MACHINE LEARNING PROJECT

1870175 -AMATULLAH MTHETHWA

1856080 – RUMBIDAZI MOYO
2111308 – MELISA MMATSHAKA
2094007 – VHUGALA MUDAU
2095458 – VUYO NCUME

HEART FAILURE CLINICAL RECORDS DATASET

DESCRIPTION OF THE DATASET

The dataset we have chosen to use comes from kaggle.com. It contains 13 attributes, which are used to determine mortality by heart failure in patients. Heart failure is a chronic condition in which the heart doesn't pump blood as well as it should. A person's health plays a big role in determining heart failure.

ATTRIBUTES

- ***Age*** – Age of patient in years. Ranges from 40-95 years old.
- ***Anemia*** – Decrease of red blood cells or hemoglobin (haematocrit levels were lower than 36%). It is represented in Boolean form, 0 meaning no and 1 meaning yes.
- ***Creatinine_phosphokinase*** – Level of the CPK enzyme in the blood. Measured in mcg/L. It ranges from 23-7876mcg/L.
- ***Diabetes*** – When blood glucose is too high. It is represented in Boolean form. 0 represents no diabetes and 1 represents having diabetes.
- ***Ejection_fraction*** – Percentage of blood leaving the heart at each contraction. It ranges from 14-80%.
- ***High_blood_pressure*** – If a patient has hypertension. It is represented in Boolean form. 0 represent no high blood pressure and 1 represents having high blood pressure.
- ***Platelets*** – Blood cells that help blood clot. This represents the platelets in the blood and is measured in kiloplatelets/mL. It ranges from 25.01-850.00kiloplatelets/mL.
- ***Serum_creatinine*** – Level of creatinine in the blood. It is measured in mg/dL and rages from 0.50-9.40mg/dL.
- ***Serum_sodium*** – Level of sodium in the blood. It is measured in mEq/L and ranges from 114-148mEq/L.
- ***Sex*** – Represents a woman or a man. It is represented in binary form. 0 represents a woman and 1 represents a man.
- ***Smoking*** – If the patient smokes or not. It is represented in Boolean form. 0 meaning no and 1 meaning yes.
- ***Time*** – Follow-up period. It is represented in days. It ranges from 4-285days.

TARGETS

- ***DEATH_EVENT*** – This represents if the patient died during the follow-up time. It is represented in Boolean form. 0 represents alive and 1 represents dead.

NUMBER OF DATAPOINTS: 299

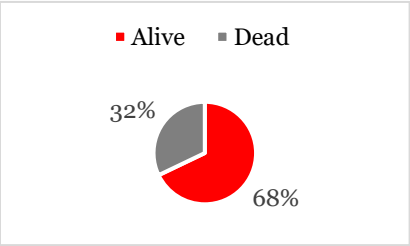
SAMPLE DATAPONTS FROM THE DATASET

	age	anaemia	creatinine_phosphokinas	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	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

The main goal from this data is to predict if a person is in high risk of dying due to heart failure or not from this data.

CLASS BALANCE

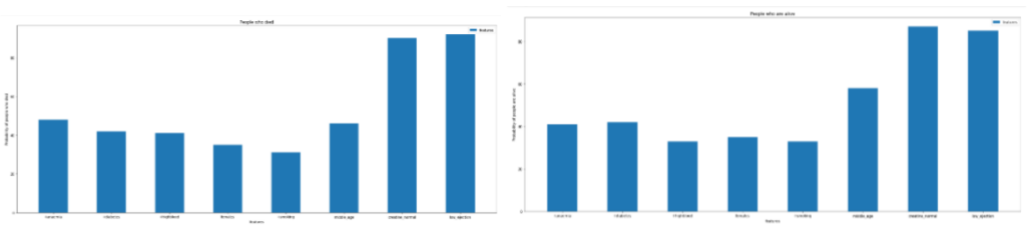
Classes: 0-Alive, 1-Deadx



These classes are imbalanced because they are not represented equally. There are 203 patients that are alive and 96 patients that died. To handle the imbalance it is possible to use different techniques to get better results.

This problem is important because it can improve the entire research and prevention process, making sure that people can live healthy lives. If is predicted well , it can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis.

Bar graph of probabilities of people who died and people who are alive for each feature.



DATA STRUCTURING

We first used the describe method [**data.descibe()**] to gain insight on how the dataset is distributed, and we obtained the following results:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000	299.000000	299.00000	299.000000	299.00000
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418	0.648829	0.32107	130.260870	0.32107
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477	0.478136	0.46767	77.614208	0.46767
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000	0.000000	0.00000	4.000000	0.00000
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000	0.000000	0.00000	73.000000	0.00000
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000	1.000000	0.00000	115.000000	0.00000
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	1.40000	140.000000	1.000000	1.00000	203.000000	1.00000
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000	1.000000	1.00000	285.000000	1.00000

How we structured your inputs/targets and normalized and preprocessed the data – For normalization we used the standard data normalization techniques min-max. The min-max technique preserves all the relationships in the original dataset exactly. The outliers of the data points usually have large values. In order to represent those large outlier data, we used the sigmoidal normalization is an appropriate approach.
What did we do with missing values – We deleted the values that have null values. A model trained with the removal of all missing values creates a robust model.
How we split into training/validation/test data – we set 60% of the data is for training, 20% for validation and 20% for testing. We split this way because more training data is nice because it means our model sees more examples and thus hopefully finds a better solution. The validation data helps us make a better decision about which model is the best. The testing data gives us a better sense of how well our model generalizes to unseen data. The most important thing is finding a better solution.

CLASSIFICATION ALGORITHMS WITH RESULTS

- DECISION TREE ALGORITHM
- LOGISTIC REGRESSION ALGORITHM
- NAÏVE BAYES ALGORITHM
- NEURAL NETWORK ALGORITHM

DECISION TREE ALGORITHM

We chose this algorithm because the decision tree approach is more powerful for classification problems. Compared to other algorithms decision trees require less effort for data preparation during preprocessing.

How it works: The decision tree algorithm tries to solve the problem, by using tree representation. Each **internal node** of the tree corresponds to an attribute, and each **leaf node** corresponds to a class label.

To implement this algorithm, we chose to use the following attributes: **'age', 'anaemia', 'diabetes', 'high_blood_pressure'**, and **'smoking'.**

For this algorithm, **normalizing is not necessary**, so we use unnormalized features. Decision trees **do not require an assumption of linearity in the dataset.**

How we implemented the decision tree algorithm:

1. Defining the x and y matrices for the training, validation and testing data. We defined the attributes we chose as the X matrices and DEATH_EVENT as the Y matrices.
2. Saving the feature list.
3. Defining the dictionary of hyperparameters. We used 10 samples and put the maximum depth as 5.
4. Setting up and training the algorithm.
5. Printing the tree structure.
6. Making prediction on **training data,** getting confusion matrix and accuracy of the model after training. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
after training:
 [[80 20]
 [32 48]]
accuracy:  71.11111111111111 %
```

7. Making prediction on **validation data**, getting confusion matrix and accuracy of the model after training. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
validation:
 [[29 18]
 [ 7  6]]
accuracy:  58.333333333333336 %
```

8. Finding the best hyperparameters.
9. Validating with the new hyperparameters.
10. Making prediction on **testing data**, getting confusion matrix and accuracy of the model after training . The following is the **result** of the confusion matrix and the **final accuracy of the model:**

```
final results:
 [[50  6]
 [ 3  0]]
accuracy:  84.7457627118644 %
```

HYPERPARAMETERS WE SELECTED

max tree depth = 5. We chose this because we can't have the depth of our tree to be more than the number of features used.

minimum split samples = 10. We chose this to avoid overfitting for specific data points we set it not to split on data points less than 10.

No regularization was used because model didn't overfit thus regularizing wasn't necessary.

LOGISTIC REGRESSION ALGORITHM

Logistic regression is one such relatively used machine learning algorithms for studies involving risk assessment of complex diseases. Thus, the study intends to identify the most significant predicators of cardiovascular diseases and predicting the overall risk by using logistic regression.

We chose this algorithm because Logistic regression is easier to implement, interpret, and very efficient to train. The number of observations is more than the number of features so it wont lead to overfitting.

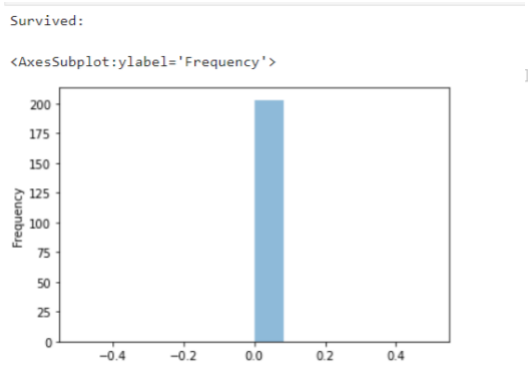
How it works: Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

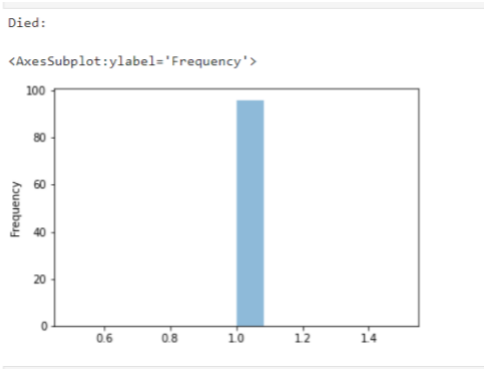
How we implemented the logistic regression algorithm:

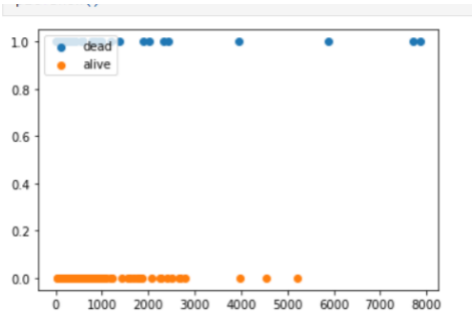
1. Defining the x and y matrices for the training, validation and testing data. We creatinine_phosphokinase as the X matrices and DEATH_EVENT as the Y matrices.

	1	creatinine_phosphokinase
0	1	582
1	1	7861
2	1	146
3	1	111
4	1	160

2. Splitting the deaths and survivals. The following are frequency and scatter plot graphs of deaths and survivals:







3. Using closed form solution because case is simple.
4. Getting confusion matrix and Accuracy from **training** data. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
Confusion Matrix:
 [[97  3]
 [76  4]]

Accuracy:  56.111111111111114 %
```

5. Getting confusion matrix and Accuracy from **validation** data. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
Confusion Matrix:
 [[47  0]
 [13  0]]

Accuracy:  78.33333333333333 %
```

6. Getting confusion matrix and Accuracy from **testing** data. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
Confusion Matrix:
 [[56  0]
 [ 3  0]]

Accuracy:  94.91525423728814 %
```

There are no hyperparameters because the closed form solution was used. This type of solution was used because it was feasable on our data set and would lead to a more accurate result than an iterative solution.

No regularization was used because model didn't overfit thus regularising wasn't neccessary.

The basis function used was a straight line because a scatter plot of our features versus our target showed that there was a linear relationship/correlation.

NAÏVE BAYES ALGORITHM

We chose this algorithm because it is fast and easy to predict the class of the test data. The algorithm also performs better when compared to other classification models such as logistic regression, provided that the assumption of independence holds. The Naïve Bayes algorithm also doesn't require as much training data.

How it works: It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class.

To implement this algorithm, we chose the following features**: anaemia', 'diabetes', 'highblood', ' females', 'smoking', 'middle_age', 'creatine_normal' and 'low_ejection'.**

How we implemented the naïve bayes algorithm:

1. Applying the algorithm to the training data for DEATH_EVENT == 1. Probability:


```
{'age': [0.41025641025641024, 0.3333333333333333, 0.528735632183908], 'anaemia': [0.43617021276595735, 0.4534883720930232], 'cr
eatine_phosphokinase': [0.4146341463414634, 0.49367088607594933, 0.36842105263157887], 'diabetes': [0.42201834862385323, 0.4788
7323943661975], 'ejection_fraction': [0.507936507936508, 0.34615384615384615, 0.24999999999999994], 'high_blood_pressure': [0.4
205607476635514, 0.47945205479452047], 'gender': [0.46774193548387094, 0.4322033898305084], 'smoking': [0.4576271186440678, 0.4
193548387096774], 'DEATH_EVENT': []}
```


2. Applying the algorithm to the training data for DEATH_EVENT == 0. Probability:


```
{'age': [0.5128205128205129, 0.4166666666666667, 0.660919540229885], 'anaemia': [0.5452127659574467, 0.566860465116279], 'creat
ine_phosphokinase': [0.5182926829268293, 0.6170886075949367, 0.460526315789473 7], 'diabetes': [0.5275229357798165, 0.5985915492
957746], 'ejection_fraction': [0.634920634920635, 0.4326923076923077, 0.3125], 'high_blood_pressure': [0.5257009345794392, 0.59
93150684931507], 'gender': [0.5846774193548387, 0.5402542372881356], 'smoking': [0.5720338983050848, 0.5241935483870969], 'DEAT
H_EVENT': []}
```


3. Getting confusion matrix and Accuracy from **training** data. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
confusion matrix :
[[100   0]
 [ 80   0]]

accuracy =  55.55555555555556
```

4. Getting confusion matrix and Accuracy from **validation** data. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
confusion matrix :
[[47  0]
 [13  0]]

accuracy =  78.33333333333333
```

5. Getting confusion matrix and Accuracy from **testing** data. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
confusion matrix :
[[56  0]
 [ 3  0]]

accuracy =  94.91525423728814
```

This algorithm does not have hyperparmeters because it generalizes very well.

NEURAL NETWORK ALGORITHM

We chose this algorithm because Neural Networks have the ability to learn by themselves and produce the output that is not limited to the input provided to them. They can perform multiple tasks in parallel without affecting the system performance.

How it works: They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

To implement this algorithm, we chose to use the following attributes: **'age'**, **'anaemia'**, **'diabetes'**, **'high_blood_pressure'**, and **'smoking'.**

How we implemented the neural network algorithm:

1. Defining the x and y matrices for the training, validation and testing data. We defined the attributes we chose as the X matrices and DEATH_EVENT as the Y matrices.
2. Initialize a network.
3. Sum the inputs.
4. Define the sigmoid function.
5. Define the derivative of the sigmoid.
6. Forward propagate input to a network output
7. Backpropagate error and store in neurons
8. Update the weights.
9. Train the network.
10. Make a prediction of the network.
11. Make prediction.
12. Initialize network with 5 inputs, 2 hidden layers with 5 nodes each, 1 output.

```
network:
 [[{'weights': [0.8672964064713189, 0.051918323518298615, 0.8777221545630779, 0.023192825496437086, 0.3412197075504807, 0.8266648435687826]}, {'weights': [0.38036999422258855, 0.941094353723295
6, 0.08481676532756377, 0.5340291732638601, 0.9661679690103335, 0.38199258878588616]}, {'weights': [0.9600309729955925, 0.9828389510858212, 0.1343150672828377, 0.10635462088176206, 0.70723218391
55744, 0.4771313752711108]}, {'weights': [0.2111694057297625, 0.9870006428351343, 0.8442212729207369, 0.6574906761540925, 0.7422752673104499, 0.13645811347942227]}, {'weights': [0.33394102911831
08, 0.2257800584484012, 0.0010166776099179708, 0.4677248573360696, 0.7185812596049536, 0.6567453775082727]}], [{'weights': [0.6180274355040279, 0.34051019859539755, 0.0914185678162861, 0.8568026
270026244, 0.78650892047583 92, 0.6348591604173996]}, {'weights': [0.01679842750313343, 0.09882684106026318, 0.28943375654483416, 0.7545008568202504, 0.08570497401157218, 0.0216302904830884]},
{'weights': [0.832442121846556, 0.7417283539929196, 0.7471616925449094, 0.45814991520238757, 0.31900065094525 4, 0.021564668734668 78]}, {'weights': [0.6085716629051997, 0.6048112580010441, 0.7850
845412328894, 0.935367339897664, 0.8978739355174928, 0.6121983615814056]}, {'weights': [0.9124003274814104, 0.23256267858565516, 0.5787761835488827, 0.9069107629752275, 0.7710402207441086, 0.140
15441099620685]}]], [{'weights': [0.030209936845224217, 0.4992654830665 4395, 0.7578457220274444, 0.7512999386066084, 0.9066022681311016, 0.01995014188899269]}]]
```

13. Making prediction on **training data,** getting **confusion matrix** and accuracy of the model a fter training. The following is the **result** of the confusion matrix and the **accuracy of the mo del**:

```
Confusion matrix:
 [[100   0]
 [ 80   0]]

Accuracy:  55.55555555555556 %
```

14. Making prediction on **validation data,** getting **confusion matrix** and accuracy of the model after training. The following is the **result** of the confusion matrix and the **accuracy of the model**:

```
Confusion matrix:
 [[47  0]
 [13  0]]

Accuracy:  78.33333333333333 %
```

15. Adding new hyperparameters.
16. Comparing the two networks on **testing data**. **Result:**

```
comparing the two networks:
first one:
Confusion matrix:
 [[56  0]
 [ 3  0]]

Accuracy:  94.91525423728814 %
versus the tuned one:
Confusion matrix:
 [[56  0]
 [ 3  0]]

Accuracy:  94.91525423728814 %
```

learning rate = 0.01, epochs = 1000, epsilon = 0.001.
We chose the learning rate to be small enough (2 decimal places) so that we were less likely miss the optimum (values between 0.01-0.09 were tried and 0.01 worked the best). the number of epochs were chosen because most algorithms chose 1000 as their standard number of epochs, also changing the number of epochs once the learning rate was set had no notable affects. We would have been satisfied with results correct to 2 decimal places, so we chose an epsilon accurate to 2 decimal places.

No regularization was used because model didn't overfit when number of hidden layers wasn't above 2 thus regularizing wasn't necessary.
We didn't have basis functions but did have activation functions because it's a neural network. The activation function used was a sigmoid function because we needed the output to be a value between 0 and 1. We used a Stochastic Gradient Descent.

ANALYSING RESULTS

Heart failure prediction accuracy may be a problem that appears. For better heart failure prediction accuracy, we propose a feature selection method that uses a floating window with adaptive size for feature elimination. After the feature elimination, we can apply the algorithms.

Randomly guessing the class would not produce good results as you have only a small chance of guessing correctly.

DISCUSSION

Considering the data set has 299 points after cleaning and has approximately 3 to 5 outliers or noise data points, without overfitting the highest accuracy you can obtain is 98,33 %.
Logistic Regression, Naïve Bayes and Neural Network were found to be the best algorithms. These algorithms work quickly and can save a lot of time. They are easy to maintain and all efficient to train. For the Naïve Bayes algorithm, because the assumption of the independence of features holds true, it performs better than other models and required much less training data.
Decision Tree algorithm was the worst because the **stability of decision trees** depends on the number of leaves in the tree.

Comparison Table:

Algorithm	***Accuracy***
Decision tree	84.75%
Logistic Regression	94.92%
Naïve Bayes	94.92%
Neural Network	94.92%

For someone trying to work with this data, we would recommend working with a collection in a wide variety of clinical settings but not with too many outliers. And also the data should be splitted accordingly into training, validation and testing data.

NEURAL NETWORK

This Algorithm struggled with Outliers of points which didn't follow the trend such as having high blood pressure, diabetes and being old but not being susceptible to heart failure.
Time taken for the Neural Network Algorithm to run:

```
Time taken to run 76.01252350000141
```

LOGISTIC REGRESSION

The data points we struggled with are data points that have high creatinine_phosphokinase levels but surviving.
Time taken for the Logistic Regression Algorithm to run:

```
Time taken to run:  0.20749129999967408
```

DECISION TREE

The data points we struggled with are outliers of points which didn't follow the trend such as having high blood pressure, diabetes and being old but not being susceptible to heart failure.

Time taken for the Decision Tree Algorithm to run:

```
Time taken to run:  3.767344100000628
```

NAÏVE BAYES

The data points we struggled with are outliers of points of the features we chose. Another struggle was the Zero probability problem. This problem was solved by adding value one to the frequency of each attribute.

Time taken for the Naïve Bayes Algorithm to run:

```
Time taken to run :  0.3803215820007608
```

REFERENCES

- Bhanot, K. (2019, MAY 03). *Predicting presence of Heart Diseases using Machine Learning*. Retrieved from Towards datascience: https://towardsdatascience.com/predicting-presence-of-heart-diseases-using-machine-learning-36f00f3edb2c
- https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ by : Sunil Ray Date : Originally published on Sep 13th, 2015 and updated on Sept 11th, 2017
- https://www.geeksforgeeks.org/naive-bayes-classifiers/ by : Sambhav Khurana Date : 15 May, 2020
- https://www.youtube.com/watch?v=7O_cya8q3BA&t=607s by : Sebastian Mantey date : 13 Nov 2020
- https://www.youtube.com/watch?v=g9Su_zR-LYk&t=702s by : Sebastian Mantey date : 14 Nov 2020
- https://www.youtube.com/watch?v=N0QslF05MyQ by : Sebastian Mantey date : 15 Nov 2020
- https://www.researchgate.net/publication/228845263_An_Empirical_Study_of_the_Naive_Bayes_Classifier by:Irina Rish date : January 2001
- Lecture Notes by : Benjamin Rosman date : 2021 Lecture slides
- D. Chicco, G. Jurman. "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone", 2020
- Lupón J, Vila J, Bayes-Genis A. Risk prediction tools in patients with heart failure.**JACC Heart Fail**. 2015; *3*:267. doi: 10.1016/j.jchf.2014.10.010.