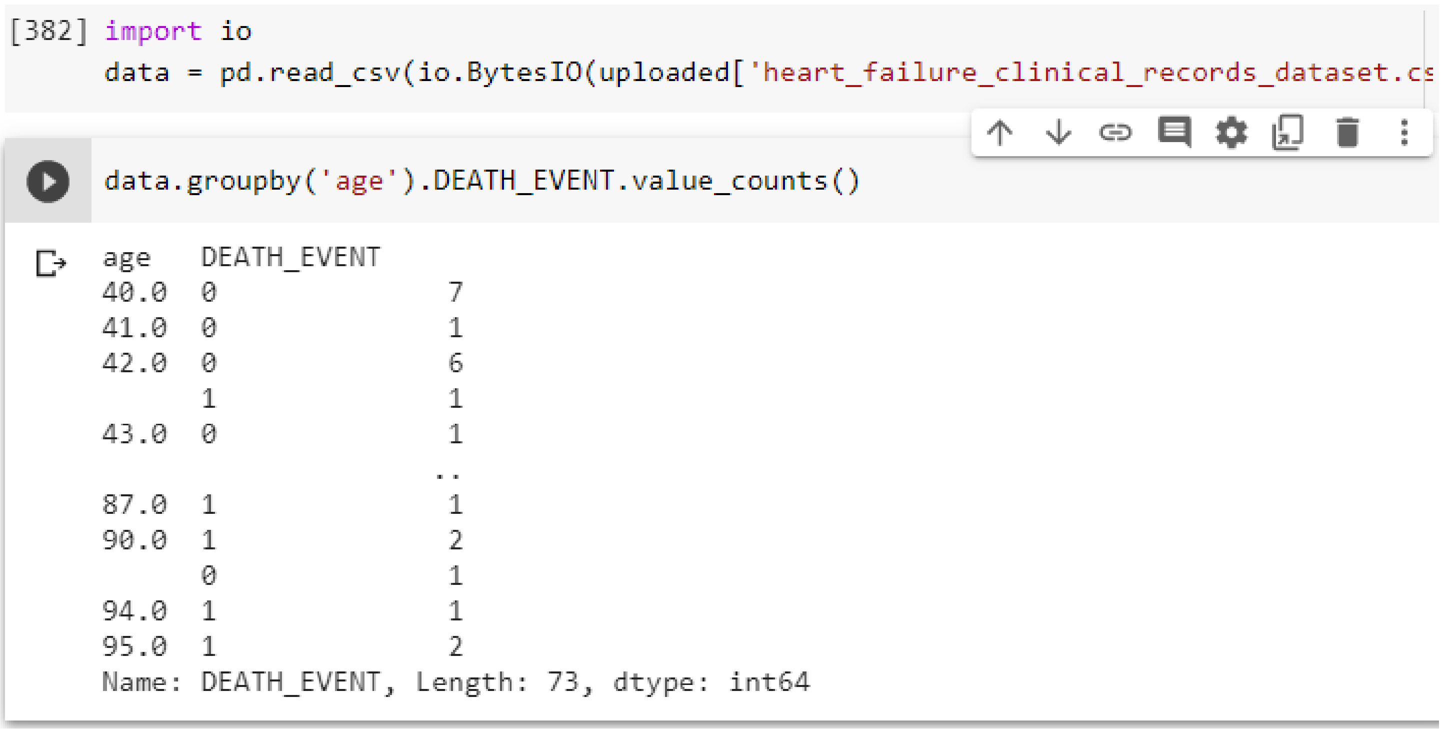
The dataset contains 13 features and 300 rows.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	anaemia	creatinine	diabetes	ejection_f	high_bloo	platelets	serum_cre	serum_so	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	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	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	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	1.83	134	1	0	27	1	
33	85	0	23	0	45	0	360000	3	132	1	0	28	1	
34	50	1	249	1	35	1	319000	1	128	0	0	28	1	
35	50	1	159	1	30	0	302000	1.2	138	0	0	29	0	
36	65	0	94	1	50	1	188000	1	140	1	0	29	1	
37	69	0	582	1	35	0	228000	3.5	134	1	0	30	1	
38	90	1	60	1	50	0	226000	1	134	1	0	30	1	
39	82	1	855	1	50	1	321000	1	145	0	0	30	1	

Exploratory Data Analysis:

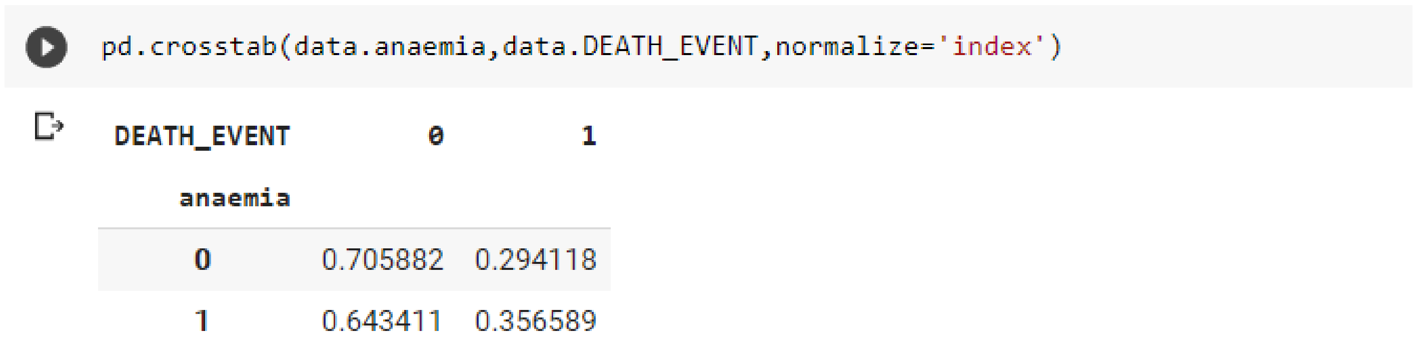
First we check all the features in relation to the death event which is the Y variable of this problem and eliminate the negatively affecting variables. We use the function crosstab to check the variation of Y with respect to all the features and we remove the randomized variables as it will not add to the accuracy of the model. All the variables were compared individually and in pairs to the Y variable to check it's influence on the result. After the EDA the data is cleaned to eliminate null values. Finally the undesired features of the dataset are also dropped to increase the accuracy of the model.



```
[382] import io
      data = pd.read_csv(io.BytesIO(uploaded['heart_failure_clinical_records_dataset.cs
```

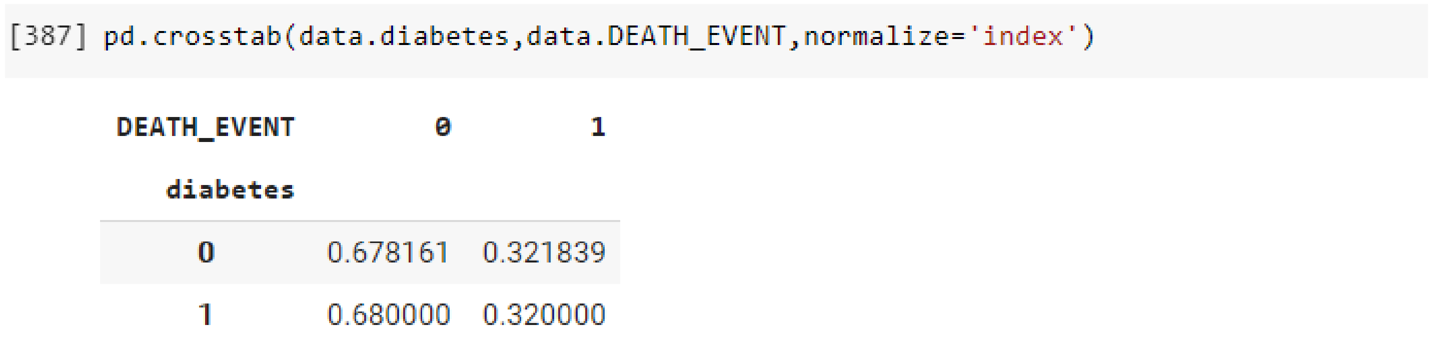
```
data.groupby('age').DEATH_EVENT.value_counts()
```

```
age   DEATH_EVENT
40.0  0              7
41.0  0              1
42.0  0              6
      1              1
43.0  0              1
                    ..
87.0  1              1
90.0  1              2
      0              1
94.0  1              1
95.0  1              2
Name: DEATH_EVENT, Length: 73, dtype: int64
```



```
pd.crosstab(data.anaemia,data.DEATH_EVENT,normalize='index')
```

DEATH_EVENT	0	1
anaemia		
0	0.705882	0.294118
1	0.643411	0.356589



```
[387] pd.crosstab(data.diabetes,data.DEATH_EVENT,normalize='index')
```

DEATH_EVENT	0	1
diabetes		
0	0.678161	0.321839
1	0.680000	0.320000

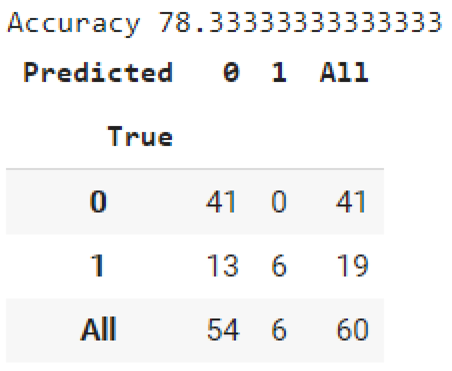
Implementation of KNN:

In statistics, the k-nearest neighbors algorithm (k-NN) is a non-parametric classification method. n k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. First the data is split into training and testing set.

Code:

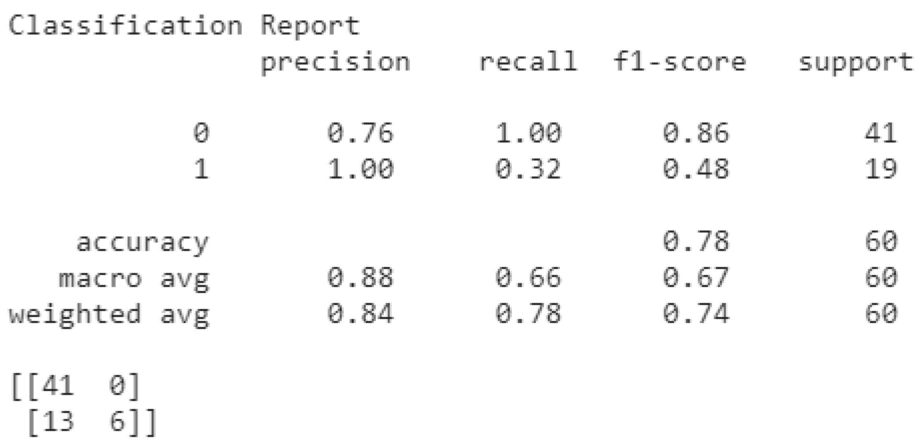
```
import io
data = pd.read_csv(io.BytesIO(uploaded['heart_failure_clinical_records_dataset.csv'])
)
data=data.drop(columns=['serum_creatinine','creatinine_phosphokinase','serum_sodium',
])
data=data.dropna()
X=data.iloc[:,:9]
y=data.iloc[:,9]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=42,s
tratify=y)
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
def euclid_dist(v1,v2):
    dist = np.sqrt(np.sum((v1-v2)**2))
    return dist
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)

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:



```
Accuracy 78.33333333333333
```

Predicted	0	1	All
True			
0	41	0	41
1	13	6	19
All	54	6	60



```
Classification Report
               precision    recall  f1-score   support

           0       0.76      1.00      0.86        41
           1       1.00      0.32      0.48        19

    accuracy                           0.78        60
   macro avg       0.88      0.66      0.67        60
weighted avg       0.84      0.78      0.74        60

[[41  0]
 [13  6]]
```

Finally we have achieved the given output plot for the random forest model with and accuracy of 78.4%. We have used a in built code to check if the starch code is working properly with the help of the classification report and confusion matrix which gives out the same result.

Conclusion:

The model can predict the death event in the given situation but with a margin of 20%. Thus this model helps 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 to all death worldwide.