**Project Title: Predicting Heart Attack Probability Using Patient Data**

# Heart Disease Model Report

**Author: Robert Ferdinand**

**Email: UOFLOUISIANA@GMAIL.COM**

**Project Title: Predicting Heart Attack Probability Using Patient Data**

## *Table of Contents*

**Project Title: Predicting Heart Attack Probability Using Patient Data**

***Section 1: Introduction of Problem***

Medical data on several patients is collected over a period of time. This data consists of features such as age, sex, body weight, resting blood-pressure, resting EKG, cholesterol levels and so on. In total there are 14 features. Additionally the data on each patient has a target label zero (0) if the patient did not suffer a hear attack and a target label one (1) if the patient DID suffer a heart attack. Using this labeled data, a research facility would like to look at feature data for any patient and be able to predict whether the respective patient has a high probability of suffering a heart attack in the near future, labeled as 1, or NOT, labeled as 0. In case the patient has a high probability of suffering an imminent heart attack, one would resort to using preventative medicine to avoid an impending disaster.

To be able to make such a prediction, one would need to analyze the data, to be able to "learn" from it. This will be done using Machine Learning (ML) and Deep Learning (DL) which are both Data Science tools. The PYTHON programming language and all of its excellent packages and modules will be fully utilized to accomplish the task at hand.

**Project Title: Predicting Heart Attack Probability Using Patient Data**

***Section 2: The Data***

The patient data is acquired from the open-source website Kaggle.com. This website has several sets of highly interesting and accurate data sets that have important real-world, societal applications. The patient data in particular contains information on features of several patients taken over time. The features include quantities such as age, sex, body weight, heart blood-pressure, cholesterol-level, resting EKG and so on. There are a total of 14 features in all. The data also contains a label or target column which indicates the patient suffered a heart attack (1) or has NOT suffered a heart attack (0).

All this points in the direction of a supervised or labeled, classification-based Machine Learning (ML) and/or Deep Learning (DL) project.

The data will need to be cleaned of any missing values and then removed of any duplicate records or rows. Also the data will need to be normalized so that Machine Learning (ML) and/or Deep Learning (DL) algorithms can work efficiently with it. Finally we will need to make sure that all of our data makes sense. For example the age of a patient would need to be an integer and so on.

We note here that the data is in a comma-separated version (CSV) file that will need to be read into a technology platform that will be ultimately used for ML and DL.

**Project Title: Predicting Heart Attack Probability Using Patient Data**

***Section 3: The Methodology***

In this section we use the data set, obtained from Kaggle, that is described in the section above. Then we proceed with learning from this data set as follows:

***Step I: Data Cleaning***

(a) Import data set from Kaggle and read the .CSV file as a pandas Data Frame.

(b) Check Data Frame for any missing or not number (NaN) data values. If there are missing data, the respective rows are deleted from the Data Frame.

(c) Drop any duplicate rows of the Data Frame to avoid redundancy.

(d) Ensure that all columns are of data types that makes sense. For example the age of a patient should be an integer and so on.

***Step II: Data Exploration***

(a) Using the PYTHON SCIPY.STATS package we calculate the following statistics for each feature column. Note these statistics will NOT be computed for the target or label column: *minimum, maximum, mean (average), standard deviation, skewness and kurtosis*. These statistics help us determine whether any of the data is outlying in its data set and should probably be removed.

(b) Using SCIPY.STATS again we calculate the correlation matrix for the feature columns. Matrix values inform us whether any of the features are highly correlated to each other in which case

multiple feature columns that ARE highly correlated can be reduced to one or two columns. This helps with redundancy and ease of modelilng as well.

(c) Using PYTHON's MATPLOTLIB package and a random sample of n = 100 data points we create pair-wise scatter plots between the feature columns. Scatter plots present a visualization about how the feature columns may or may not be correlated to each other. Later, in the modeling section, we perform a Principal Component Analysis (PCA) of the data by reducing its dimensionality to k = 3 to further discover the presence of correlation or even clustering.

***Step III: Data Extraction, Transformation and Loading (ETL)***

(a) First we note that none of the feature columns contains data that is textual. Hence the need for encoding the respective columns using the popular ONE-HOT ENCODING method is not necessary in this case. However, the usefulness of one-hot encoding does need to be mentioned for any Data Science project.

(b) Next, several of the feature columns have numeric data that needs to be standard normalized to data with mean or average 0 and standard deviation 1. This is carried out by writing a PYTHON function on the respective feature columns that gets this accomplished.

(c) From above, our data should now be available and ready for the analysis that follows. I would like to emphasize the important of this step since it is real helpful to the models we develop in the future.

***Step IV: Modeling***

This is the most FUN part of the project. We apply the Machine Learning (ML) and Deep Learning (DL) algorithms of Data Science to help us learn from the data set. Here we proceed as follows:

(a) The data set from Step III above is converted into an Apache Spark Data Frame. This to enable us to able to apply the available Apache Spark's Machine Learning (ML) and Keras Deep Learning (DL) algorithms to our model.

(b) Machine Learning Model-1 (ML1): Here we split our data set into Training set (80%) and Testing Set (20%). Then we use the Gradient Boosting Tree (GBT) Classification Method, since we have a supervised ML problem with target or labeled data. A pipeline method is followed to set up the implementation of this ML model. A MultiClassClassificationEvaluator is used to measure the accuracy of the prediction of the model. The metric used is ACCURACY.

(c) Machine Learning Model-2 (ML2): Here we split our data set into cross evaluation Training set (80%) and Testing Set (20%), with k = 4 random states, using TRAIN_TEST_SPLIT function from SKLEARN's MODEL_SELECTION package. Then we use the Gradient Boosting Tree (GBT) Classification Method once again along with the pipeline method. A MultiClassClassificationEvaluator is used here as well to measure the accuracy of the prediction of the model with the ACCURACY metric.

(d) Principal Component Analysis (PCA): Principal Component Analysis (PCA) or dimensionality reduction is performed on the feature columns of the data set. This reduces the dimension of the data set to k = 3. This is followed by a 3-D graph of the data set to detect any clustering or correlation among the data. Apache Spark's PCA algorithm is utilized.

(e) Deep Learning Model-1 (DL1): Using the randomly split data into Training Set (80%) and Testing or Validation Set (20%), we employ a Keras Deep Feed Forward Neural Network (NN). Here we use a Keras Sequential model with one dense input layer with RELU activation function, one dropout layer and one dense output layer with SOFTMAX output function. The model is compiled with loss = 'SPARSE_CATEGORICAL_CROSSENTROPY', optimizer = 'SGD' (steepest gradient descent method) and metrics = ['ACCURACY']. The model is fit to the training data and run over several EPOCHS. Care is taken to have the BATCH_SIZE be a divisor (with zero remainder) of the training data set size. The model is then evaluated over the test data to determine its accuracy. The loss and accuracy are then recorded.

(f) Deep Learning Model-2 (DL2): The train-test data set from DL1 in (e) above is used. A Keras Deep Feed Forward neural network (NN) is employed. A Keras sequential model with one dense input layer with RELU activation function, one dropout layer and one dense output layer (1 neuron only) with SIGMOID output function is implemented. The model is compiled with loss = 'BINARY_CROSSENTROPY', optimizer = 'SGD' (steepest gradient descent method) and metrics = ['ACCURACY']. The model is fit to the training data and run over several EPOCHS. The BATCH_SIZE is set to be a divisor (with zero remainder) of the training data set size. The model is then evaluated over the test data to determine its accuracy. The loss and accuracy are recorded.

(g) Note: Many other models can be used here as well. For example, in parts (e) and (f) one can employ several other metrics such as F-1 SCORE and so on.

**Project Title: Predicting Heart Attack Probability Using Patient Data**

***Section 4: The Results***

In this section we discuss the results from Section 3 above. The analytical and graphical results are available to view in the IPYTHON notebook *HeartDisease.model.ipynb* uploaded on Github:

(a) The uni-variate statistics from our Data Exploration do not reveal the presence of heavily skewed data or the presence a large number of outliers in any of the data in the feature columns.

(b) The multivariate statistics using the Correlation Matrix values do not show a strong pairwise correlation between any two of the column features. Hence all the feature columns data available will be useful in creating our Machine Learning (ML) and Deep Learning (DL) models.

(c) The Machine Learning Model ML1 gives us a 99.58% accuracy score. Hence the data fits the model almost perfectly.

(d) Machine Learning Model ML2 gives us a 99.59% accuracy score. The fit is almost exact here as well. It's only a very slight improvement from ML1 for comparison purposes.

(e) The PCA analysis with k = 3 gives us a 3D graph. The total absence of clustering eliminates redundancy from feature columns data.

(f) Deep Learning Model DL1 presents a Loss Score of 0.32 and an Accuracy Score of 0.88 which indicates a high accuracy of model fitting the data.

(g) Disappointingly, Deep Learning Model DL2 submits a Loss Score of 0.70 and an Accuracy Score of 0.49 which DOES NOT indicate a good model fit. Could a non-sequential Keras model have worked better here? A good question for further consideration!

**Project Title: Predicting Heart Attack Probability Using Patient Data**

***Section 5: Discussion and Conclusion***

Pandas DataFrames, Apache Spark Machine Learning and Keras Deep Learning Neural Networks were used extensively in this project. The implementation was carried out on a PYTHON notebook on IBM Watson Studio platform which proves to be an indispensable resource.

Fitting the model to the data gave us excellent results using the Apache Spark Machine Learning algorithms but were met with slightly lesser success with the Keras Deep Learning techniques. It may be that the data set is not large enough for a Keras Deep Learning Feed Forward Neural Network method. Another possibility is that a Keras non-sequential model with an Embedding Layer may have worked better. These are questions for future consideration.

Either way, we have a model that can be used to predict whether a patient has a high probability of suffering a heart attack in the near future (label = 1) or NOT suffering a heart attack in the near future (label = 0) based on feature information about the patient such as age, sex, body weight, blood pressure, cholesterol level and so.

The model can be presented or sold to a research or medical facility to use as a diagnostic tool. Certainly the model can further be refined using more data to train it. Further, the model can be tuned as well, using different parameters.