iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VIII **Month of publication:** August 2022

DOI: https://doi.org/10.22214/ijraset.2022.46341

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VIII Aug 2022- Available at www.ijraset.com

Heart Disease Prediction Based on Machine Learning

Aarya Shah[1], Rutvik Patel[2]
[1, 2]*Graduate, Department of Electronics & Instrumentation Engineering, Institute of Technology, Nirma University, Ahmedabad, Gujarat-382481*

Abstract: Now-a-days, if you know how to analyze the data and derive conclusions from it, then data becomes extremely valuable. And the main reason for that is the growing importance of using previous data to predict possible future scenarios with higher accuracy. We have used a dataset of around 70000 patients having symptoms related to heart diseases along with their daily activities and medical measurements like heart rate, cholesterol level, blood pressure and used 5 different binary prediction machine learning models for predicting the chances of a person getting heart related diseases in future based on the values entered by any person into our program. If you are a beginner in machine learning and you do not know where to begin or how to use machine learning then this paper will be extremely helpful for you to learn how machine learning works.
Keywords: machine learning; data; heart disease prediction; machine learning models.

I. INTRODUCTION

As more and more technology is evolving, data privacy debates are also increasing, but along with that data is extremely beneficial to us, because of the opportunity to predict the future based on historical data and outcomes with an higher accuracy gives us an edge to be prepared for the upcoming scenarios like the one we are dealing with in this paper. Just think how much more you will take care of yourselves if you can predict the chances of you being afflicted by a heart disease in future based on your current habits and medical report? There is an actual possibility that you can even avoid such situation in future if you knew your chances, and that is the actual power of data.

II. STEP BY STEP PROCESS

A. Developing Generic Program

While implementing different models there was a set pattern of steps which we followed due to which we can make a generic program which anyone can use to predict the chances of getting a heart disease even with a different dataset and with minimal knowledge.

B. Importing the Data

Firstly, we imported the dataset and deleted the incomplete or blank data entries.

C. Data Pre-processing

Then we pre-processed each and every input parameter in which we tried to normalize data distribution of each parameter which basically removes the outliers and helps us improve our prediction accuracy as well.

D. Train Test Split

In this we basically separate the data that will be used for training the model and 20 random testing data entries, so that we can check whether our model can predict outcomes with accuracy outside of our training data or not and hence avoid over- fitting of the model on dataset (When testing accuracy and training accuracy are poles apart because the model is too focused on the data that we provided).

E. Implementing Different Models

Now we basically implement the machine learning model we want to implement.

F. Testing Accuracy of the Model

We will find out both training and testing accuracy (on data which the model has never seen before) of the model.

G. Trying for real life patients

We will use the model which gives us the highest training and testing accuracy for getting real life patient data and predicting it

III. WHY WE SELECTED THE PARTICULAR DATASET

The dataset we selected contained data points of over 70000 patients which helped us to make accurate predictions. Along with that it had 11 different parameters which were extremely common, so anyone can use our program. Also, this dataset was genuine because all other datasets available online were showing weird insights like odds of getting heart disease increase with decrease in age, which makes no sense. That's the reason we used this dataset and you can see the parameters in figure 1.

Information about the DataSet:

- *Age Final* | Objective Feature | Years(int)
- *Gender* | Objective Feature | 1-Women, 2-Men
- *Height* | Objective Feature | Cm(int)
- *AP_HI* (Systolic Blood Pressure) | Examination | int
- *AP_Lo* (Diastolic Blood Pressure) | Examination | int
- *Cholesterol* | Examination | 1:Normal, 2:Above Normal, 3:Well Above Normal
- *Gluc* (Glucose) | Examination | 1:Normal, 2:Above Normal, 3:Well Above Normal
- *Smoke* | Subjective Feature | 0:No, 1:Yes(binary)
- *Alco* (Alcohol Intake) | Subjective Feature | 0:No, 1:Yes(binary)
- *Active* (Physical Activity) | Subjective Feature | 0: No, 1: Yes(binary)
- *Cardio* (Presence of cardiovascular Disease) | Target Variable | 0: No, 1: Yes(binary)

Figure: 1 Parameters in the Dataset

IV. PRE-PROCESSING OF DATA

A. Data Import

You can see the code we used to import our dataset in figure 2.

Importing relevant Libraries

```
In [33]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.linear_model import LogisticRegression
         from sklearn.linear_model import LinearRegression

         from sklearn.model_selection import train_test_split
         import statsmodels.api as sm
         sns.set()
```

Importing the data

```
raw_data=pd.read_csv('cardio_train.csv')
raw_data.head()
```

	id	age final	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	50	2	168	62.0	110	80	1	1	0	0	1	0
1	1	55	1	156	85.0	140	90	3	1	0	0	1	1
2	2	52	1	165	64.0	130	70	3	1	0	0	0	1
3	3	48	2	169	82.0	150	100	1	1	0	0	1	1
4	4	48	1	156	56.0	100	60	1	1	0	0	0	0

Figure: 2 Importing the Libraries and Dataset

B. Normalizing Age Parameter

As you can see in figure 3 that there are only few entries on left most part of part of the graph, so for normalizing what we did is we eliminated the lowest 0.01 percentile data to make it a normal distribution.

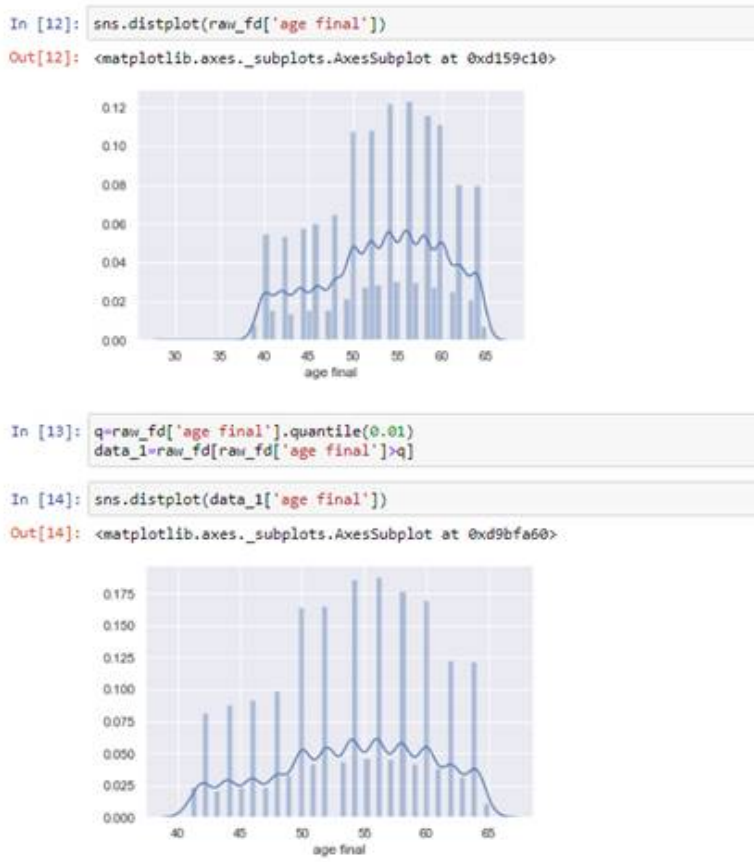


Figure: 3 Before & After Data Distribution of Age Parameter

C. Height Parameter

As you can see height parameter is evenly distributed as you can see in figure 4, so there is no need of normalizing.

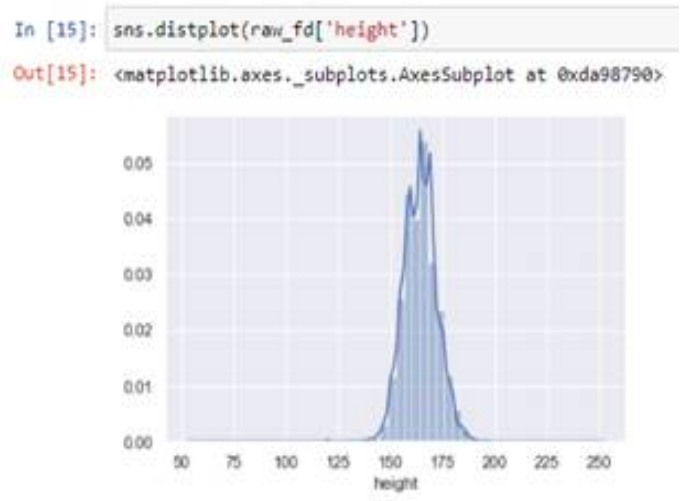


Figure: 4 Data Distribution of Height Parameter

D. Normalizing Weight Parameter

As you can see in figure 5 that the graph is extended from the bottom part on the right, so we simply eliminated top 0.01 percentile data to make it a normal distribution.

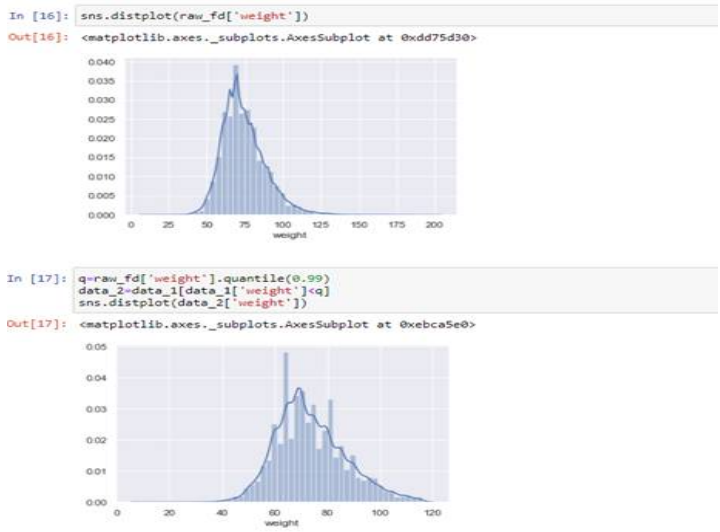


Figure: 5 Before & After Data Distribution of Weight Parameter

E. Normalizing AP_HI Parameter

As you can see in figure 6 that the original graph is extended from both the sides, so, we eliminated both top and lowest 0.01 percentile to achieve normal distribution as shown in figure 7.

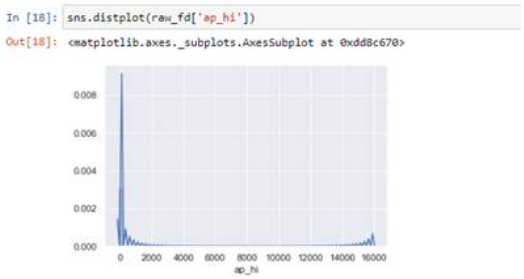


Figure: 6 Original Distribution of AP_HI Parameter

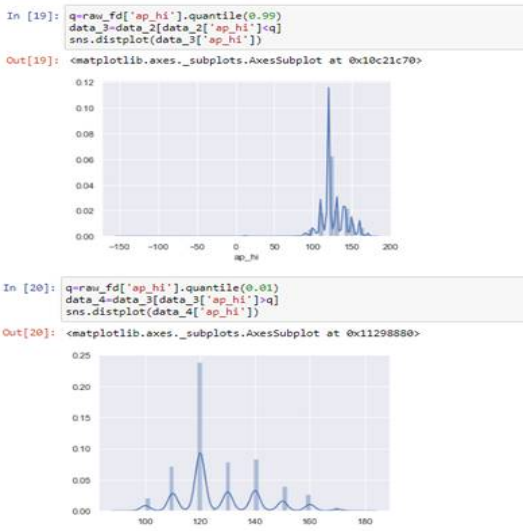


Figure: 7 Normalizing AP_HI Parameter

F. Normalizing AP_LO Parameter

As you can see in figure 8 that the original graph is extended from right side so, we eliminated both top and 0.014 percentile to achieve normal distribution. Also, this is the last parameter for normalizing as there are no more value based parameters remaining.

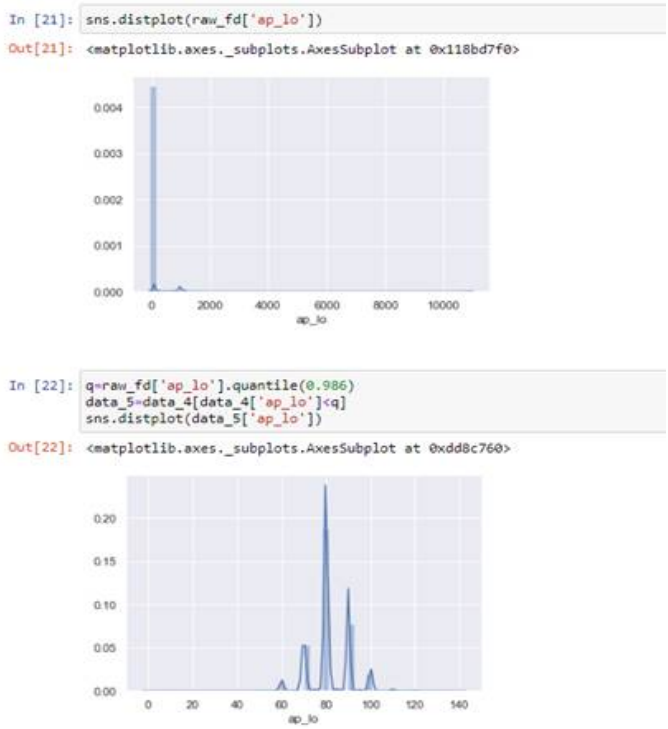


Figure: 8 Before and After Data Distribution of AP_LO Parameter

G. Train Test Split

After data pre-processing and eliminating the outliers we are left with 63228 data entries out of 70000.
Now we are randomly extracting 20 data entries to check out the accuracy after training each model.

Train Test Split

```
a_train,a_test=train_test_split(a,test_size=20,shuffle=False,random_state=362)

a_train
```

	age final	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	50	2	168	62.0	110	80	1	1	0	0	1	0
1	55	1	156	85.0	140	90	3	1	0	0	1	1
2	52	1	165	64.0	130	70	3	1	0	0	0	1
3	48	2	169	82.0	150	100	1	1	0	0	1	1
4	48	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
63203	47	1	165	76.0	140	90	1	1	0	0	1	1
63204	61	2	175	72.0	130	80	1	1	0	0	1	0
63205	50	1	168	75.0	120	80	1	1	0	0	1	0
63206	58	2	182	100.0	120	80	1	1	0	0	1	1

Figure: 9 Train Test Split Code

V. MACHINE LEARNING MODELS

After importing the data, removing blank entries, normalizing the value based parameter and applying the train test split we are now ready to check different machine learning models on our dataset. So, let's train few machine learning algorithm and check out their training and testing accuracy.

A. Logistic Regression Model

- Logistic Regression predicts the possible outcomes in binary (0-1), that means it basically predicts the probability of the event occurring. To get an idea refer to figure 10.

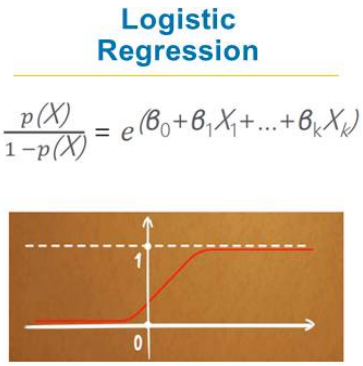


Figure: 10 Logistic Regression Formula

- Figure 11 below shows the algorithm for implementing the logistic regression model on our dataset.

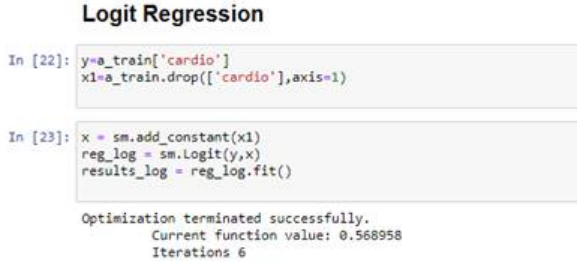


Figure: 11 Implementing Logistic Regression Model

- Now once the model is trained now we have to calculate the training and testing accuracy with the confusion matrix (A 2*2 matrix which shows how many times the model predicted 0 & 1 when the result actually was 0 and same goes for results which are actually 1). You can see the algorithm for confusion matrix and training accuracy in figure 12.

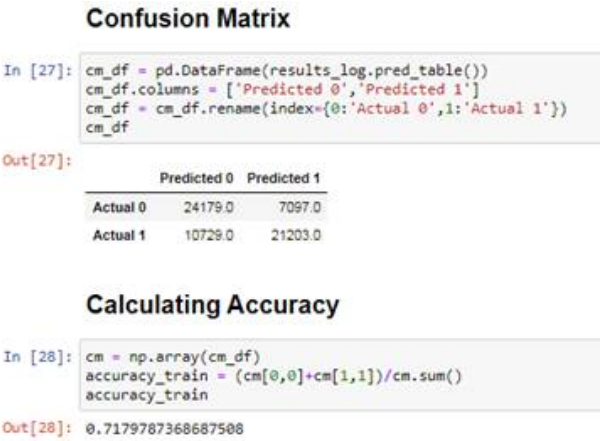


Figure: 12 Logistic Regression Training Accuracy Code

- Now we have to do the same for testing data set.

```
In [32]: cm_df = pd.DataFrame(cm[0])
         cm_df.columns = ['Predicted 0','Predicted 1']
         cm_df = cm_df.rename(index={0: 'Actual 0',1:'Actual 1'})
         cm_df
```

Out[32]:

	Predicted 0	Predicted 1
Actual 0	7.0	2.0
Actual 1	3.0	8.0

Accuracy for Test Data

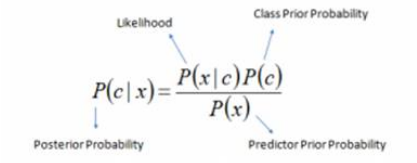
```
In [33]: cm = np.array(cm_df)
         accuracy_test = (cm[0,0]+cm[1,1])/cm.sum()
         accuracy_test
Out[33]: 0.75
```

Figure: 13 Logistic Regression Testing Accuracy Code

- So, the training accuracy was 71.79% and testing accuracy was 75%.

B. Naive Bayes Classifier Model

- The model is based on Bayes Theorem by assuming that all the predicting parameters are independent as you can see in figure 14.



- P(c|x) is probability of class(c , target) given predictor(x , attributes).
- P(c) is probability of class.
- P(x|c) is probability of predictor given class.
- P(x) is the prior probability of predictor.

Figure: 14 Bayes Theorem Formula

- *Working:* First the dataset is converted into frequency table and a likelihood table is created by finding the probabilities. Then Naïve Bayesian equation is used to calculate posterior probability for each class and the class with highest probability becomes the prediction outcome.
- Figure 15 below shows the algorithm for implementing the Naïve Bayes Classifier model on our dataset.

Naive Bayes Classifier Model

```
In [26]: y=a_train['cardio']
         x1=a_train.drop(['cardio'],axis=1)
         x_test=a_test.drop(['cardio'],axis=1)
         y_test=a_test['cardio']

In [28]: gnb = GaussianNB()
         nb=gnb.fit(x1,y)
```

Figure: 15 Implementing Naïve Bayes Classifier Model

- Now, we have to find the training and testing accuracy the same way we did in logistic regression, once the model is implemented. And we found the training accuracy to be 70.62% and testing accuracy to be 66%.

C. Decision Tree Model

- The model is basically a supervised machine learning model where the outcome is continuously according to a certain parameter.
- There are three parameters of decision tree: **Nodes** (Testing for certain attribute's value)**, Edges** (Shows test outcome and connects the next node)**, Leaf Nodes** (Nodes that shows the final outcome). To get a better idea of a decision tree refer to figure 16.

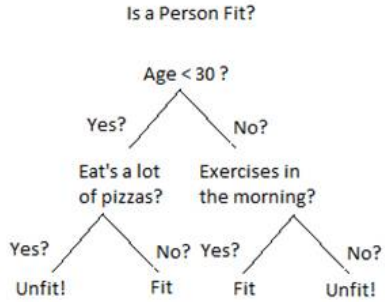


Figure: 16 Decision Tree

- Figure 17 below shows the algorithm for implementing the Decision Tree model on our dataset.

Decision Tree Classifier Model

```
In [84]: y=a_train['cardio']
         x1=a_train.drop(['cardio'],axis=1)
         x_test=a_test.drop(['cardio'],axis=1)
         y_test=a_test['cardio']

In [85]: clf = tree.DecisionTreeClassifier()
         clf = clf.fit(x1,y)
```

Figure: 17 Implementing Decision Tree Classifier Model

- The training accuracy for this model was 97.08% and testing accuracy was 62%.

D. Support Vector Machine (SVM)

- It is a supervised machine learning model which can be used for both problems that is regression and classification. For our application we need classification.
- The main objective of the model is to find an optimal hyper plane in an N-dimension space (N: number of features) that properly and distinctly classifies the result of our target variable.
- When there are two classes of features, there are many probable hyper-planes that divide both of them and what we want is maximizing the margin distance that is distance between data points of both target variable which provides more confidence while classifying future data points. To understand in detail refer to figure 18.

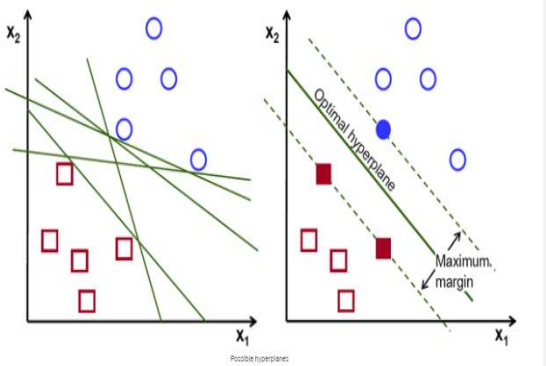


Figure: 18 Possible and Optimal Hyper-Plane

- Figure 19 below shows the algorithm for implementing the SVM model on our dataset.

Support Vector Machine (SVM) Model

```
In [27]: y=a_train['cardio']
         x1=a_train.drop(['cardio'],axis=1)
         x_test=a_test.drop(['cardio'],axis=1)
         y_test=a_test['cardio']

In [28]: svm=LinearSVC(C=0.0001)
         svm.fit(x1, y)
Out[28]: LinearSVC(C=0.0001)
```

Figure: 19 Implementing Support Vector Machine Model

- Setting the regularization parameter C at 0.0001 helps us increase the quality of prediction and decrease over-fitting
- The training accuracy for this model was 70.48% and testing accuracy was 68%.

E. Random Forest Classification Model

- It basically consists of a huge number of individual decision tress that operates together.
- Each tree in random forest gives out a class prediction and prediction with most votes becomes the result of our model's prediction.
- It basically uses bagging and randomness feature which tries to create a forest of trees that are completely uncorrelated which increases the accuracy of our model prediction.
- The most important key to accurate results is low correlation between models.

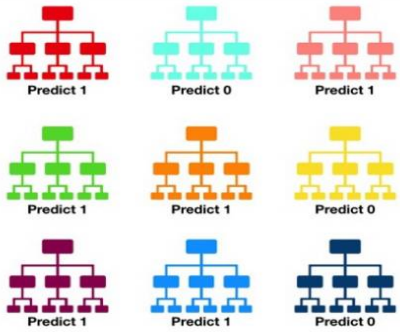


Figure: 20 Random Forest Model

- Figure 21 shows the algorithm for implementing the Random Forest model on our dataset.
-

Random Forest Regression

```
In [22]: y=a_train['cardio']
         x1=a_train.drop(['cardio'],axis=1)
         x_test=a_test.drop(['cardio'],axis=1)
         y_test=a_test['cardio']

In [23]: from sklearn.ensemble import RandomForestRegressor

         regressor=RandomForestRegressor(n_estimators=20,random_state=0)
         regressor.fit(x1,y)
Out[23]: RandomForestRegressor(n_estimators=20, random_state=0)
```

Figure: 21 Implementing Random Forest Model

- The training accuracy for this model was 96.48% and testing accuracy was 80%.

VI. CONCLUSION

A. Accuracy Comparison

Sr No:	Prediction Model	Training Accuracy	Testing Accuracy
1	Logistic Regression	71.79%	75%
2	Naive Bayesian Classifier	70.62%	66%
3	Decision Classifier Tree	97.08%	62%
4	Support Vector Machine	70.48%	68%
5	Random Forest Classification	96.48%	80%

Table: 1 Accuracy Comparison

B. Our Insights

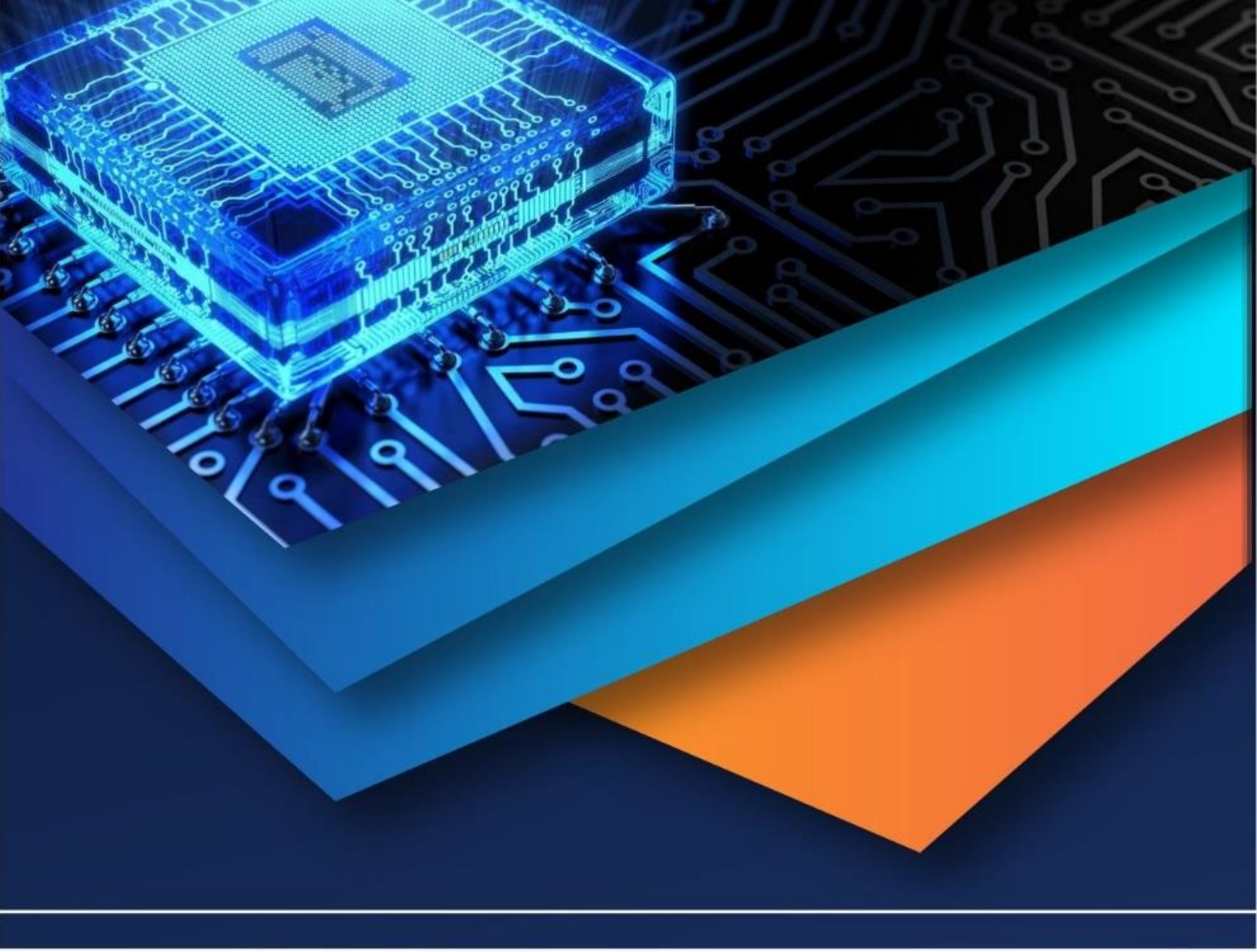
- As we can see from the table 1, even though over-fitting is slightly evident in Random Forest Classification Model, but it provides the best accuracy in comparison to all other models for this particular dataset.
- The highest over-fitting is noticed in Decision Tree Model and the lowest is seen in Logistic Regression, Naïve Bayesian Classifier and Support Vector Machine.
- Finally for our scenario of predicting heart disease possibility the most suitable model is Random Forest.

VII. ACKNOWLEDGEMENT

We would like to express gratitude to Prof. Sneh Soni (Assistant Professor, Department of Electronics & Instrumentation Engineering, Institute of Technology, Nirma University, Ahmedabad) for being our guide for this research project and helping us throughout our research.

REFERENCES

[1] Top 10 Binary Classification Algorithm | By Alex Ortner | Published on May 28, 2020 on Medium.
[2] https://www.youtube.com/watch?v=Y6RRHw9uN9o
[3] https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset
[4] The Data Science Course 2022: Complete Data Science Bootcamp | 365 Careers | Udemy
[5] Maryam Ajanabi, Mahmoud H. Qutqut and Mohammad Hijjawi, "Machine Learning Classification Techniques for Heart Disease Prediction: A Review" in International Journal of Engineering and Technology, October 2018,



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY