GENERATING A DATA MODEL FOR PREDICTING RECOVERY or DEATH of THE HEART-FAILURE PATIENT

Syed Nazmul Kabir

Date: 22nd May 2021

TABLE OF CONTENTS

ABSTRACT:

The purpose of this analysis is to establish two machine learning models for predicting survival or death incidents of the heart-failure patients. This analysis is addressing a pure classification problem of data-science. While building-up models, the analysis also focuses on important features that have strong relationships with the death events of the heart-failure patients. So, the analysis will also find out the key factors contributing towards increased risk of mortality among heart failure patients.

Before creating and implementing data models, each attribute in the dataset is explored using appropriate visualisations and descriptive statistics. Then the relationship between pairs of attributes is identified following the same approach. Finally, two data models are selected to train the given dataset according to the findings from the data exploration activities. In this report, a detailed explanation will be provided focusing on how the models are trained up and how the models are validated.

INTRODUCTION:

The heart failure clinical records dataset is a public dataset that is available from the UCI Machine Learning Repository website, it contains the medical records of 299 heart failure patients. Data is collected during their follow-up period and this dataset has 13 clinical features (UCI Machine Learning Repository,2020).

The 13 clinical features include: age, anaemia, high blood pressure, creatinine phosphokinase, diabetes, ejection fraction, platelets, sex, serum creatinine, serum sodium, smoking, time, and death event. With this dataset, we are interested in the death event, which is our target feature for the data model.
Before developing our model, we first retrieve the data and prepare the data for exploration. Then we analyse the attributes that covers the exploration of searching relationships between different attributes. From the data exploration, we develop our model and finally we tune our model to achieve better performance.

The two models we have chosen for this data set are the Decision Tree model and the K-Nearest Neighbour model. We will further explain the reasons behind the model selection in the report later. We will also explain how we have identified the best parameters for our models and present our findings.

METHODOLOGY:

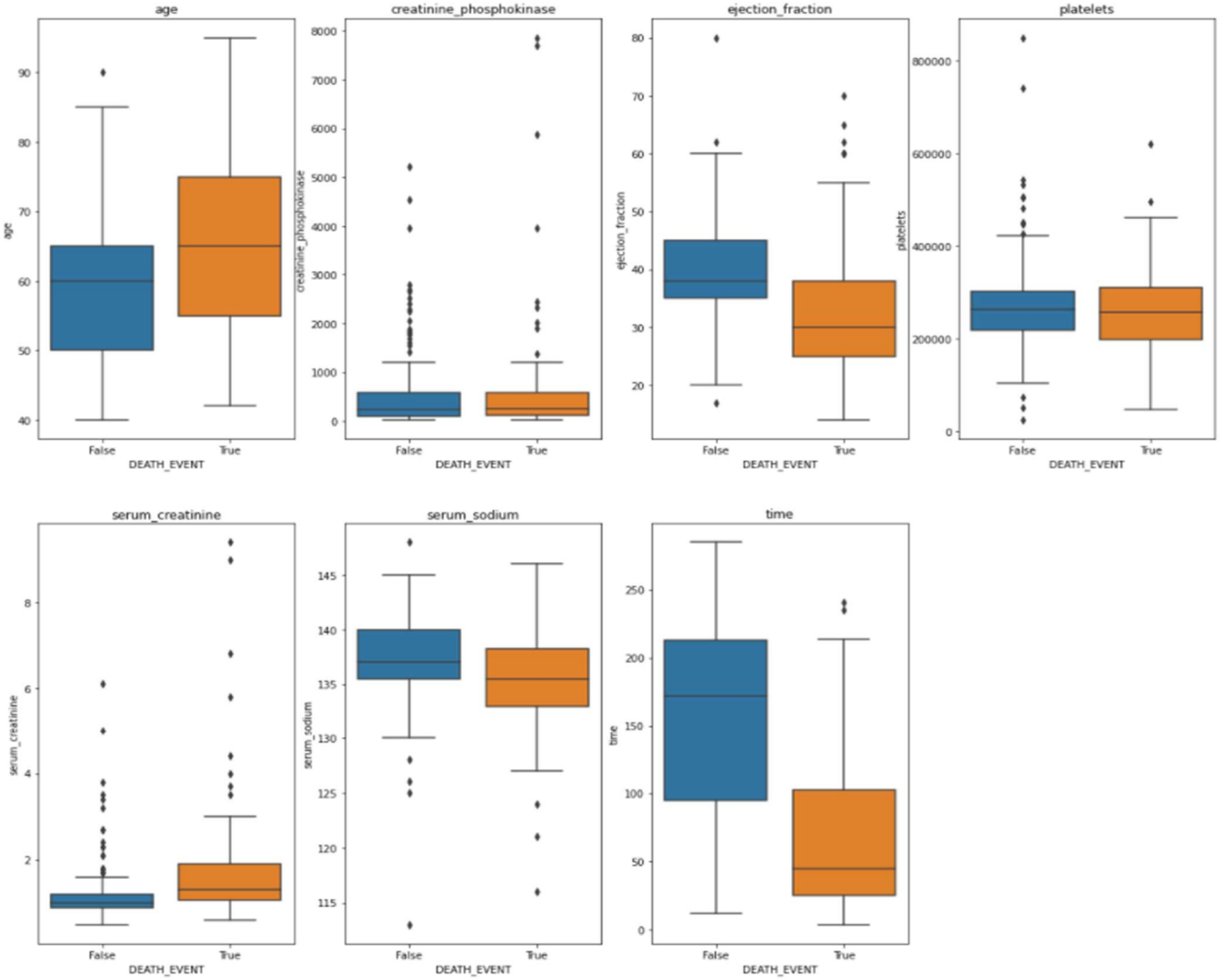
1. Retrieving and Preparing the Data

First, we start with retrieving the data using panda and preparing it for data exploration. After retrieving the data, we have checked whether there are any null values in the dataset. From the code we know that there is no such data. Then we check the imported data types. By checking the data types, we can see that the ‘anaemia’, ‘diabetes’, ‘high_blood_pressure’, ‘smoking’ and ‘DEATH_EVENT’ attributes are incorrectly labelled as int64 data type when they should be the Boolean data type. The ‘sex’ attribute is also incorrectly labelled as type category where it should be the category data type. Therefore, we have changed these columns to the correct data type according to the data description.

Then we have produced a serial of box charts check the distribution of every attributes against the target value to find out whether there are any data outliers as shown in figure 1.1 of next page. The box charts shows that there are some data that fall very far from the median. From the chart of ‘creatinine_phosphokinase’ distribution amongst HF patients’, we see that the upper whisker of the box chart is approximately 1200 for both patients who was dead and alive. But 7 patients had creatinine phosphokinase over 3000, from the chart we can see two patients’ creatinine phosphokinase level even over 7000. These levels are extremely high compared to the majority. Similarly, from the ‘platelets distribution amongst HF patients, we can see that 3 patients had platelets level of over 600000 units and is much higher than the median, which is of approximate 300000. For the ‘serum_creatinine’ distribution amongst heart-failure (HF) patients’ we observe that the median level is very different depending on the death event. The medium is significant higher on a patient who was not able to survive. However, we can still see data outliers: there are 5 patients had serum creatinine that is above 5. Another attribute that has outliers is the serum_sodium attribute. From the ‘serum_sodium’ attribute distribution amongst HF patients’ boxplot, we can see there are 2 patients had serum sodium level below 2.

To solve the data outlier issue, we have removed a few of these data that are extremely high or extremely low. we kept some of the data that are relatively high or low. The reason of keeping some of the high/low values is that there was lack of guidelines on maximum/minimum value of different medical data features. A diagnosis on a particular heart failure patient may result extreme data, such as extremely high serum creatinine level or a very low ejection fraction. Those are not invalid or impossible data at all.

Figure 1.1: Box plot showing distribution of all numerical attributes with respect to DEATH_EVENT attributes in heart failure dataset



2. Data Exploration

Data exploration has been conducted following two approaches:

1) Exploring individual attributes of the dataset
2) Exploring relationships between pair of attributes

2.1 Exploration of Individual Attributes:

2.1.1 Exploration of 'age' column:

The descriptive statistics is clearly showing that the patients age varies from 40 to 95 of years. The bar graph of 2.1.1 of ipynb file indicates that maximum number of heart-failure patients are in the range of 60 to 70 years that is closely followed by the age range of 50 to 60 years. Both the old and young age groups occupy less than 10 number of patients.

2.1.2 Exploration of ‘serum_sodium’ column:

The boxplot of figure number 2.1.2 of ipynb file tells us that serum sodium level among patients vary in the range of 120 to just below of 150 mEq/L. However, majority of the patients hold serum sodium content from 134 to 140 mEq/L. The distribution is almost a normal distribution.

2.1.3 Exploration of ‘platelets’ column:

The histogram of platelets of ipynb file attributes shows that patient’s platelets varie in a wide range. The distribution is positively skewed. Majority of the patient’s platelet level falls in between 200000 to 300000 kiloplatelets/mL

2.1.4 Exploration of ‘time’ column:

Plot 2.1.4 of ipynb file shows that ‘time’ data are distributed with positive skewness. Though follow-up days varies from 4 to 285 days, majority of patient undergo follow-up days from 73 to 203 days.

2.1.5 Exploration of 'ejection_fraction' column:

Graph 2.1.5 of ipynb file illustrates that ejection fraction of 30 to 40% is the most common figure among the heart-failure patients. Very low (10-20%) and very high (60-80%) ejection fraction levels are relatively very uncommon among the patients.

2.1.6 Exploration of 'sex' and boolean variables:

- Fig. 2.1.6 of ipynb file indicates that majority of the patients were not suffering from high-blood pressure.
- Nearly same percentage of heart-failure patients had anaemia (42%) or diabetes (44%) disease.
- Two-third of heart failure patients are male. One-third of heart failure patients are smoker.

2.1.7 Exploration on 'serum_creatinine' feature:

The histogram of 2.1.7 of ipynb file emphasises the very high skewness (positive) of the distribution. Half of the patient’s serum creatinine content is around 1 mg/dL. Only a few numbers of heart-failure patients had serum creatinine content above 2.5 mg/dL.

2.1.8 Exploration of 'DEATH_EVENT' column:

This is the target column of this analysis. The bar graph 2.1.8 of ipynb file tells us that survival events in heart-failure incident is more than two times of death events among all heart-failure patients.

2.2 Exploring Relationships Between Pair of Attributes:

At first, a pair plot (fig 2.2 of ipynb file) (that uses mainly scatter plot) is used to find out relationships among all numerical attributes in the data. Then we check the correlation using the panda’s correlation function and visualise the correlation in a heatmap (fig 2.2.0 of ipynb file). The scatter plots for all the attributes does not indicate any clear strong relationship between the numerical attributes. However, the correlation heap map for all attributes shows some strong correlation among few pair of attributes. It also confirms that smoking status, diabetes, and gender of patients have no effect on death events. In the section below, we have listed ten attribute pairs and provide important insights regarding what relationship they may have with each other.

2.2.1 Relationship between 'age' and 'DEATH_EVENT' columns:

- Age has direct effect on death events. The mean value in descriptive statistics is clearly showing that patients who survived in heart failure case is on average seven years younger than those who died.
- The bar graph in the next page of fig 2.2.1 indicates higher death rate when the patient is aged above 70 years. Survival rate is 50-50 in the age ranged from 70 to 80 years. On the other hand, large majority of the patient died for heart failure cases at the age above 80 years.

2.2.2 Relationship between 'anaemia' and 'DEATH_EVENT' feature:

- 14% of total heart patients who died were suffering from anaemia disease.
- 40% of total patients who survived had no anaemia.

- However, 47% dead-patients were suffering from anaemia diseases.
- On the other-hand, 42% survived patients were suffering from anaemia diseases.
- So, anaemia positive percentage is higher in dead people compare to that of alive patients.

Figure 2.2.1: Relationship between 'age' and 'DEATH_EVENT'

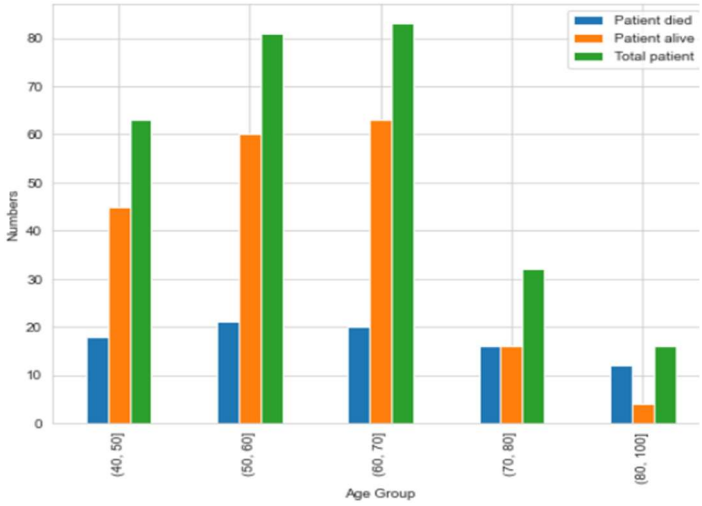


Figure 2.2.2.1 Anaemia Positive or Negative Percentages among survived patients

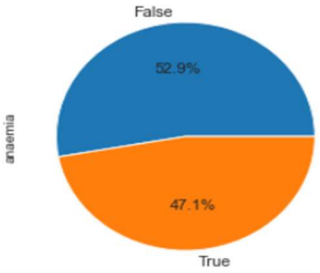


Figure 2.2.2.2 Anaemia Positive or Negative Percentages among dead patients

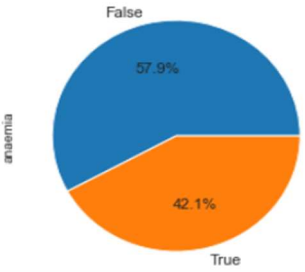
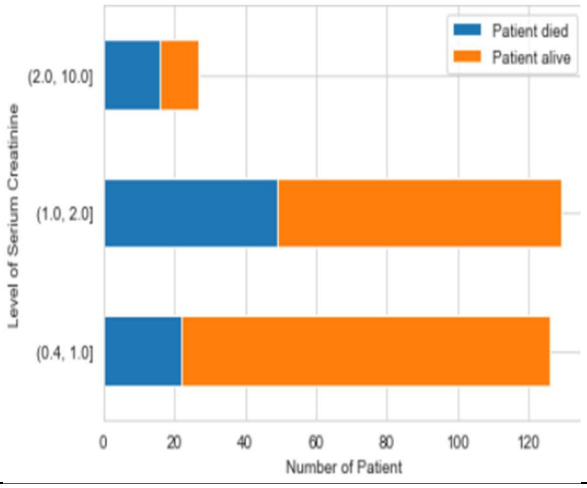


Figure 2.2.3 Relationship between 'time' and 'DEATH_EVENT'

	Patient died	Patient alive	Total patient	Death % in Same follow-up Time Group
(0, 50]	45	6	51	88.2%
(50, 100]	19	47	66	28.8%
(100, 150]	9	43	52	17.3%
(150, 200]	10	28	38	26.3%
(200, 300]	4	71	75	5.3%

Figure 2.2.4 Relationship between 'serum_creatinine' and 'DEATH_EVENT'



2.2.3 Relationship between 'time' and 'DEATH_EVENT' feature

The table: 2.2.3 clearly shows an inverse relationship in between these two columns. 88% patients died who did not go through more than 50 days of follow up time. On the other hand, only four patients died out of 75 (5%) who followed up more than 200 days. So, time and DEATH_EVENT has strong negative relationship which is also confirmed by the heat-map (value: -.53) shown in section-2.2.13 of heart_failure.ipynb file.

2.2.4 Relationship between 'serum_creatinine' and DEATH_EVENT columns:

Figure 2.2.4 indicates clearly that death percentages is increased as the serum_creatinine level increases among the heart-failure patients. Heat-map also shows strong positive (0.29) correlation among these two features.

2.2.5 Relationship between 'smoking' and ‘sex’ columns:

Only 1.1% of total heart-patients are smoker who are female [Figure 2.2.5 of ipynb file]. That means, among all smokers, only 3.4% are female and rest of all are male. So, smoking and sex have very strong relationship (0.45 as shown by the heat-map)

2.2.6 Relationship between 'diabetes' and ‘DEATH_EVENT’ columns:

Among dead persons, the percentage of being diabetic is 42.5% which is very much similar to recovered patients (Fig 2.2.6 of ipynb file). So, there is no relationship in between these two features.

2.2.7 Relationship between 'ejection_fraction' and 'DEATH_EVENT' feature:

Figure 2.2.7 illustrates the negative relationship between these two features. Death percentage is very high in the case of lower percentages of ejection fraction and vice versa. The boxplot is also showing that median value of ejection fraction for dead patient is lower than that of alive patient. Heat-map also shows the significant negative co-efficient(-0.27).

2.2.8 Relationship between 'serum_sodium' and 'DEATH_EVENT' features:

According to the figure number 2.2.8, low level of serum sodium content cause higher risk of mortality among heart-failure patients. Heat-map also shows the significant negative co-efficient(-0.20)

Figure 2.2.7 Relationship between 'ejection_fraction' and 'DEATH_EVENT'

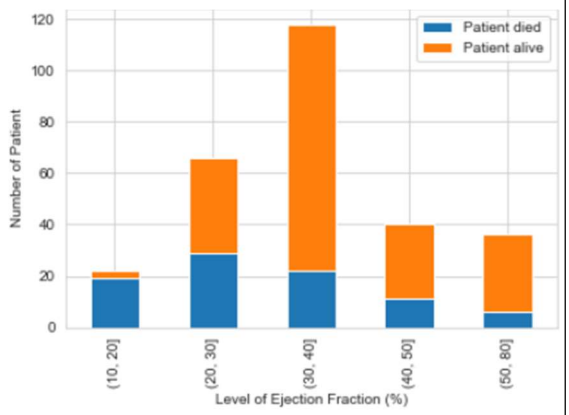


Figure 2.2.8 Relationship between 'serum_sodium' and ‘DEATH_EVENT’

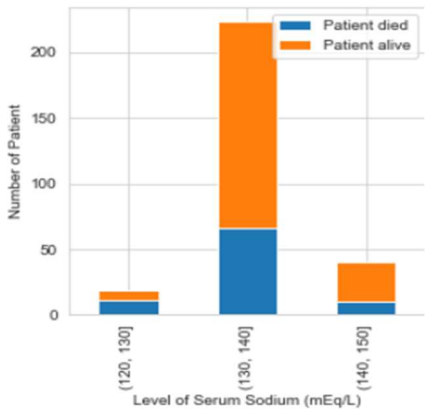


Figure 2.2.9 Relationship between ‘serum_creatinine’ and 'age'

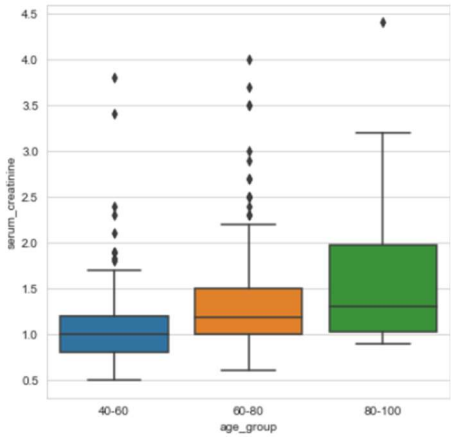
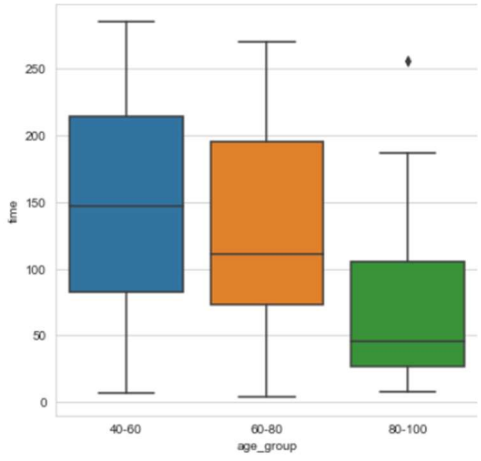


Figure 2.2.10 Relationship between 'age' and 'time'



2.2.8 Relationship between 'age' and 'serum_creatinine' features:

The boxplot in fig 2.2.9 shows that with growing of ages, serum creatinine level among the patients varied in a higher range.

2.2.9 Relationship between 'age' and 'time' features:

The boxplot in fig 2.2.10 of previous page confirms that number of follow up days decreases with the increase of age.

3 Data Modelling

As we know the target column 'DEATH_EVENT' is a Boolean type data and it can be true or false value. To predict the 'DEATH_EVENT', we need to build a classification data model from the given 12 attributes. In this section we have chosen Decision Tree and K-Nearest Neighbours model as our data-modelling approach.

3.2 Decision tree data model

Decision tree is a machine learning algorithm that is mostly used in classification problems, which is suitable for our problem in this dataset. Because in our dataset there are numerical and categorical variables, decision tree is a good choice as it works on both types of data. A decision tree algorithm will create a flowchart-like structure from the training data, the branch of the decision tree is a possible outcome, and the leaf node represents a class label. Each path on a tree represents a classification rule.

To build our model, we first split our heart failure dataset into 80% and 20%, which the 80% of data will be use in model training. The rest of the data will be used to validate or test our developed model. After splitting the data, we first train our base model using a random criterion (which is a 'gini' algorithm in our case). Then we check the performance of our base model using the classification report and confusion matrix. From figure 3.1.1, we can see that this model is predicted 30 false value correctly and predicted 11 true value correctly. It has not predicted any false value for a true class but predicted 7 true value when they belong to a false class.

We also used a classification report to understand the model performance, as shown in figure 3.1.1. The precision for the true class is lower than the false class, which means that the accuracy of the positive predictions is worse in the true class. However, the true class has a perfect recall, which means it is successfully identified all the true values in the true class. This score reflects the data in our confusion matrix. Both classes have acceptable f1-scores being 0.92 and 0.76 and the overall accuracy is 0.88 which is considerably accurate as a base model.

Figure 3.1.1 Base Model Classification Report and Confusion Matrix for Decision Tree

	precision	recall	f1-score	support
False	1.00	0.85	0.92	46
True	0.61	1.00	0.76	11
accuracy			0.88	57
macro avg	0.81	0.92	0.84	57
weighted avg	0.92	0.88	0.89	57

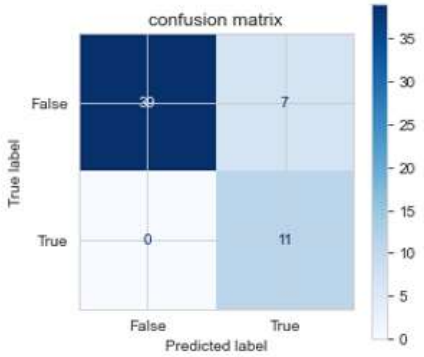
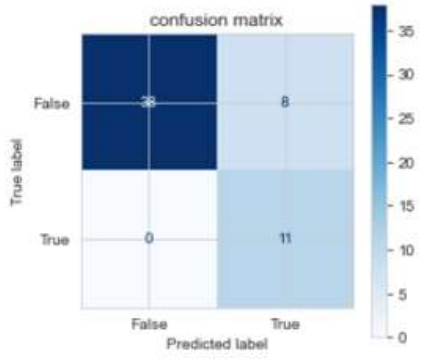


Figure 3.1.2: Final Model Classification Report and Confusion Matrix for Decision Tree

	precision	recall	f1-score	support
False	0.98	0.96	0.97	46
True	0.83	0.91	0.87	11
accuracy			0.95	57
macro avg	0.91	0.93	0.92	57
weighted avg	0.95	0.95	0.95	57



Then we work on hyperparameter tuning to improve the model. First, we used the entropy algorithm and check whether that improves our performance. After training the model we can see that the performance is better using the 'gini' algorithm as both the f1-score and accuracy is better from the previous data model. Then we

test other parameters including max_depth, min_samples_split and min_samples_leaf to find out the best model. From the experiment, we can see when max_depth =5, min_samples_split =2 and min_samples_leaf =1, the model performs the best on our testing data. Therefore, we use it as our final decision tree model. Fig 3.1.3 shows the final decision tree model.

Out[62]:

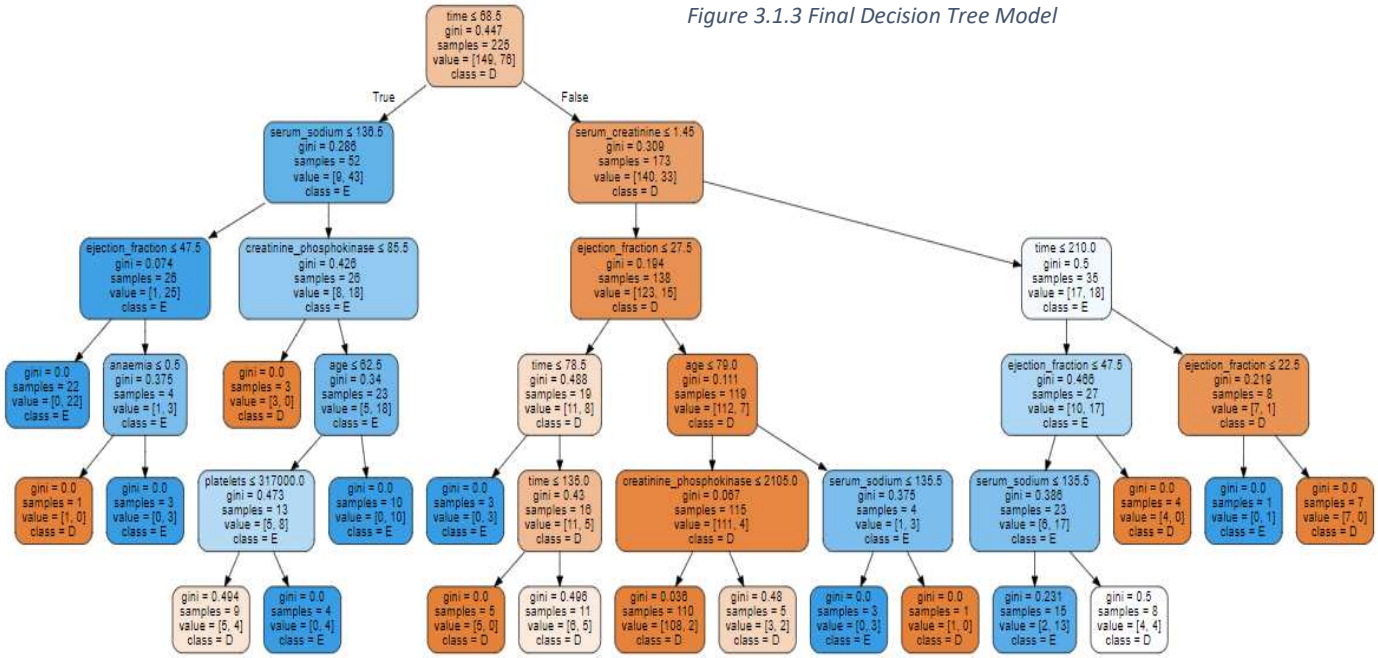


Figure 3.1.3 Final Decision Tree Model

If we look closer to the visualisation of decision tree model, we can see that the first few splits are using the 'time', 'serum_sodium', 'serum_creatinine', 'ejection_fraction' and 'creatinine_phosphokinese'. From the previous data exploration, we understand that most of these features are highly correlated to the 'DEATH_EVENT'.

3.3 K-nearest neighbour data model:

K Nearest Neighbours is a simple and a versatile (applied in classification, regression, search) supervised learning model. Here we have used this model to categorize observations into classes. KNN model works by finding distances between a query and all examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label for addressing classification problem.

From decision tree modelling and exploratory analysis, we have already seen that some features are stronger in predicting the value of target variable. So, in KNN model implementation, we directly started model build-up through feature selection using hill climbing method. While performing feature-selection, we split our heart failure dataset into 80% and 20%. Details of hill climbing method and reason of splitting dataset has been described in previous section. Here we just used KNeighborsClassifier instead of DecisionTreeClassifier. The parameters that we select initially for the base model are:

k=5, metric='minkowski', weight='distance' and p=1.

During feature selection, hill climbing technique delivers the feature names for training sets. Those are 'serum _sodium', 'time', 'age', 'serum_creatinine', 'ejection_fraction' and 'sex'. Target variable is DEATH_EVENT. After getting the features, the details of model build-ups are shown. The figure of 3.2.1 of next page is showin g confusion matrix. It indicates that initial model predicts 41 false values and 11 true values correctly. It has not predicted any false value for a true class but predicted 7 true value when they belong to a false class.

Classification report of same figure 3.2.1 clearly shows that the precision for the true class is considerably lower than the false class. So, accuracy of the positive predictions is worse in the true class. However, the true class has a perfect recall, which means it is successfully identified all the true values in the true class. This score reflects the data in our confusion matrix. Both classes have acceptable f1-scores being 0.94 and 0.81 and the overall accuracy is 0.91 which is good enough for a base model.

Figure 4.2.1 Base Model Classification Report and Confusion Matrix for KNN model

	precision	recall	f1-score	support
False	1.00	0.89	0.94	46
True	0.69	1.00	0.81	11
accuracy			0.91	57
macro avg	0.84	0.95	0.88	57
weighted avg	0.94	0.91	0.92	57

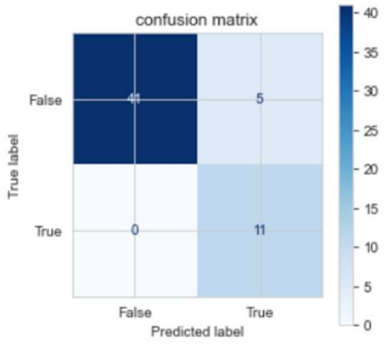
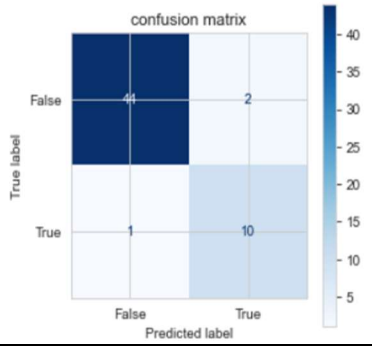


Figure 3.2.2: Final Model Classification Report and Confusion Matrix for KNN model

	precision	recall	f1-score	support
False	0.98	0.96	0.97	46
True	0.83	0.91	0.87	11
accuracy			0.95	57
macro avg	0.91	0.93	0.92	57
weighted avg	0.95	0.95	0.95	57



For increasing model efficiency, we tune the hyperparameter comprehensively. We check 'k' values ranging from 1 to 7 for p=1 or 2 or 3. These parameters are placed while setting metric as minkowski against both uniform and distance weights. So, Manhattan (1), Euclidean (2) and Minkowski (3) distance metrics are being cross-checked while tuning parameters. The precise tuning reveals that accuracy is increased to .95 from 0.91 of base model. According to figure 3.2.2 in previous page; though precision for False class and recall for True class is decreased a little bit, precision for True class and recall for False class are increased significantly. F1-score for both classes is also showing improved values.
The ultimate selected parameters are:

K=6, metric='minkowski', weight='uniform' and p=1

RESULTS:

Both the models show the score of 0.95 after performing tuning on hyper parameters. Implementation of a validation technique on those models ultimately helps to find out better model.

To validate the developed models, we have used the K-Fold Cross Validation to check our model performance. The K-Fold Cross Validation is a validation method by splitting a dataset into K sections or folds and each fold shall be used as a testing set in each iteration. In our validation we have chosen the K value as 10. So, the dataset will be divided into 10 small sets and will iterate 10 times. Each time one of the small subsets will be used to validate the training model. Our 10-folds validation ensures sufficient testing on both of our models using different subsets of data.

From our Cross Validation report, we see that K-Nearest Neighbour model performs better for all folds except fold 4 & 5; hence its performance is steadier than Decision Tree model across the dataset. The scores of both models for K-Fold Cross Validation are shown in the next page:

Figure 4.1.1 K-Fold Cross Validation scores for Decision Tree model

```
[fold 0] score: 0.793
[fold 1] score: 0.759
[fold 2] score: 0.750
[fold 3] score: 0.750
[fold 4] score: 0.821
[fold 5] score: 0.821
[fold 6] score: 0.786
[fold 7] score: 0.893
[fold 8] score: 0.857
[fold 9] score: 1.000
```

Figure 4.1.2: K-Fold Cross Validation scores for KNN model

```
[fold 0] score: 0.897
[fold 1] score: 0.759
[fold 2] score: 1.000
[fold 3] score: 0.857
[fold 4] score: 0.786
[fold 5] score: 0.750
[fold 6] score: 0.893
[fold 7] score: 0.893
[fold 8] score: 0.893
[fold 9] score: 1.000
```

DISCUSSION:

From the results outlined above, we are satisfied that both of our models performed well on the given dataset. It is worth to point out from all the classification reports that the true value ('DEATH_EVENT' = 1) always have lower f1-score and precision. This is because of imbalanced dataset in terms of True death event. In the data exploration section, we see that we have more data on 'DEATH_EVENT' = 0.

To solve this issue, we have utilized the K-Fold Cross Validation on our models. Different subsets have given us different combinations of 'DEATH_EVENT' frequency thus reliability of the model is increased. Another possible solution would be resampling of the dataset: under-sampling for the larger class or over-sampling for the insufficient class (Wu & Radewagen, 2017).

Moreover, during the decision tree model training, we have noticed its' instability nature. Although we have improved the model by putting constraints such as hyperparameter tuning to improve the overfitting issue, it is worth of mentioning that a small change in the dataset can result in a data model with varying output. One solution of this issue may be using the random forest model to improve the decision tree, which means we can create multiple decision trees and combine the output of all trees to generate the final output (Sharma, 2020). This process is known as ensemble learning that can improve average prediction performance (Brownlee, 2020)

From the research, we know that K Nearest Neighbours model can be slow for a large dataset (Vaibhab, 2020). Space complexity (refers to the total memory used by the algorithm) is also enormous for KNN model. Building the model based on 300 rows does not cause problem here. However, imposing the model on unseen large dataset may arise time and space complexity issues. To overcome the problem of time complexity, algorithms such as KD-Tree and Locality Sensitive Hashing (LSH) can be used, which is not covered in this report.

CONCLUSION:

The exploratory analysis on all major features helps us to reach at a definite conclusion. The key factors contributing towards increased risk of mortality among heart failure patients are growing age, higher serum creatinine, high blood pressure, presence of anaemia and lower values of ejection fraction (EF). Increased level of serum sodium can reduce the odds of death too. On the other hand, smoking status, diabetes and gender of patients hardly play any role in determining mortality among heart-failure patients.

Decision Tree and K-Nearest Neighbours models are used here to train and test the dataset. Both the model works well as each of the model scores same accuracy (0.95) to predict the mortality among heart-failure patients. However, K-fold Cross Validation technique reveals more stability or less variance in K-Nearest Neighbours model while testing on heart-failure dataset compared to the decision tree model. But KNN may face time and space complexity for large datasets. So, if the unseen data is very large, KNN might be a slow model to implement. On the other hand, decision tree model is more vulnerable to the variation of the data distribution, hence less stable.

In conclusion, if time and space complexity is not the issue, K Nearest Neighbours model is preferable to the decision tree model. On the contrary, if data variation is not significant and it is large dataset, decision tree model is better to use.

REFERENCES

Ahmed T., Munir A., Bhatti S., Aftab M. & Reza M. (2017, July 20). Survival analysis of heart failure patients: a case study. *Plos One.* https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0181001 (accessed May 19, 2020)

Boschetti, A. & Massaron, L. (2016) *Python Data Science Essentials* (2nd ed.). Packt Publishing.
matplotlib. (2021).

Brownlee, J. 'What are the Benefits of Ensemble Methods for Machine Learning?'. (2020, October 26). *Machine Learning Mastery*, https://machinelearningmastery.com/why-use-ensemble-learning/#:~:text=A%20key%20benefit%20of%20using,made%20by%20the%20contributing%20models (accessed May 19, 2021)

Chicco, D. & Jurman, G. (2020). Heart failure clinical records Data Set. *UCI Machine Learning Repository*. https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records (accessed May 20, 2021)

RMIT Canvas. (2021). *Practical Data Science with Python COSC2670: Modules: Week4,5,6,7* https://rmit.instructure.com/courses/79792/modules (accessed May 19, 2020)

scikit learn. *API Reference*. https://scikit-learn.org/stable/modules/classes.html (accessed May 19, 2020)

Sharma, A. A Simple Analogy to Explain Decision Tree vs. Random Forest. (2020, May 12). *Analytics Vidhya*, https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm (accessed May 18, 2021)

Vaibhab, J.(2020, Aug 25) K-Nearest Neighbours (KNN) algorithm*, towards data science* https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-23832490e3f4#:~:text=5.-,Limitation%20of%20KNN,memory%20used%20by%20the%20algorithm. *(*accessed May 20,2020)

Wu, Y. & Radewagen, R. (2017). 7 Techniques to Handle Imbalanced Data. *KD nuggets*. https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html (accessed May 18, 2021)