7072CEM MACHINE LEARNING

TOPIC: PERFORMING CLASSIFICATION TECHNIQUES ON ECHOCARDIOGRAM DATASET

TEAM MEMBER 1: Sahil Kumbhar(13140139)

TEAM MEMBER 2: Gaurav Upadhyay(12668836)

Introduction:

There has been a great modernisation in the medical field and with this rapid growth there are many more ways that machine learning can prove to be useful and effective in a long run. Radiology images or medical images interpretation by the doctors can delay the medical process due to their unavailability of doctors or radiologists. Formerly evaluated by cardiologists, emerging computer-based methods now enables deep neural networks (DNN) to help in the evaluation of obtained pictures. Perhaps this is particularly useful for sonographers with sporadic echocardiography practise and may broaden the use of echocardiogram to contexts previously regarded to be out in its capabilities, such as primary healthcare or medical treatment in distant places. In this paper we propose the echocardiography medical images interpretation using deep learning automation as unsupervised or semi-supervised learning have proven to be of higher accuracy for the large unlabelled datasets and smaller labelled datasets combined. In this paper we have used four methods to perform echocardiography on a dataset and have compared the results of the four methods and found the most optimal output. The algorithms used are Logistic regression, Decision tree, K – nearest neighbour algorithm and Random forest classifier. Throughout the process Random forest Classifier gave the most accurate output and was most efficient.

DATASET:

In this paper we have used dataset from UCI machine learning repository. This data set has 132 instances and has 12 attributes. This Data is used to determine whether patients would live for at least one year following a heart attack.

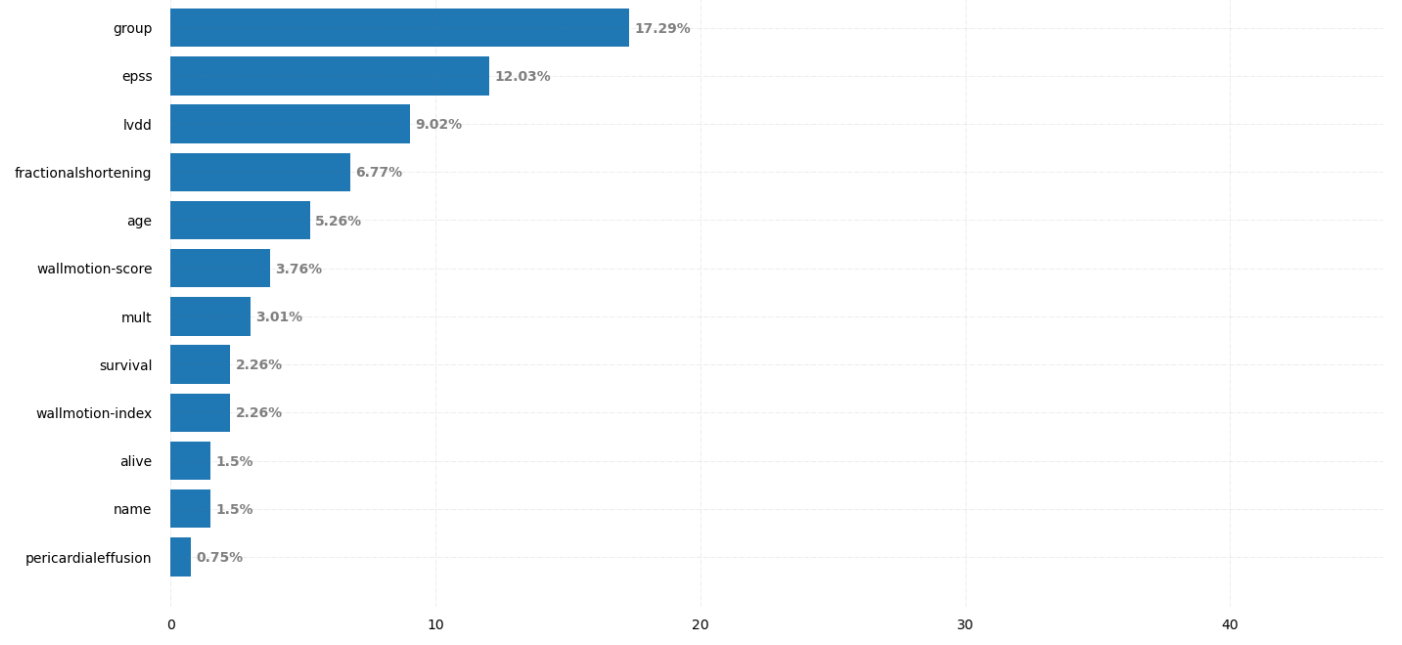
LINK: <https://archive.ics.uci.edu/ml/datasets/echocardiogram>

Experimental Setup:

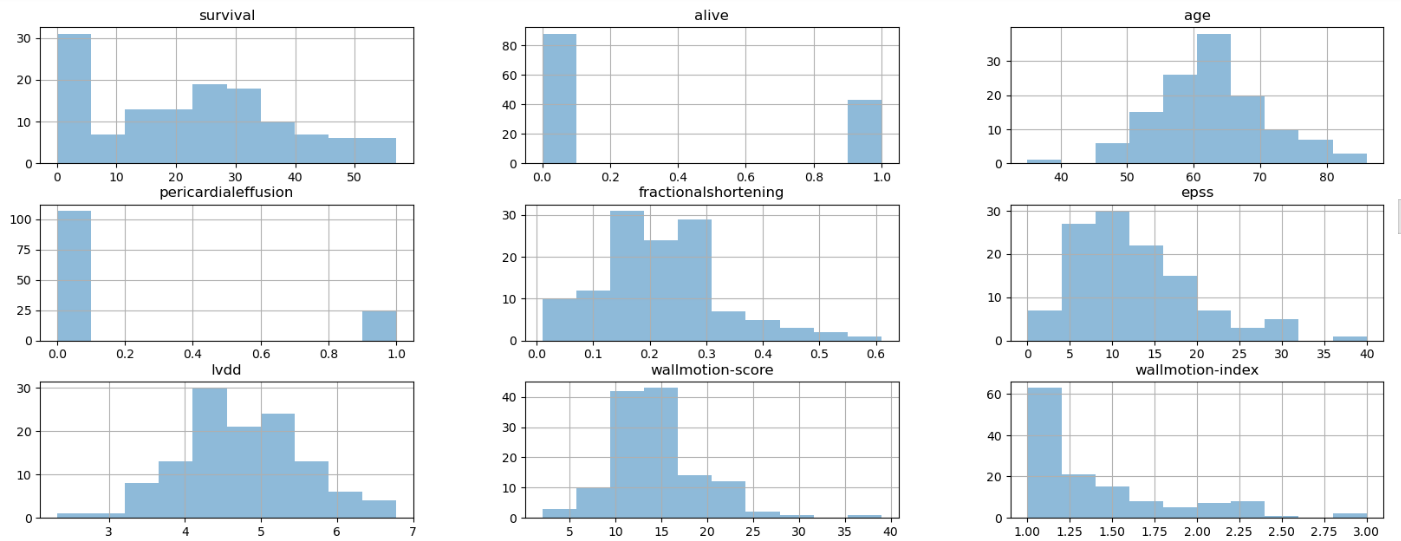
In order to perform the machine learning technique we have done multiple steps.

The steps are:

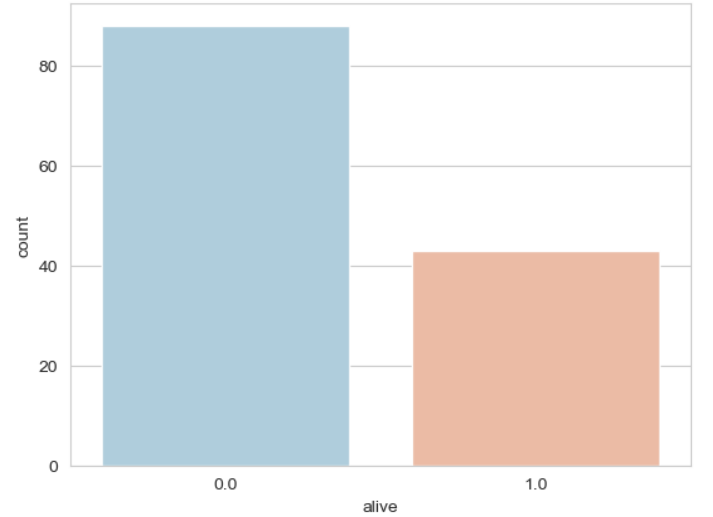
* Importing the dataset.
* Calculating the NULL Value percentage.
* Removing the NULL rows present in alive rows.



* Labelling the Data and removing the columns having many missing values in the data set.
* Feature selection.



* Database Balancing: we check whether the data base are balanced or not and can be used for our implementation.



* Feature Scaling.
* Identifying outliers and removing the outlier
* Splitting the train test data
* Training the model and testing it.

Methodology:

We have performed four various classification on the echocardiography and trying to find the optimal classification algorithm for the use. The four classifications are:

* Logistic Regression :

The method of modelling the likelihood of a discrete result based on a given variable is known as logistic regression. The most frequent regression analysis models include a binary result. Multiple linear regression can be used to describe events with more than two discrete outcomes. Logistic regression is a helpful analytical tool for classification issues, such as determining if a recent sample belongs in a specific group. logistic regression is now a valuable analytic tool.

Advantages:

Easier to implement and efficient to train.

Multinomial regression.

Feature coefficients can be seen as indications of variable importance.

Disadvantages:

Linear boundaries are created.

The hypothesis that the dependent variable as well as the independent variables are linear are the major limitation.

Complex associations are difficult to establish using logistic regression.

Output for Logistic Regression:



* Decision tree Algorithm.

Decision Tree is a Guided learning approach that can be used both for classification and regression issues, however it is most widely utilized for classification. It is a tree-structured classifier in which internal nodes contain dataset attributes, branches reflect decision rules, and every leaf node reflect the result.

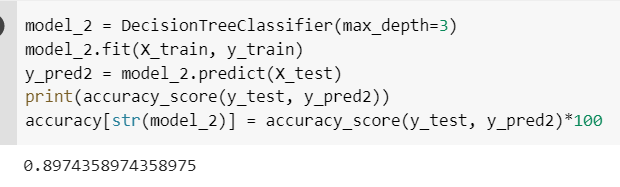
Advantages:

* A decision tree is simple to comprehend and analyze.
* Better solution and recommendations, as well as real data, can be provided.
* It can be used with other decision-making strategies.
* New situations are simple to add.

Disadvantages

* When variables with many levels are utilised in such a decision tree, the variables with more levels receive more detail.
* Equations can rapidly grow exceedingly complex, however this is generally only an issue if the tree is generated manually.

Output :



* K – Nearest Neighbours:

The k-nearest neighbours algorithm, often referred as KNN or k-NN, is a non-parametric, supervised machine learning classifier that employs proximity to classify or predict the grouping of a single data point. While it may be implemented either for the regression or classification issues, it is most commonly utilised as a classification method, based on the idea that comparable areas can be discovered nearby.

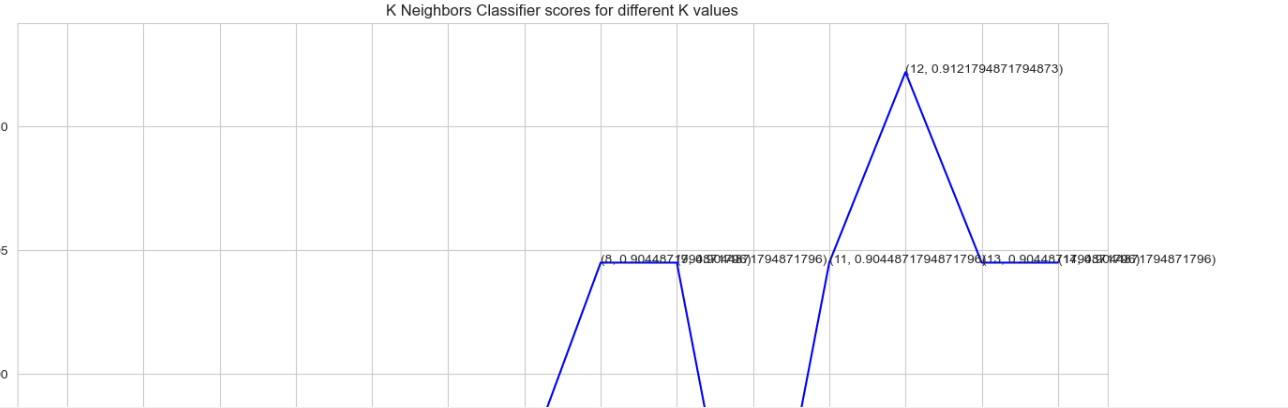
Advantage:

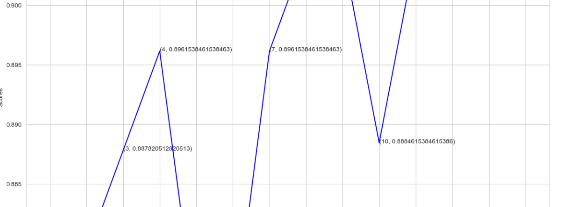
* The algorithm is straightforward and simple to implement.
* There is no need to create a model, fine-tune many parameters, or create new assumptions.
* The algorithm is adaptable. It has applications in classifications, regression, and search.

Disadvantages

* As the number of cases and/or explanatory variables increases, the method becomes much slower.

Output:





* Random Forest Classification:

A random forest is a nonlinear predictor that employs average to increase predicted efficiency and limit over-fitting by matching a variety of decision trees on different sub-samples of the dataset. Its convenience of use and versatility, as well as its ability to tackle regression and classification challenges, have boosted its popularity.

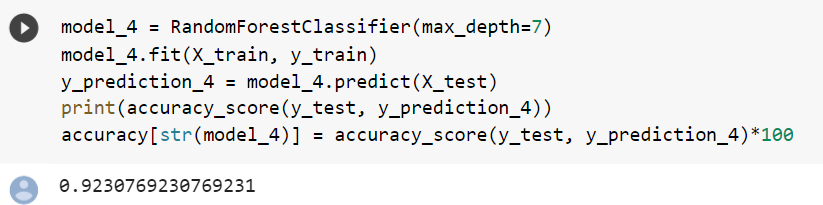
Advantage:

* **Reduced risk of overfitting.**
* **Provides flexibility.**
* **Easy to determine feature importance.**

Disadvantage:

* T**ime-consuming process.**
* **Requires more resources.**

Output:



Hence, in the following ways we were able to apply all the four methods and find out the optimal method for classification.

Conclusion:

In this project we have performed classification on the echocardiogram dataset i.e trying to predict using 4 classification algorithms which is K-Nearest Neighbours, Random Forest classification, Decision Tree classification, Logistic Regression. It is evident from the output results that Random Forest Classifier gives the best accuracy of 92% in forecasting if a patient will survive for at least one year following the heart attack whereas K-Nearest Neighbours classifier giving an accuracy of 91% with k value equal to 12, for Decision tree the accuracy rate is 89% for max-depth of the tree equal to 3 and 82% for Logistic Regression. Considering the dataset the few drawbacks present in it such as unequal distribution of dataset, size of the dataset, etc resolution would have helped in training the model on a diverse set of input data so that better and accurate predictions could be resulted.

Reference:

* Leclerc, S., Smistad, E., Pedrosa, J., Østvik, A., Cervenansky, F., Espinosa, F., Espeland, T., Berg, E. A. R., Jodoin, P.-M., et al. Deep Learning for Segmentation Using an Open Large-Scale Dataset in 2D Echocardiography. IEEE Transactions on Medical Imaging, 38(9): 2198–2210, 2019
* Wegner, F.K. *et al.* (2022) *Accuracy of deep learning echocardiographic view classification in patients with congenital or structural heart disease: Importance of specific datasets*, *MDPI*. Multidisciplinary Digital Publishing Institute. Available at: https://www.mdpi.com/2077-0383/11/3/690 (Accessed: December 15, 2022).
* Weintraub, W.S. and Ba'albaki, H.A. (1989) “Decision analysis concerning the application of echocardiography to the diagnosis and treatment of mural thrombi after anterior wall acute myocardial infarction,” *The American Journal of Cardiology*, 64(12), pp. 708–716. Available at: https://doi.org/10.1016/0002-9149(89)90752-2.
* Harrison, O. (2019) *Machine learning basics with the K-nearest neighbors algorithm*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (Accessed: December 15, 2022).
* Edgar, T.W. and Manz, D.O. (2017) “Exploratory study,” *Research Methods for Cyber Security*, pp. 95–130. Available at: https://doi.org/10.1016/b978-0-12-805349-2.00004-2.
* *Decision tree algorithm in Machine Learning - Javatpoint* (no date) *www.javatpoint.com*. Available at: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm (Accessed: December 15, 2022).

Appendix - Code

1)importing required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

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

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.preprocessing import StandardScaler

2) importing dataset and displaying it

dataset = pd.read\_csv('echocardiogram.csv')

dataset.head()

3)Data cleaning

dataset.isnull().sum()

null\_values\_percentage = dataset.isnull().sum() \* 100 / len(dataset)

null\_values\_percentage=null\_values\_percentage.sort\_values(ascending=False)

# Figure Size

fig, ax = plt.subplots(figsize =(16, 9))

# Horizontal Bar Plot

ax.barh(null\_values\_percentage.index, null\_values\_percentage.values)

# Remove axes splines

for s in ['top', 'bottom', 'left', 'right']:

ax.spines[s].set\_visible(False)

# Remove x, y Ticks

ax.xaxis.set\_ticks\_position('none')

ax.yaxis.set\_ticks\_position('none')

# Add padding between axes and labels

ax.xaxis.set\_tick\_params(pad = 5)

ax.yaxis.set\_tick\_params(pad = 10)

# Add x, y gridlines

ax.grid(b = True, color ='grey',

linestyle ='-.', linewidth = 0.5,

alpha = 0.2)

# Show top values

ax.invert\_yaxis()

# Add annotation to bars

for i in ax.patches:

plt.text(i.get\_width()+0.2, i.get\_y()+0.5,

str(round((i.get\_width()), 2))+'%',

fontsize = 10, fontweight ='bold',

color ='grey')

# Add Plot Title

ax.set\_title('Percentage of missing values',

loc ='left', )

# Show Plot

plt.show()

for labels in dataset.columns:

print(labels, ':' ,len(dataset[labels].unique()),'unique labels')

#removing columns of no use and having more missing values

dataset = dataset.drop(['name', 'group', 'aliveat1'], axis=1)

dataset.head()

#checking null values for features

feature\_null = [cols for cols in dataset.columns if dataset[cols].isnull().sum()>0]

for feature in feature\_null:

print(feature, ':', round(dataset[feature].isnull().mean(), 4), '%')

#dropping null values present in alive column

dataset = dataset.dropna(subset=['alive'])

dataset['alive'].isnull().sum()

4)Feature Selection

corr\_matrix = dataset.corr()

corr\_feat = corr\_matrix.index

plt.figure(figsize=(10,10))

#plot heat map

g=sns.heatmap(dataset[corr\_feat].corr(),annot=True,cmap="RdYlGn")

dataset.hist(alpha=0.5, figsize=(20, 10))

5)checking dataset balance

sns.set\_style('whitegrid')

sns.countplot(x='alive',data=dataset,palette='RdBu\_r')

6)Data Preprocessing

cols\_to\_scale= dataset.drop(['pericardialeffusion', 'alive'], 1).columns

label1 = ['alive']

print(cols\_to\_scale)

for cols in cols\_to\_scale:

dataset.boxplot(cols)

plt.title(cols)

plt.show()

feat\_with\_outliers = ['wallmotion-score', 'wallmotion-index', 'mult']

for feat in cols\_to\_scale:

if feat in feat\_with\_outliers:

dataset[feat].fillna(dataset[feat].median(), inplace=True)

else:

dataset[feat].fillna(dataset[feat].mean(), inplace=True)

from sklearn.neighbors import LocalOutlierFactor

loc = LocalOutlierFactor()

outliers = loc.fit\_predict(dataset)

#select all rows that are not outliers

mask = outliers != -1

#checking if any outlier present

dataset.isnull().sum()

7)Feature Scaling

dataset = dataset[mask]

dataset1 = pd.get\_dummies(dataset, columns = ['pericardialeffusion'], drop\_first = True)

scaler = StandardScaler()

dataset1[cols\_to\_scale] = scaler.fit\_transform(dataset1[cols\_to\_scale])

dataset1.head()

8)Train-Test Split

X = dataset.drop(['alive'], 1)

y = dataset['alive']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

X\_train.shape, X\_test.shape

accuracy = {}

9)Model Training and Testing

model\_1 = LogisticRegression(max\_iter=200)

model\_1.fit(X\_train, y\_train)

y\_prediction1 = model\_1.predict(X\_test)

print(accuracy\_score(y\_test, y\_prediction1))

accuracy[str(model\_1)] = accuracy\_score(y\_test, y\_prediction1)\*100

model\_2 = DecisionTreeClassifier(max\_depth=3)

model\_2.fit(X\_train, y\_train)

y\_pred2 = model\_2.predict(X\_test)

print(accuracy\_score(y\_test, y\_pred2))

accuracy[str(model\_2)] = accuracy\_score(y\_test, y\_pred2)\*100

from sklearn.model\_selection import cross\_val\_score

scores\_for\_knn = []

for k in range(1,15):

knn\_model\_3 = KNeighborsClassifier(n\_neighbors = k)

score=cross\_val\_score(knn\_model\_3,X,y,cv=10)

scores\_for\_knn.append(score.mean())

plt.figure(figsize=(15,15))

plt.plot([k for k in range(1, 15)], scores\_for\_knn, color = 'blue')

for i in range(1,15):

plt.text(i, scores\_for\_knn[i-1], (i, scores\_for\_knn[i-1]))

plt.xticks([i for i in range(1, 15)])

plt.xlabel('Number of Neighbors (K)')

plt.ylabel('Scores')

plt.title('K Neighbors Classifier scores for different K values')

knn\_model\_classification = KNeighborsClassifier(n\_neighbors = 12)

acc=cross\_val\_score(knn\_model\_classification,X,y,cv=10)

acc.mean()

model\_4 = RandomForestClassifier(max\_depth=7)

model\_4.fit(X\_train, y\_train)

y\_prediction\_4 = model\_4.predict(X\_test)

print(accuracy\_score(y\_test, y\_prediction\_4))

accuracy[str(model\_4)] = accuracy\_score(y\_test, y\_prediction\_4)\*10