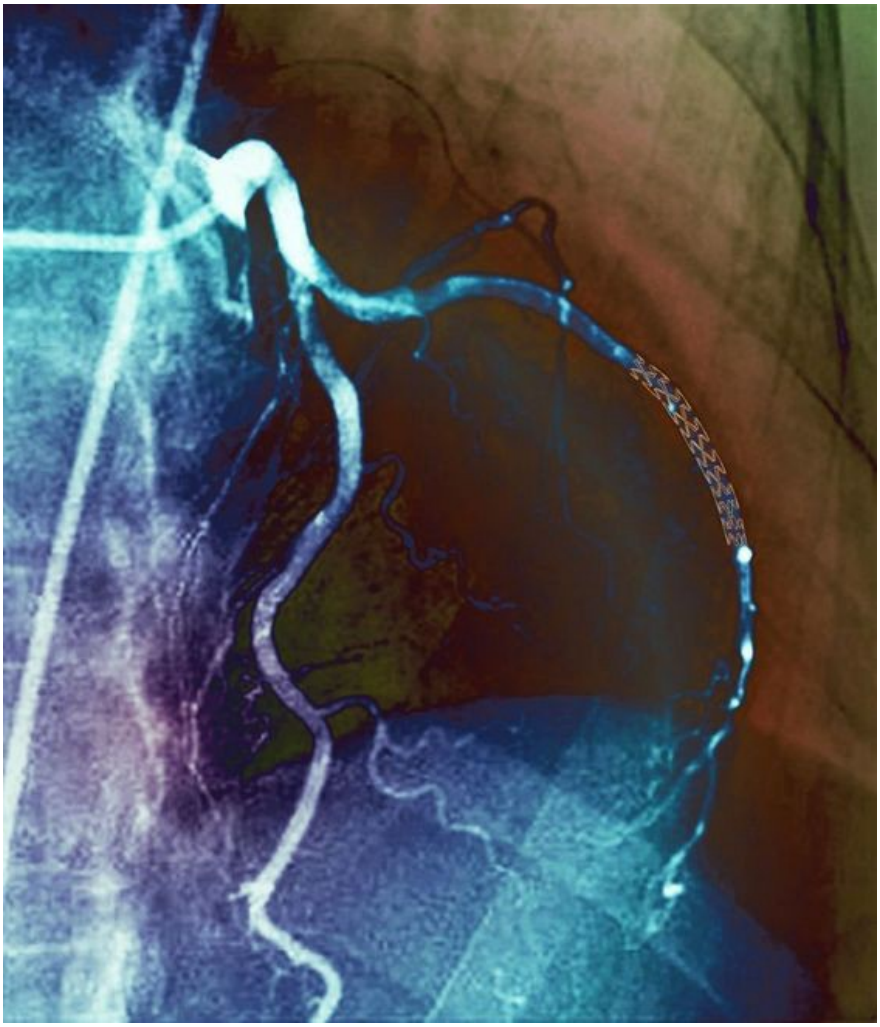
Sahil Verma Follow
Python Developer | Machine Learning enthusiast | Freelancer |Amateur Chef | Reader | Runner |
ping me at: sahilverma0696@gmail.com
Oct 22 · 4 min read

heart disease prediction



The project is about predicting coronary heart disease by using three different ML algorithms.

Support Vector Machine

K Nearset Neighbour

ANN Multilayer Perceptron

And to know which is the best approach.

DATA DESCRIPTION

South Africa Heart Disease Dataset Source :
https://www.openml.org/d/1498

A retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa.

There are roughly two controls per case of CHD. Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their occurrence of CHD. In some cases the measurements were made after these treatments. These data are taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal.

Attributes

1. Systolic blood pressure (Sbp)
2. Cumulative tobacco consumption (kg)
3. Low density lipoprotein (LDL-cholesterol)
4. Adiposity
5. Family history of heart disease (Present/Absent)
6. Type-A behavior
7. Obesity
8. Current alcohol consumption
9. Age during onset of condition
10. CHD response

Total value count : 462

In [1]:
```
# Importing primary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook
```

DATA PREPROCESSING

In [2]:
```
# Reading the data
data = pd.read_csv('https://www.openml.org/data/get_csv/1592290/phpgNaXZe')
```

In [3]:
```
# Setting up the column
column = ['sbp','tobacco','ldl','adiposity','famhist','type','obesity','alcohol','age','chd']
```

In [4]:
```
data.columns=column
```

In [5]:
```
data.head()
```

Out[5]:

	sbp	tobacco	ldl	adiposity	famhist	type	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	1	49	25.30	97.20	52	2
1	144	0.01	4.41	28.61	2	55	28.87	2.06	63	2
2	118	0.08	3.48	32.28	1	52	29.14	3.81	46	1
3	170	7.50	6.41	38.03	1	51	31.99	24.26	58	2
4	134	13.60	3.50	27.78	1	60	25.99	57.34	49	2

The table determines the raw data.

Now, we don't have any missing values in our data and to equalize the values of various attributes, we Apply Feature Scaling and Min-Max Scaling.

And the data now appears to be as follows:

In [8]:
```
# Feature Scaling, making categorical data precise
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data['famhist']=encoder.fit_transform(data['famhist'])
data['chd']=encoder.fit_transform(data['chd'])
```

In [9]:
```
data.head(5)
```

Out[9]:

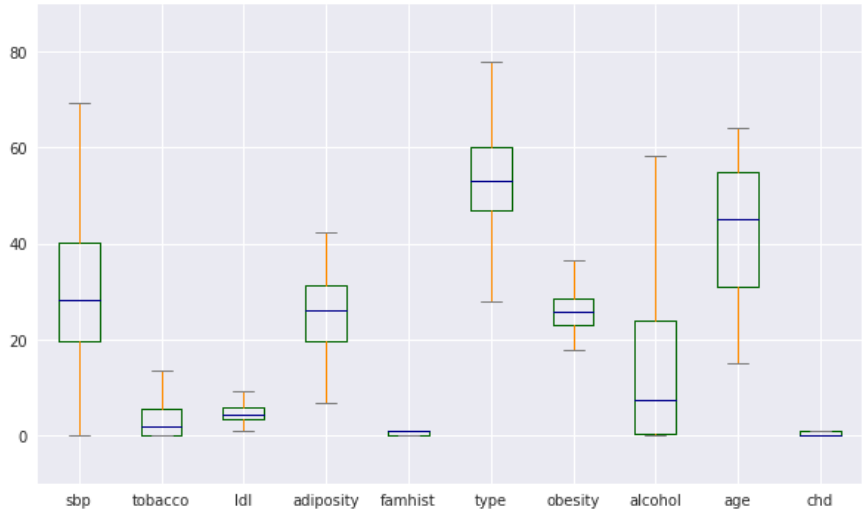
	sbp	tobacco	ldl	adiposity	famhist	type	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	0	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	1	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	0	52	29.14	3.81	46	0
3	170	7.50	6.41	38.03	0	51	31.99	24.26	58	1
4	134	13.60	3.50	27.78	0	60	25.99	57.34	49	1

And the data description after preprocessing is as follows:

In [13]:
```
# Data after modification
data.describe()
```

Out[13]:

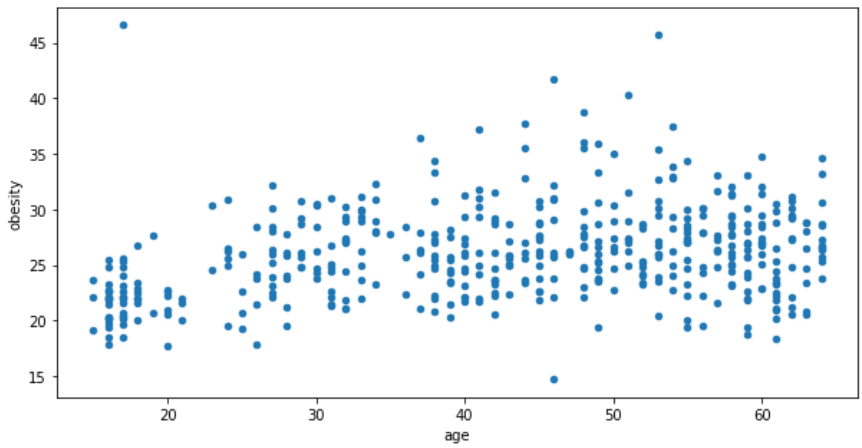
	sbp	tobacco	ldl	adiposity	famhist	type	obesity	alcohol	age	chd
count	462.000000	462.000000	462.000000	462.000000	462.000000	462.000000	462.000000	462.000000	462.000000	462.0
mean	31.903282	3.635649	4.740325	25.406732	0.584416	53.103896	26.044113	17.044394	42.816017	0.346
std	17.518220	4.593024	2.070909	7.780699	0.493357	9.817534	4.213680	24.481059	14.608956	0.476
min	0.000000	0.000000	0.980000	6.740000	0.000000	13.000000	14.700000	0.000000	15.000000	0.000
25%	19.658120	0.052500	3.282500	19.775000	0.000000	47.000000	22.985000	0.510000	31.000000	0.000
50%	28.205128	2.000000	4.340000	26.115000	1.000000	53.000000	25.805000	7.510000	45.000000	0.000
75%	40.170940	5.500000	5.790000	31.227500	1.000000	60.000000	28.497500	23.892500	55.000000	1.000
max	100.000000	31.200000	15.330000	42.490000	1.000000	78.000000	46.580000	147.190000	64.000000	1.000



Box plot of different features

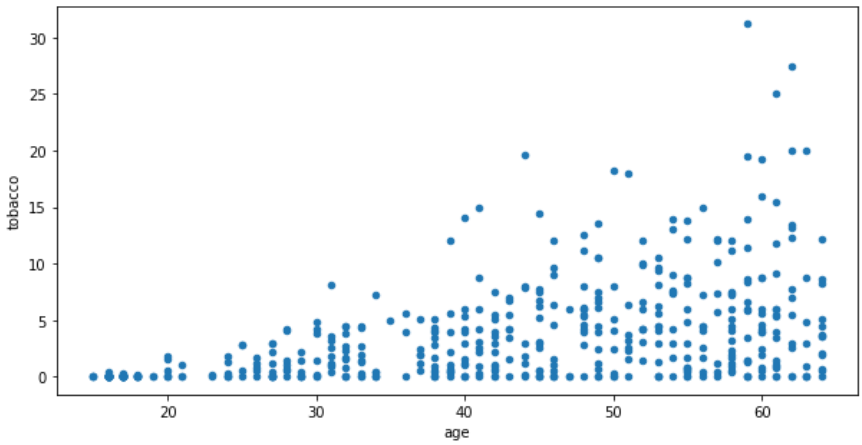
DATA VISUALIZATION

After plotting various attributes against each other, we get some useful results as follows.

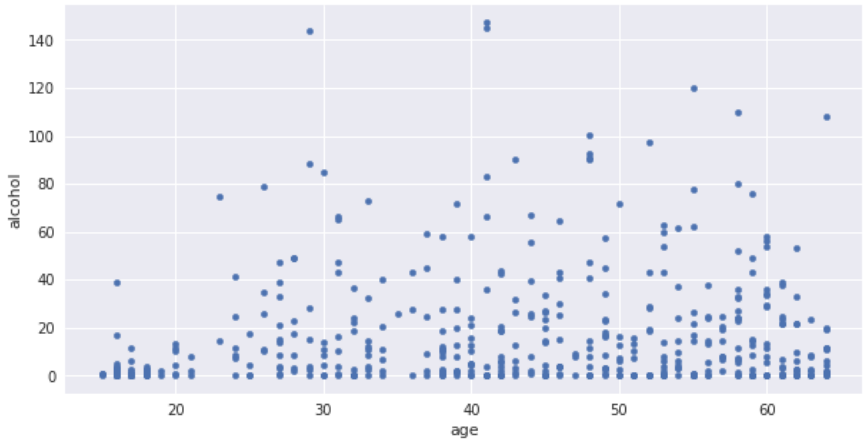


The above scatter graph (obesity vs age) gives us information; we can observe the clusters mainly in the following age groups:

1. Before the age of 20.
2. After the age of 40 and mostly in 50s.



In the tobacco consumption vs. age graph, we can see that the consumption increases after the age of 30.



According to alcohol consumption vs. age graph, it is observed that maximum alcohol consumption begins right after 25 years of age.

PREDICTION

SUPPORT VECTOR MACHINE

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.
Description : https://en.wikipedia.org/wiki/Support_vector_machine

```
In [28]: from sklearn import svm
         svm_clf = svm.SVC(kernel ='linear')

In [29]: svm_clf.fit(X_train,y_train)

Out[29]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)

In [30]: y_pred_svm =svm_clf.predict(X_test)

In [31]: y_pred_svm

Out[31]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0,
                0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
                0, 0, 0, 1, 0])

In [38]: # Making the Confusion Matrix
         from sklearn.metrics import confusion_matrix
         cm_svm = confusion_matrix(y_test, y_pred_svm)
         cm_svm

Out[38]: array([[51,  9],
                [15, 18]])

In [34]: from sklearn.metrics import accuracy_score
         svm_result = accuracy_score(y_test,y_pred_svm)
         print("Accuracy :",svm_result)

         Accuracy : 0.7419354838709677
```

Accuracy obtained in SVM ~74%

k-NEAREST NEIGHBOR

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. … In k-NN classification, the output is a class membership. Description : https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

```
In [124]: from sklearn.neighbors import KNeighborsClassifier
          knn_clf = KNeighborsClassifier(n_neighbors =5,n_jobs = -1,leaf_size = 60,algorithm='brute')

In [125]: knn_clf.fit(X_train,y_train)

Out[125]: KNeighborsClassifier(algorithm='brute', leaf_size=60, metric='minkowski',
                    metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                    weights='uniform')

In [126]: y_pred_knn = knn_clf.predict(X_test)
          y_pred_knn

Out[126]: array([0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
                 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                 0, 0, 0, 1, 0])

In [127]: # Making the Confusion Matrix
          from sklearn.metrics import confusion_matrix
          cm_knn = confusion_matrix(y_test, y_pred_knn)
          cm_knn

Out[127]: array([[49, 11],
                 [22, 11]])

In [128]: knn_result = accuracy_score(y_test,y_pred_knn)
          knn_result

Out[128]: 0.6451612903225806
```

Accuracy obtained in KNN ~65%

ANN Multilayer Perceptron Classifier

Artificial neural networks or connectionist systems are computing systems vaguely inspired by the biological neural networks that

constitute animal brains.

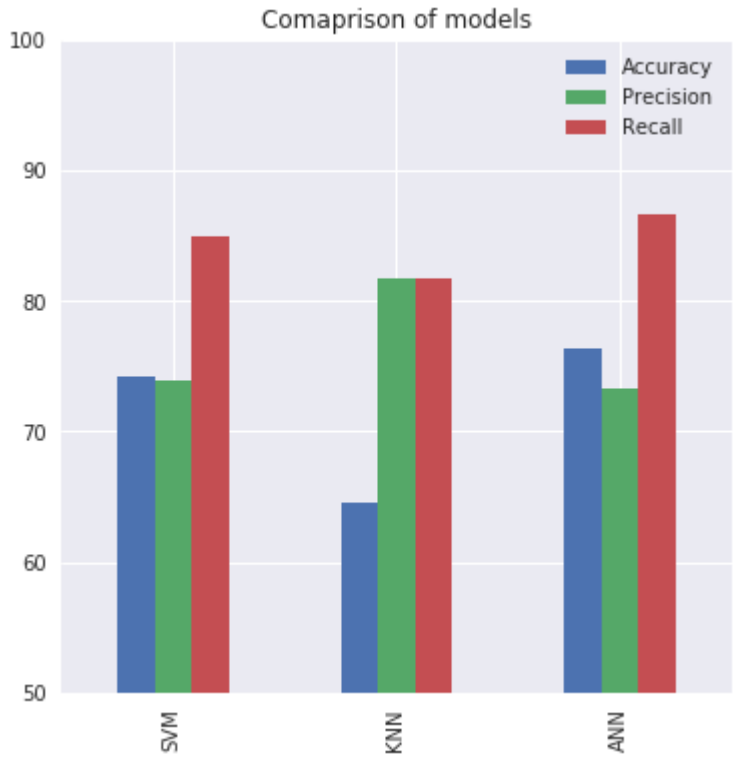
Description : https://en.wikipedia.org/wiki/Artificial_neural_network

```
In [137]: from sklearn.metrics import make_scorer, accuracy_score
          from sklearn.model_selection import GridSearchCV
          from sklearn.neural_network import MLPClassifier

In [138]: ann_clf = MLPClassifier()

          #Parameters
          parameters = {'solver': ['lbfgs'],
                        'alpha':[1e-4],
                        'hidden_layer_sizes':(9,14,14,2),   # 9 input, 14-14 neuron in 2 layers,1 output la
          yer
                        'random_state': [1]}

In [139]: # Type of scoring to compare parameter combos
          acc_scorer = make_scorer(accuracy_score)

          # Run grid search
          grid_obj = GridSearchCV(ann_clf, parameters, scoring=acc_scorer)
          grid_obj = grid_obj.fit(X_train, y_train)

          # Pick the best combination of parameters
          ann_clf = grid_obj.best_estimator_

In [140]: # Fit the best algorithm to the data
          ann_clf.fit(X_train, y_train)

Out[140]: MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                 beta_2=0.999, early_stopping=False, epsilon=1e-08,
                 hidden_layer_sizes=14, learning_rate='constant',
                 learning_rate_init=0.001, max_iter=200, momentum=0.9,
                 nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
                 solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
                 warm_start=False)

In [141]: y_pred_ann = ann_clf.predict(X_test)

In [142]: # Making the Confusion Matrix
          from sklearn.metrics import confusion_matrix
          cm_ann = confusion_matrix(y_test, y_pred_ann)
          cm_ann

Out[142]: array([[52,  8],
                 [14, 19]])

In [143]: ann_result = accuracy_score(y_test,y_pred_ann)
          ann_result

Out[143]: 0.7634408602150538
```

Accuracy obtained by MLP ~76%

COMPARISION OF THE THREE APPROACHES

After applying the three prediction algorithms, we obtain the following results:



1. Best accuracy obtained in ANN.
2. Best Precision in KNN.
3. Best Recall in ANN.

Therefore, using Artificial Neural Network is the best approach out of the three followed by k-nearest neighbours and Support Vector Machine (even on small datasets).

For the complete code, please visit my github repo:

sahilverma0696/heart-disease-prediction

Heart Disease Prediction using SVM, KNN and MLP and comparison of results - sahilverma0696/hea...

github.com

Remember to give this post some 👏 if you liked it. Follow me for more content :)