



# Performance Analysis Between Combinations of Optimization Algorithms and Activation Functions used in Multi-Layer Perceptron Neural Networks

Akhil Santosh Tirupathi  
Geetha Charan Valmiki

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Science. The thesis is equivalent to 10 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

**Contact Information:**

Author(s):

Akhil Santosh Tirupathi

E-mail: [akti19@student.bth.se](mailto:akti19@student.bth.se)

Geetha Charan Valmiki

E-mail: [geva19@student.bth.se](mailto:geva19@student.bth.se)

University advisor:

Suejb Memeti, Assistant Professor

Department of Computer Science

Faculty of Computing  
Blekinge Institute of Technology  
SE-371 79 Karlskrona, Sweden

Internet : [www.bth.se](http://www.bth.se)  
Phone : +46 455 38 50 00  
Fax : +46 455 38 50 57

---

# Abstract

**Background:-** Artificial Neural networks are motivated from biological nervous system and can be used for classification and forecasting the data. Each neural node contains activation function could be used for solving non-linear problems and optimization function to minimize the loss and give more accurate results. Neural networks are bustling in the field of machine learning, which inspired this study to analyse the performance variation based on the use of different combinations of the activation functions and optimization algorithms in terms of accuracy results and metrics recall and impact of data-set features on the performance of the neural networks.

**Objectives:-** This study deals with an experiment to analyse the performance of the combinations are performing well and giving more results and to see impact of the feature segregation from data-set on the neural networks model performance.

**Methods:-** The process involve the gathering of the data-sets, activation functions and optimization algorithm. Execute the network model using 7X5 different combinations of activation functions and optimization algorithm and analyse the performance of the neural networks. These models are tested upon the same data-set with some of the discarded features to know the effect on the performance of the neural networks.

**Results:-** All the metrics for evaluating the neural networks presented in separate table and graphs are used to show growth and fall down of the activation function when associating with different optimization function. Impact of the individual feature on the performance of the neural network is also represented.

**Conclusions:-** Out of 35 combinations, combinations made from optimizations algorithms *Adam, RMSprop and Adagrad* and activation functions *ReLU, Softplus, Tanh Sigmoid and Hard\_Sigmoid* are selected based on the performance evaluation and data has impact on the performance of the combinations of the algorithms and activation functions which is also evaluated based on the experimentation. Individual features have their corresponding effect on the neural network.

**Keywords:** Activation functions, Neural networks, Optimization algorithms, Performance analysis

---

## Acknowledgments

We would like to express our sincere thanks to our supervisor at BTH, Suejb Memeti for his valuable guidance through out the project. We would also like to express gratitude to our family and friends for their support in completion of the thesis.

**Authors:**

Akhil Santosh Tirupathi

Geetha Charan Valmiki

---

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgments</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Aims and objectives . . . . .	2
1.2 Research questions . . . . .	2
1.3 Overview . . . . .	3
<b>2 Background</b>	<b>4</b>
2.1 Machine Learning . . . . .	4
2.2 Supervised Learning . . . . .	4
2.3 Unsupervised learning . . . . .	4
2.4 Neural networks . . . . .	5
2.4.1 Activation functions . . . . .	5
2.4.2 Optimization Algorithms . . . . .	6
2.4.3 Independent and Dependent Variables in Machine learning . .	7
2.4.4 Multi-layer Perceptron . . . . .	8
<b>3 Related Work</b>	<b>9</b>
<b>4 Method</b>	<b>11</b>
4.1 Working Environment . . . . .	11
4.2 Data collection and Manipulation . . . . .	12
4.3 Feature Selection . . . . .	12
4.4 Training and Testing Of Model . . . . .	14
4.5 Construction of Results . . . . .	15
<b>5 Results</b>	<b>16</b>
5.1 Experiment-1: Model trained with all features . . . . .	16
5.2 Experiment-2: Model trained without least ranked features . . . . .	24
5.3 Experiment-3: Impact of each feature of the neural networks . . . . .	25
<b>6 Analysis and Discussion</b>	<b>27</b>
<b>7 Conclusions and Future Work</b>	<b>31</b>
<b>References</b>	<b>33</b>



---

## List of Figures

2.1	Multi-layer perceptron with hidden layers . . . . .	8
4.1	CorrelationAttributeEval with Ranker search method . . . . .	13
4.2	InfoGainAttributeEval with Ranker search method . . . . .	14
5.1	Loss for model trained with less instances(303) data-set(Adadelata) . .	17
5.2	Loss for model trained with more instances(70000) data-set(Adadelata)	18
5.3	Loss for model trained with less instances(303) data-set(Adagrad) . .	18
5.4	Loss for model trained with more instances(70000) data-set(Adagrad)	19
5.5	Loss for model trained with less instances(303) data-set(Adam) . . .	19
5.6	Loss for model trained with more instances(70000) data-set(Adam) .	20
5.7	Loss for model trained with less instances(303) data-set(SGD) . . . .	20
5.8	Loss for model trained with more instances(70000) data-set(SGD) . .	21
5.9	Loss for model trained with less instances(303) data-set(RMSprop) .	21
5.10	Loss for model trained with more instances(70000) data-set(RMSprop)	22
5.11	Accuracy comparison based on the data-set . . . . .	24
5.12	Accuracy comparison based on the modification of the data-set . . . .	25

---

## List of Tables

5.1	Ranked feature with InfoGainAttributeEval method . . . . .	16
5.2	Accuracy for model trained with 300 instances . . . . .	17
5.3	Accuracy for model trained with 70000 instances. . . . .	17
5.4	Loss for the model trained with 300 instances . . . . .	22
5.5	Loss for the model trained with 70000 instances . . . . .	22
5.6	Precision for the model trained with 300 instances . . . . .	23
5.7	Precision for the model trained with 70000 instances . . . . .	23
5.8	Recall for the model trained with 300 instances . . . . .	23
5.9	Recall for the model trained with 70000 instances . . . . .	23
5.10	Accuracy for the combination of algorithms without 3 functionalities .	24
5.11	Loss for the combination of algorithms without 3 functionalities . . .	24
5.12	Precision for the combination of algorithms without 3 functionalities .	25
5.13	Recall for the combination of algorithms without 3 functionalities . .	25
5.14	Accuracy of Combination of algorithms without least ranked features	26

Emergence of machine learning has large impact on the fields of science, business and medical sciences and engineering. The systems with machine learning algorithms have become smarter in predictions and utilizing these has improved the quality and efficiency of the service. It has already become a part of our life and it's shaping drastically and simplify the lives in the coming future[27]. This makes the computer learns without actually being programmed to perform a task. Huge spike in generation of data,computational processing which is cheaper and powerful and affordable data storage have made it possible to quickly and automatically produce precise models that can analyse complex problems and deliver faster results. All the machine learning algorithms require large amount data to train the model.Data has been major requirement in the machine learning field. We can state that as more data more experience[3].

Despite decades of research , we still don't have the algorithms for some of the tasks we as human beings can do and do effortlessly unaware of how we are actually doing it. We can recognize and make decisions. In machine learning the idea is to do such type tasks, which could be possible with Neural networks[16]. Neural networks has gained fresh momentum with increase in computational performance. Neural networks are inspired from biological nervous system where each neuron perform small tasks to make a decision. Similarly artificial neural networks has nodes and simulating elements that perform small calculation to accomplish a task[15]. Neural networks can be used to classify and forecast the data. The neural networks is made of several node from input layer to output layer and combination of activation functions and optimization functions.

Optimiser are algorithms or methods used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses. Activation functions are used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc[30].

The goal of this thesis is to investigate and tune the performance of the neural networks based on using the different combinations of activation functions and optimization algorithms and also the effect of data-set on the neural network. A feed forward neural network with back propagation with combinations of optimization algorithms and is analysed in this study. Over five optimization algorithms and seven activation functions have been selected. In this study, the appropriate selection of

the combination from the set of optimization algorithms and activation functions have been identified. Based on the performance of the neural networks, the best possible combinations are picked out.

Other part of the study is the effect of the data-set on the performance of the neural network is analysed. The performance of the combinations used in the neural network is measured which is dependent on the data-set to produce more accurate values. The analysis is based on how the combinations are getting affected with the use of different features of the data-set. One of the task is to segregate the features from the data-set and train and evaluated on the same data-set to obtain the metrics to make conclusions. Overall this study shows how the combination is affecting the performance and also the it's selection impacted with the data-set.

## 1.1 Aims and objectives

The aims of the thesis are:

1. To tune the Feed-forward neural network with the possible combinations of the activation function and optimization algorithms to increase the accuracy of the results.
2. To analyse the impact of the data-set on the performance of the combination of activation function and optimization algorithms used in the neural networks.

To achieve the specified aims, the objectives can be defined as follows:

1. Collect the performance variation data of a given neural network by using different combinations of the activation functions and optimization algorithms.
2. Analyse the performance variations of the different combinations used in the neural networks.
3. Identify the factors of the data-set that affect the performance of the neural network.

## 1.2 Research questions

To be able to reach the goal, this project will focus on the following research questions:

1. How the different combinations of optimization algorithm and activation functions affect the performance of the neural network?
2. How does a data-set impact in choosing the best combination?
3. How does the missing data in the instances affect the outcome of neural network?

## 1.3 Overview

Other part of the thesis is structured as follows. First the background contains the explanation of the concepts used for the analysis. By this we can convene the requirements of the neural networks such as optimization algorithms, activation functions and data-set. Next the related works contains the explored content and discussion of the previous works on the neural networks and it's inner mechanism done by the researchers. Next the related works contains the previous researches. Using this we explain how the research gap is found. Next the methods comprising the processes involved starting from gathering to execution of the neural networks. The results obtained from execution of the model are displayed and analysed based on the evaluation of the obtained metrics. The analysed part is discussed with an intention to justify the theme of the project. The final part of the thesis is conclusions made based on the results obtained and discuss few ideas that can be used for future improvements over the project.

## 2.1 Machine Learning

Machine learning is an area of artificial intelligence used to train the system to learn to solve the task without any program or rules to guide the machine. Machine learning is programming a system to optimize the performance criterion with the help of preset examples or existing output data. The machine learning uses theory of statistics and mathematical model because the core task is to make inferences from a sample data set. There are four types of machine learning algorithms namely supervised, unsupervised, semi-supervised and reinforcement learning. In this study we have used supervised learning and unsupervised learning algorithms.[16].

## 2.2 Supervised Learning

Supervised learning is the most common branch in machine learning. In supervised machine learning the machine is trained by giving previous outputs as present inputs. In training phase the algorithms gain the information or identify the pattern from the data-set. After the training phase completes the algorithm is fed with new data and gives the output based on the patterns learned during the training phase. The main objective of supervised learning is to predict the classification correctly using its training knowledge gained by analysing the labeled data. There are several approaches and algorithms used in supervised learning, few of them are Analytical learning, Artificial neural networks, back propagation, Bayesian statistics etc [17].

## 2.3 Unsupervised learning

Unsupervised machine learning looks for previously undetected patterns in the given data-set without depending on human given labels. This algorithm is trained with the data-set without the labels. Unlike supervised learning there is minimum human supervision in Unsupervised learning. The two main learning principles in unsupervised learning's are the component analysis and the cluster analysis. The best time to use unsupervised machine learning is when you have no prior data-set to train the machine [17].

## 2.4 Neural networks

A Neural network is a information processing system implemented on the machines with complex software and hardware to implement learning and memorizing factors of a human brain. Basically Neural network is a network of neurons. Biological neural network is made up of real biological neurons where as the artificial neural network with software and hardware components for solving problems related to artificial intelligence(AI). A neural network in case of artificial neurons is called artificial neural network(ANN). the artificial neural network can be applied to real world tasks. We consider ANN as simplified model of the biological neural network. Neural network comprises of several activation function and optimization algorithms which in combination can produce the outcome of a prediction. We are using a combination of 7 activation functions and 5 optimization algorithms [29][26].

### 2.4.1 Activation functions

In neural networks every neuron contains 'n' no of inputs and a single output. The output result will be based on the activation function. The activation function calculates the weighted sum of all inputs and check the values produces by the neurons ultimately deciding weather outside connections should consider the neuron to activate or not [32][30][31].

#### 1. Linear

When the activation is proportional to the input is a linear function. It gives the range of activation's in a line hence the output of the function will be not in any range given its range can be from minus infinity to infinity which does not help in complex parameters that is given to the neural networks [31].

$$f(x) = CX$$

#### 2. ReLU

The ReLU is the most popular and used function right now because it is used in all neural networks and deep learning the range of the ReLU is from 0 to infinity and the function is monotonic. In this function all the negative values will become zero which is a drawback to this function [30].

$$f(x) = \frac{1}{1 + e^{-x}}$$

#### 3. TanH

TanH is a better version of sigmoid the range of TanH is from minus one to one and shape of the TanH is sigmoidal. the advantage of the TanH is the zero

will be nearly mapped as zero and negative inputs will be strongly mapped as negative inputs [31].

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

#### 4. Sigmoid

The curve shape of the sigmoid function is in S shape. The range of the function is between 0 to 1. This function is mainly used to predict the probability as an output since the probability of any output ranges between 0 to 1 [31].

$$f(x) = \frac{1}{1 + e^{-x}}$$

#### 5. Softplus

The softplus is similar to ReLU the threshold of this network is at 0 network trained with this function can find local minima of the greater or equal quality than those obtained with smooth counterpart, the softplus [30].

$$f(x) = \ln(1 + e^x)$$

#### 6. Hard Sigmoid

The main difference between sigmoid and hard sigmoid is its computing speed, The computing speed of hard sigmoid is faster than that of sigmoid function the range is 0 to infinite. this function is faster than sigmoid because it does not need to calculate the exponent and it provides reasonable results on classification tasks [30].

$$f(x) = \frac{1}{1 + e^{-x}}$$

#### 7. Exponential

This function is an upgraded version of ReLU but it is less widely used. It leads to higher classification results than traditional ReLU, this function controls the negative scale by default setting it to zero[30].

$$f(x) = \alpha e^x - 1$$

### 2.4.2 Optimization Algorithms

In neural network optimization algorithms are used to reduce the randomness and lacking of predictable order in neural network which is the main goal of the optimizer. Optimizers are also used to change attributes of the neural network to reduce the losses

with the reduction of the losses with the help of optimization algorithm the accuracy of the result increases. The optimization algorithms used in the experimentation are the following [28].

### 1. Adagrad

Adagrad is an algorithm for gradient-based optimization which performs small updates to the system to increase the learning rate to the parameters [18].

### 2. Adadelta

Adadelta is an extension of the adagrad which reduces its learning rate instead of acquiring all squared gradients adadelta restricts the window of accumulation of past gradient [28].

### 3. Adam

Adaptive moment estimation also known as adam is a method that performs adaptive learning rates for each parameters. It also store exponential decay average to past squared gradients like adadelta and RMSprop [18].

### 4. RMSprop

RMSprop is developed along with adadelta to resolve the problem in adagrad's radical reduction in learning rates. RMSprop is well at dividing learning rates by exponentially decaying squared gradient [28].

### 5. SGD

Stochastic gradient descent (SGD) performs by updating each training example and label. The SGD performs redundant computerization of large databases as it recomputes gradient before each parameter update [18].

## 2.4.3 Independent and Dependent Variables in Machine learning

Independent and dependent variables are two important terms in machine learning. based on these variables the machine learning techniques such as linear regression, logical regression, cluster analysis, principle analysis gets decided. Both independent and dependent variables effect each other means when one variable is changed other

also will be change. The independent variable is determined by dependent variable. In independent variable the variable can control over its its choice and manipulation which affects dependent variable also. A dependent variable is what measured in experiment and what is effected in experiment. It is called dependent because it depends on independent variable. In an experiment dependent variable cannot exist without independent variable [20][16].

#### 2.4.4 Multi-layer Perceptron

The Multi-layer perceptron is simple inter-connection of Neuron system or nodes. The nodes are connected with by weights and output signals, Which are function to the sum of inputs to the node modified by an activation function. If the transfer(activation) function was linear then the multi-layer perceptron would only be able to model linear functions. The output of a node is scaled by the connecting weight and fed forward to be an input to the nodes in the next layer of the network. The architecture of a multi-layer perceptron is variable but in general will consist of several layers of neurons. The input layer plays no computational role but merely serves to pass the input vector to the network[24].

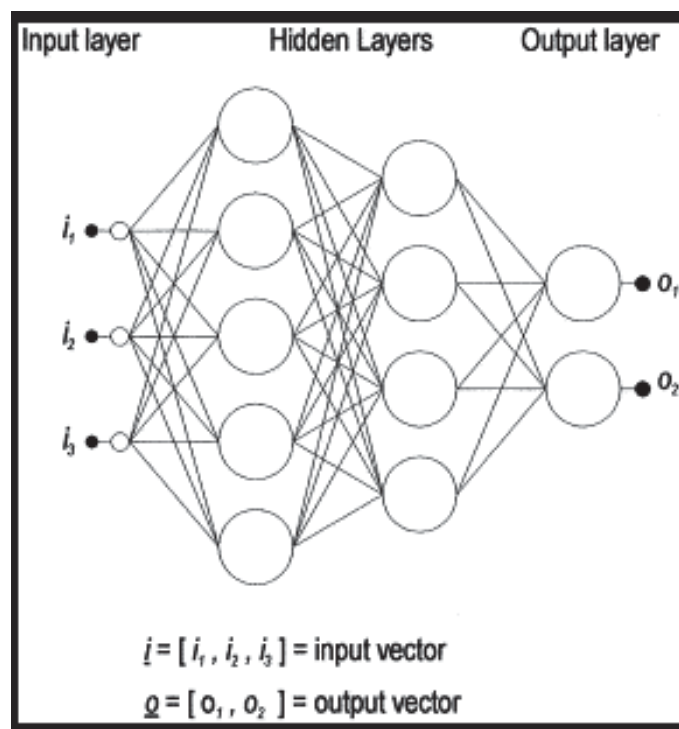


Figure 2.1: Multi-layer perceptron with hidden layers

P.SIBI, S.ALLWYN JONES and P.SIDDARTH has done analysis on different propagation activation functions using back propagation on neural networks. Their purpose is to identify the optimal optimization algorithm for a problem. The activation functions are used transform the activation level of every neuron into an output. They have done analysis on different activation functions and benchmarks them to figure out the optimal function for the specific problem[31].

SEBASTIAN RUDER has done an overview on gradient optimization algorithms which are used as black/box optimizer. the author has provided information on behavior of different optimization algorithm which will be used according to the problem. the article has overviews various algorithms of gradient decent summarize challenges and review architectures in parallel and distributed setting[28].

PARNIA BAHAR, TAMER ALKHOULI AND JAN-THORSTEN PETER has done an Empirical investigation of optimization algorithms in neural machine translation in their article they have trained the neural networks in high dimensional optimization problems. the article gives the most popular optimization techniques used to train neural networks. evaluation of these optimization algorithms in done by the convergence speed, translation quality and training stability with these tests the optimal algorithm is selected according to the give problem[18].

Tulay Karayilan has done prediction of heart disease's using neural networks. In his study he used neural network back-propagation algorithm to predict heart disease. The clinical features were given as the inputs for the neural network. Then the neural networks were trained with back propagation algorithm to predict the heart disease which is 95 percent accurate stated in his study[23].

Robin Forsman and Jimmy Jonsson has done study on best algorithms to apply in machine learning based on diabetic database. The study shows the comparison and analysis of different algorithms to give the best algorithm to use. The data has been acquired by the patients that have been re-admitted or did not re-admitted to the hospital within 30 days after being admitted as a patient. With the data several algorithms are used and has given the best algorithm to use[6].

Bekir Karlik and Vehbi Olgac has done the performance analysis of various Activation functions. The study is about analysis of various MLP architectures which

has back propagation algorithm using various different back-propagation algorithms using various activation functions. The activation functions which are used for the Comparison are sigmoid, Tanh, bi-polar sigmoid, Uni-polar sigmoid. These activation functions are analysed and compared to give the best activation function for the problem at hand[24].

Bhaskar DasGuptha and Georg Schnitger had done the comparison of activation functions. The comparison of the functions is done in term of approximation power of their feed-forward nets. The study considers the case of analog and Boolean input. The experimentation gives the information on which activation function gives the optimal output for different cases[21].

From the above analysed research papers we found that author has either compared only activation functions or the optimization algorithms alone. In this thesis we are analysing the combination of the optimization algorithm and activation function.

To answer the research question we have chosen the experimentation method where 35 different combinations of optimization algorithms and activation functions used in the neural networks are compared based on the variations in the accuracy of the neural networks. The heart disease and cardiac disease data-set is used to train the neural networks to create classifier that could be used for the predictions as well as for our analysis based on the accuracy values. This experimentation process undergo four stages detailed as follows.

1. Gather and preprocess the data to get the consistent and error free data.
2. Select the feature that has more influence on the model using feature selection algorithm.
3. Train the model with data-set and retrieve the results.
4. Represent the results.

The working environment and software used for this analysis are detailed in the next section.

### 4.1 Working Environment

The above mentioned methods and steps are experimented on the laptop with the following specifications. All the items used in this experimentation process are latest versions and detailed with version number and description.

- Windows 10 64-bit Operating System.
- Intel Octa-core processor that runs at 2.90GHz - 2.56GHz.
- 8.00 GB RAM that runs at 2667MHz.
- Python V .3.8.3 - Open source programming language[9].
- Anaconda V .2020.02 -Open Source environment provider, package management and deployment[14].
- numpy V .1.18.1 - library for N-dimensional array processing and scientific tools[4][11].

- pandas V .1.0.1 - Open source library for data analysis and manipulation[5].
- scikit-learn V .0.22.1 - Open source library for predictive data analysis[7][10].
- matplotlib V .3.1.3 - Open Source library for visualizing data[2].
- Tensorboard v .2.0.2 - Open Source Tensorflow's visualization toolkit[8].

## 4.2 Data collection and Manipulation

Two different data-set are considered for this experimentation process. The heart disease data is collected from the Kaggle open source database. This particular data in the Kaggle is collected in the Cleveland database[1]. This data is mostly used by the researcher till date. The original database has around 414 instances with over 76 attributes. Kaggle has fully refined and manipulated data with around 308 instances. The heart disease data has 13 attribute or features which can be called as independent variables and one dependent variable which are explained in section 2. The cardio vascular data-set is also collected from the Kaggle data-set. This data has 13 attributes and contains around 70000 instances. The collected heart disease and cardio vascular data-sets are stored in the comma-separated value file(CSV). We need not do preprocessing where all imprecise data which could have occurred due to human errors has already been cleansed. All the data has gone through the preprocessing stage where the null values or the missing entries are eliminated by completely removing the instance or have gone through imputation techniques which fills the missing values with an estimated value based on the other available information. Redundant instance are also removed so that they won't add extra burden during model execution.

The data is loaded from CSV to data frame using the pandas library functions. This created data frame can be used for the training of the neural network and testing for the precision of the model. So to divide the data frame, a function named train test split is used. This function divides the data frame into two parts, one is the training data holding the 80% of the data frame and remaining 20% is the testing data which is used for evaluation. The data is pushed randomly into two of the sets and contrast to each other.

## 4.3 Feature Selection

This part answers the research question 3. Here the features that has more contribution for determining the target variable is selected. Already 13 attributes are selected from the 76 attributes from the original data-set. We will use weka tool to rank the importance of the different features[22]. With this we can get rid of the features which do not contribute much benefit to the model or will not put the model in confused state to make wrong decision. These attributes or features can be irrelevant to use. So, these attributed could be ranked with the help of feature selection

algorithm, doing so would benefit in reducing over fitting, improving accuracy and reducing training time.

The attribute evaluator technique is applied to identify the evaluating the attribute in the context of the output or target. A search method is also applied with the attribute evaluator which try different combinations of the attribute in order to arrive on a short list of attributes[19]. In this experiment we used two types of evaluators with *Ranker* search method. One with the *CorrelationAttributeEval* and other with the *InfoGainAttributeEval*. The attributes with the lowest rank will be removed from the data-set and this new data-set is used for the training[22]. Analysis of the neural network is performed based on the accuracy results extracted from the training of the neural network.

See figure 4.1 and 4.2, on the X-axis, is the names of the attributes arranges according to their ranked values and the feature importance on the Y-axis. The features are arranged in descending order. Three features with the least rank and the comprising values of the feature columns will be removed from the data-set and used for the training of the neural networks. From the below graphs we can observe that the last four feature are same in the both figures. Both the methods gave similar results but we have considered the *InfoGainAttributeEval* with *Ranker* search method because this has shown the values which are stating the information gained or the strength of relation between the independent attributes and target attribute. So, the three attribute removed from the data-set are *Restecg*, *Chol* and *FBS*.

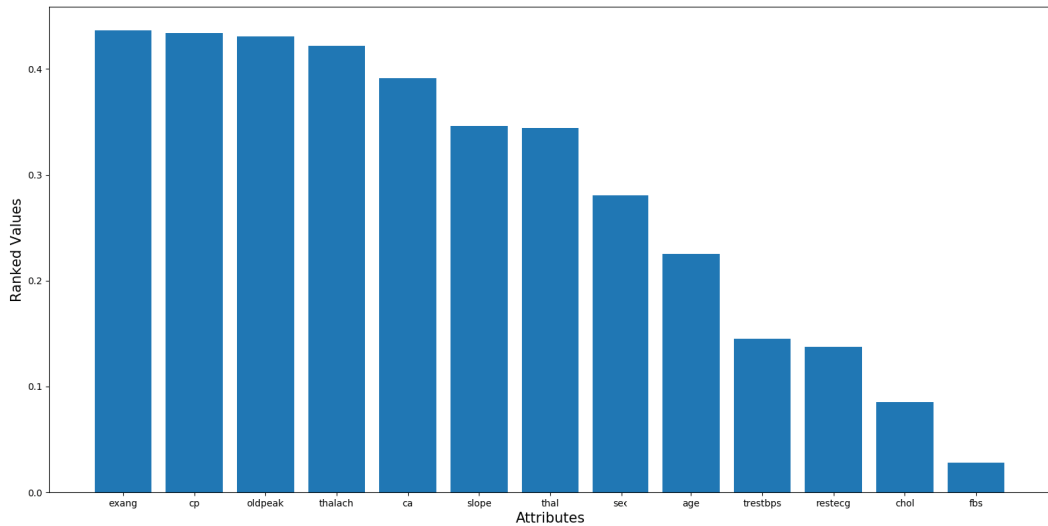


Figure 4.1: *CorrelationAttributeEval* with *Ranker* search method

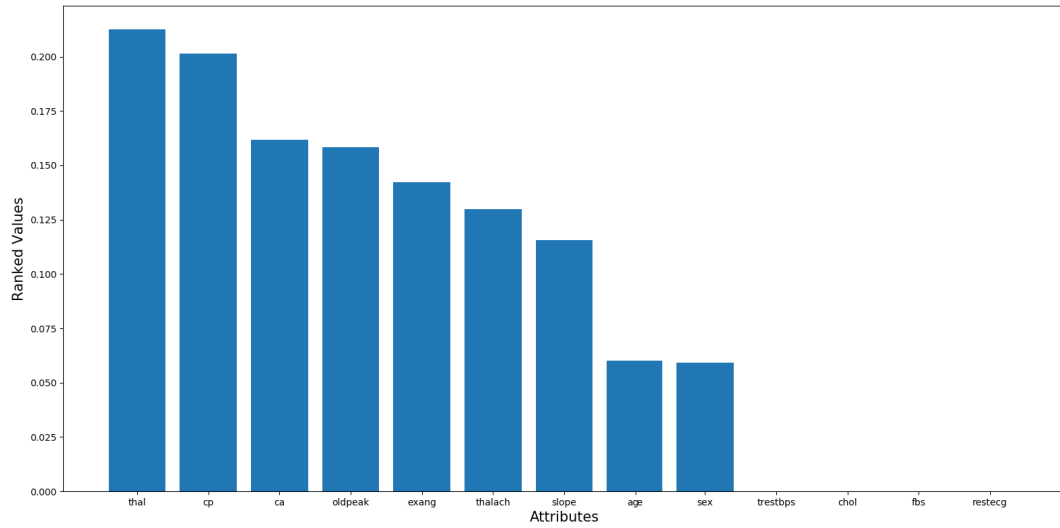


Figure 4.2: InfoGainAttributeEval with Ranker search method

## 4.4 Training and Testing Of Model

Feed forward neural network model with back propagation techniques is implemented with three layers. Since it has back propagation optimization algorithms are used. We used seven activation function and five optimization algorithms. Total of 35 different combinations of the activation functions and optimization algorithms are infused in the neural networks. The neural network has 13 and 8 densely featured input layer for heart disease and cardiac disease data set respectively, 128 hidden layer nodes and output layer with one node. The data-set in CSV format is converted to *tf.data* data-set to be able to feed it to the neural network. Training and evaluation depends on the number of iterations performed. Iteration involves cycle of feed forward and back propagation. Based on the number of iterations performed, we can analyse the performance of each combination of the activation and optimization functions, based on the factors such as execution of the model, accuracy rate, precision and metrics recall. This iteration is called epoch. The model is trained with 100 epoch or iterations and data is fed into the neural in batches of size 32.

In the stage 2, the feature with less ranks are deprecated from the data-set which are selected based on the feature selection algorithms. So, the model is updated with eight input nodes and similar experimentation is done as explained in the previous paragraph. By doing this we can analyse the impact of each feature or the combination of features in the data-set on the performance of the neural network. The impact of the instance on the neural network prediction is analysed by evaluating the neural network by passing the instance with discarded values or null values. The performance metrics are compared with the original instance and conclusion are made.

## 4.5 Construction of Results

After the results are generated from the training and testing of the model, the results are organised in a table. The results are divided into accuracy, recall, precision and loss tables. The graphs representing the loss. The tables are structured as activation function names in the first row and optimization algorithms in first column and the results are arranged as correspondingly. These tables are in CSV format and visualized using the "matplotlib" library to view the performance variance.

In this chapter, the results are presented that are obtained during the research and are analysed in the next chapter. The tables in this chapter presents the performance of the combination of the activation function and optimization algorithm. Next the three features from the data-set are removed and similar performance tables of the combination is presented. The graph compares the performance of the neural network infused with the combination of the optimization algorithm and activation function which are trained with original and modified data-sets. The separate tables are used for presenting the accuracy, loss, precision and recall. The first row is the activation functions and the first column contains the optimization algorithms and the rest of the columns contains performance of the combination. A bar chart of containing the performance of each of the combinations that are obtained by training the neural network.

Table 5.1: Ranked feature with InfoGainAttributeEval method

Rank	feature	Info Gain
1	thal	0.2127
2	cp	0.2013
3	ca	0.1617
4	oldpeak	0.1585
5	exang	0.1422
6	thalach	0.1297
7	slope	0.1157
8	age	0.0602
9	sex	0.0591
10	trestbps	0
11	chol	0
12	fbs	0
13	restecg	0

## 5.1 Experiment-1: Model trained with all features

In Experiment 1 the outputs are obtained by combination of optimization functions and activation functions. There are 7 activation functions and 5 optimization algorithms table from 5.2 to 5.9 presents the accuracy, loss, precision and recall of the combinations. In this experiment all the features are used, present in the table 5.1 to train the neural network.

Table 5.2: Accuracy for model trained with 300 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	0.497041	0.497041	0.437870	0.473373	0.473373	<b>0.562130</b>	0.502959
Adagrad	0.760624	<b>0.783217</b>	0.686391	0.416353	0.733728	<b>0.791824</b>	0.637877
Adam	<b>0.881119</b>	<b>0.874664</b>	<b>0.759548</b>	0.414201	<b>0.795051</b>	<b>0.862282</b>	0.738569
SGD	0.5508338	<b>0.864443</b>	0.665402	0.437870	0.674556	<b>0.729962</b>	0.718666
RMSprop	<b>0.777300</b>	0.748790	<b>0.802044</b>	0.457773	<b>0.758472</b>	<b>0.7842293</b>	<b>0.766361</b>

Table 5.3: Accuracy for model trained with 70000 instances.

	Linear	ReLU	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	0.6784	<b>0.6929</b>	0.6876	0.4986	<b>0.6931</b>	<b>0.6937</b>	0.6805
Adagrad	0.6643	<b>0.7054</b>	0.7029	0.5019	<b>0.7039</b>	<b>0.7064</b>	0.7001
Adam	0.6716	<b>0.714</b>	<b>0.7084</b>	0.4998	<b>0.7015</b>	<b>0.712</b>	<b>0.7016</b>
SGD	0.6402	<b>0.6885</b>	<b>0.6957</b>	0.501	0.6825	<b>0.6899</b>	<b>0.6911</b>
RMSprop	0.6772	<b>0.7134</b>	<b>0.7423</b>	0.5005	0.705	<b>0.7137</b>	<b>0.7099</b>

Graphs for loss generated by the model with Activation functions and Adadelata have been mentioned below.

Figure 5.1: Loss for model trained with less instances(303) data-set(Adadelata)

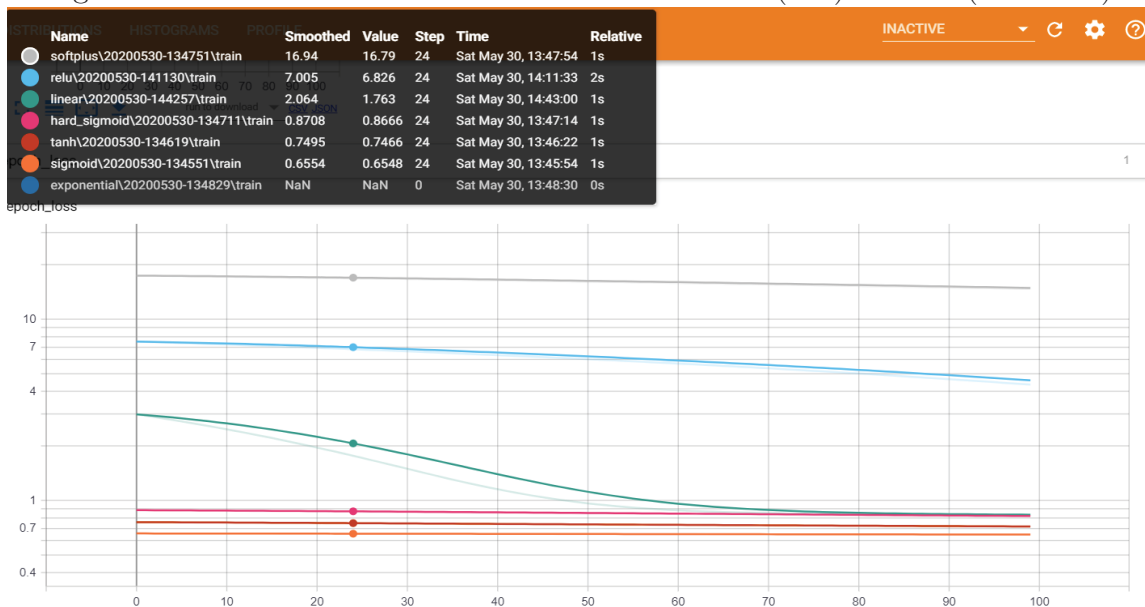


Figure 5.2: Loss for model trained with more instances(70000) data-set(Adadelta)

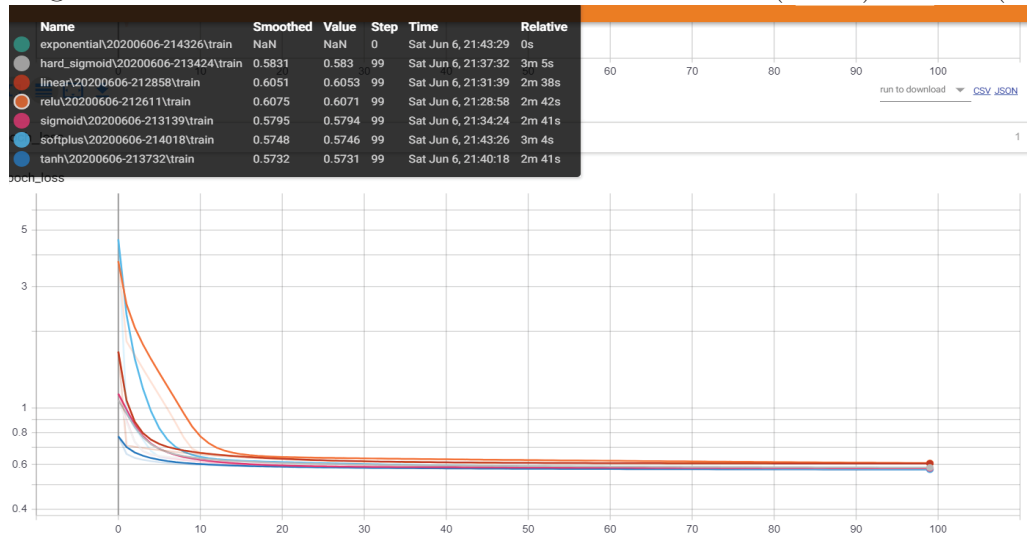


Figure 5.3: Loss for model trained with less instances(303) data-set(Adagrad)

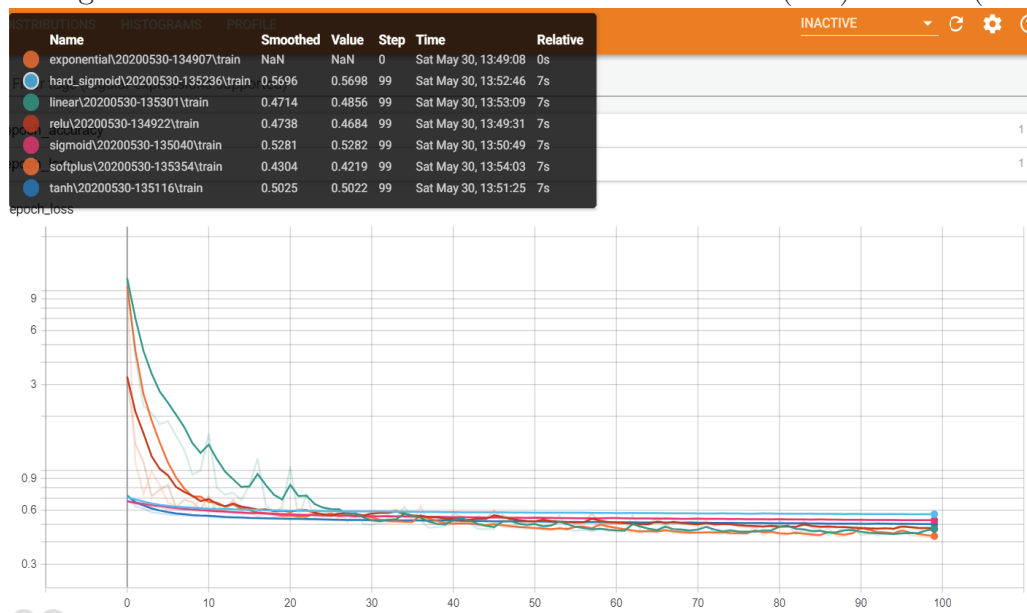


Figure 5.4: Loss for model trained with more instances(70000) data-set(Adagrad)

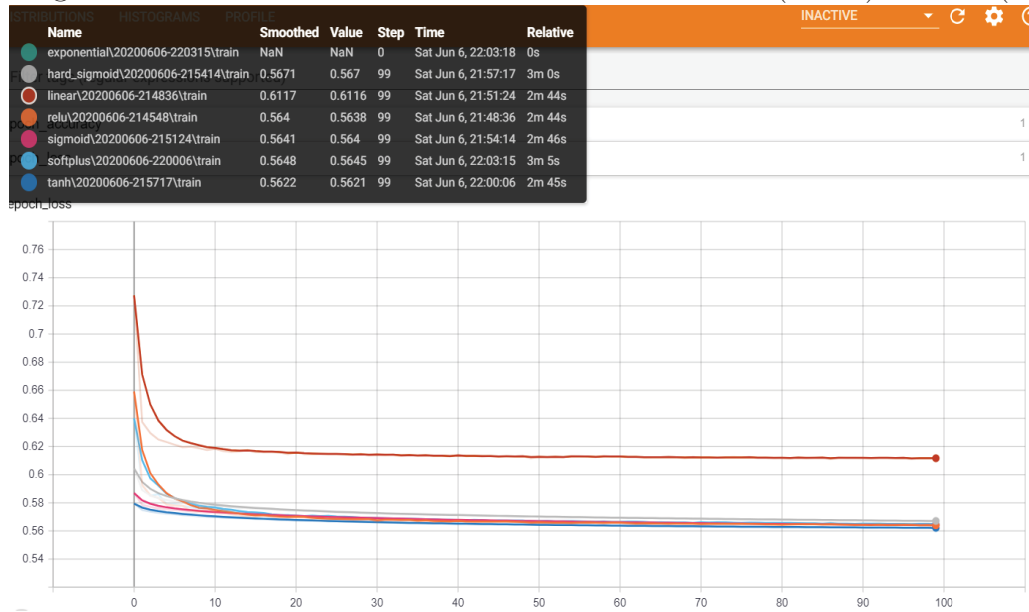


Figure 5.5: Loss for model trained with less instances(303) data-set(Adam)

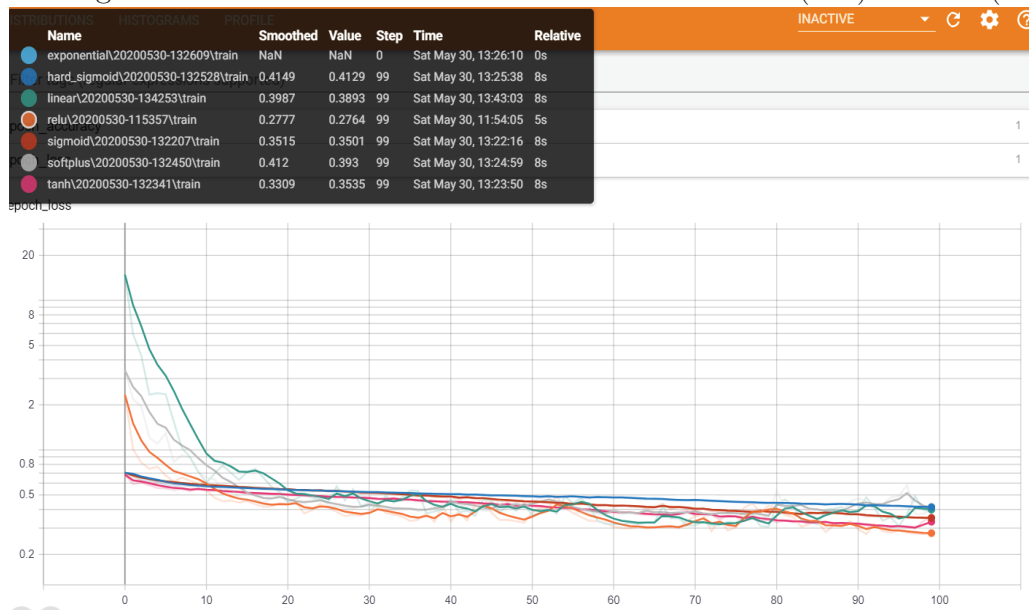


Figure 5.6: Loss for model trained with more instances(70000) data-set(Adam)

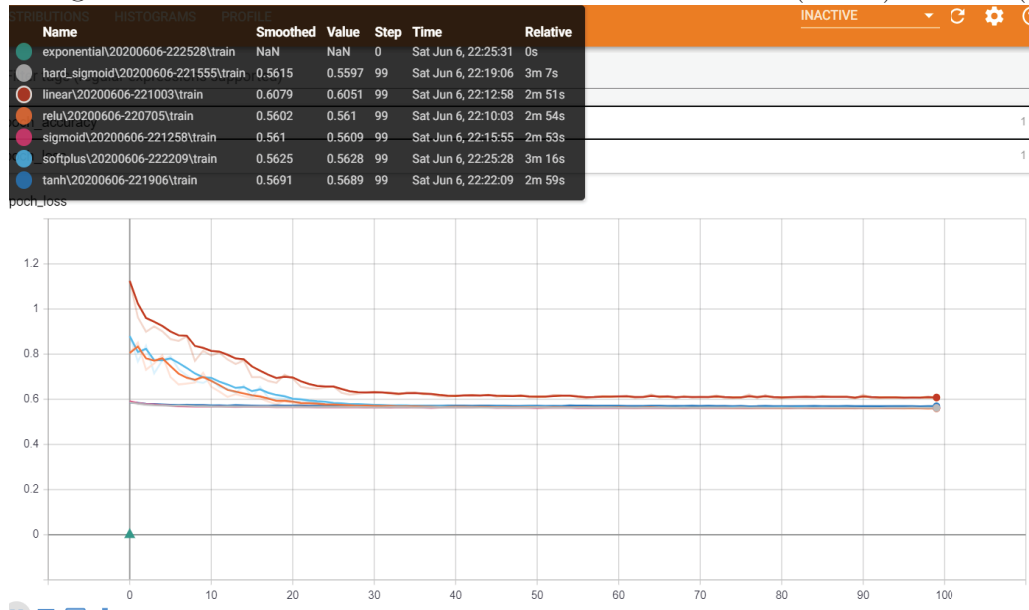


Figure 5.7: Loss for model trained with less instances(303) data-set(SGD)

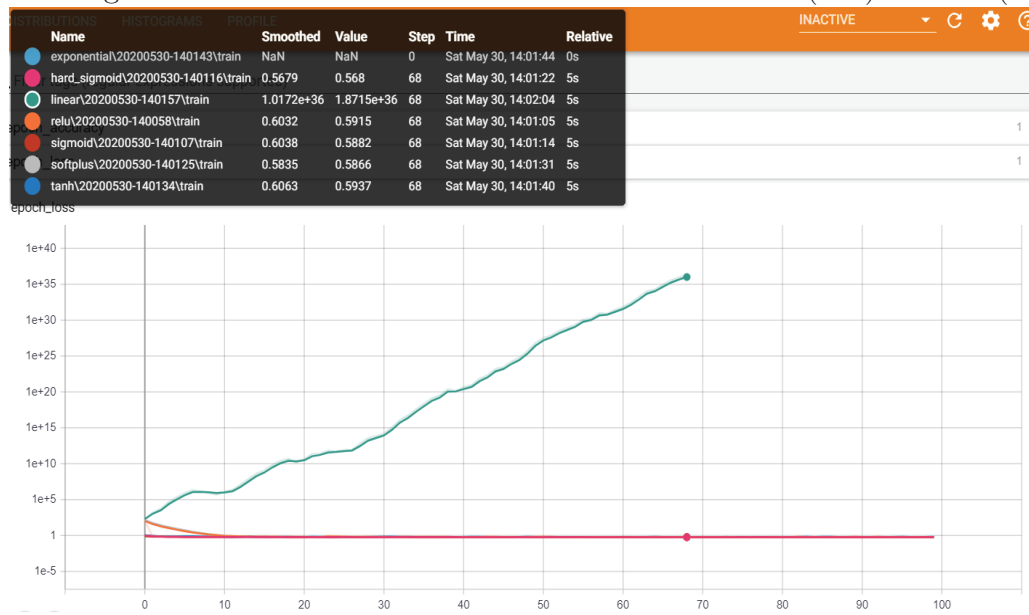


Figure 5.8: Loss for model trained with more instances(70000) data-set(SGD)

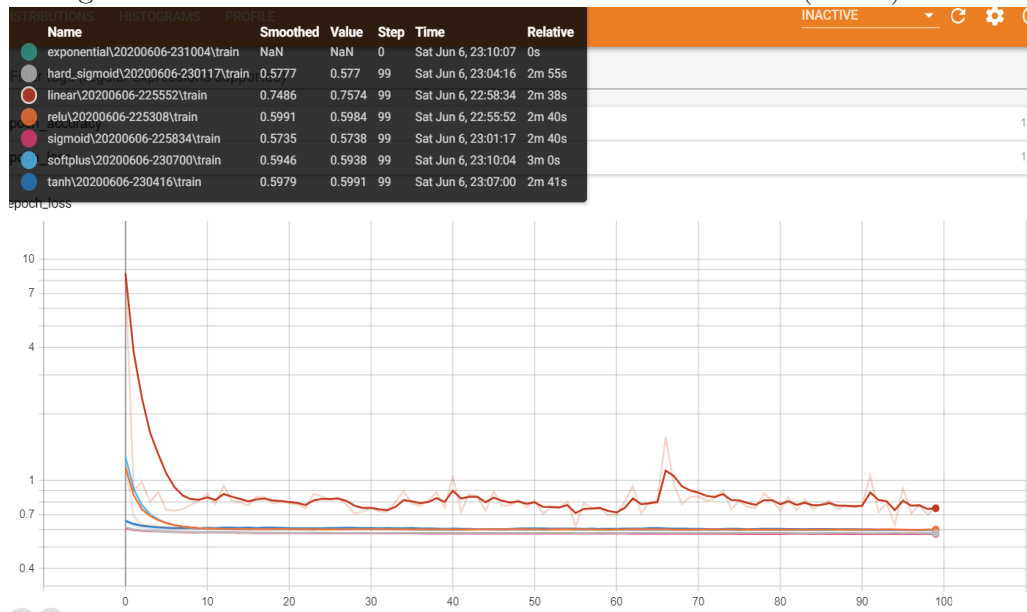


Figure 5.9: Loss for model trained with less instances(303) data-set(RMSprop)

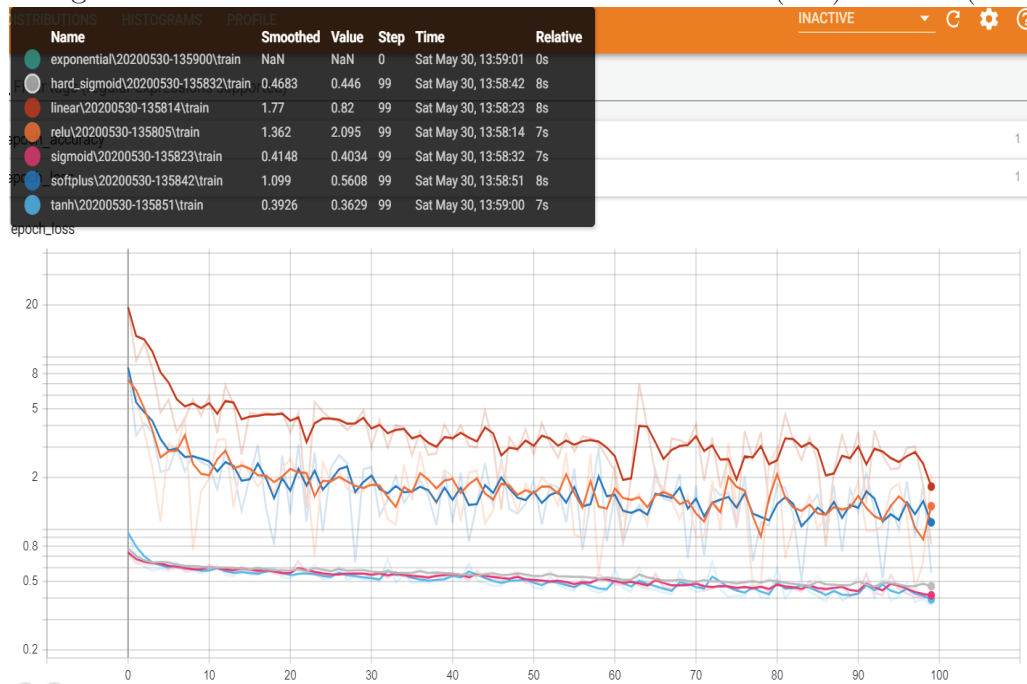
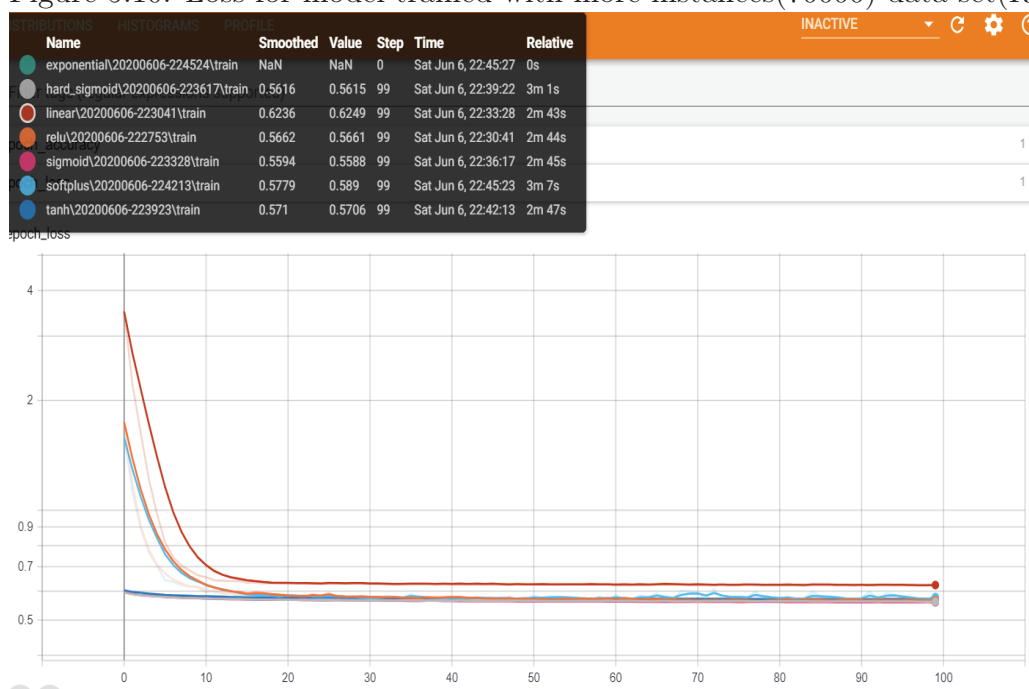


Figure 5.10: Loss for model trained with more instances(70000) data-set(RMSprop)



The loss tables 5.4 and 5.5 are prepared from the above graphs. Two different tables are constructed for the model trained with 300 instances and 70000 instances.

Table 5.4: Loss for the model trained with 300 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	4.323979	29.030635	<b>0.701901</b>	NaN	0.739079	9.973505	<b>0.705915</b>
Adagrad	0.524276	<b>0.507154</b>	0.606358	NaN	0.539681	<b>0.492948</b>	0.618857
Adam	<b>0.320927</b>	<b>0.329954</b>	0.486109	NaN	0.425594	<b>0.339876</b>	0.542919
SGD	NaN	0.352500	0.610486	NaN	0.572615	0.551932	0.569249
RMSprop	0.842227	0.541947	<b>0.446704</b>	NaN	0.483857	0.460927	0.514308

Table 5.5: Loss for the model trained with 70000 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	1.763	6.826	<b>0.6548</b>	NaN	<b>0.7466</b>	16.79	<b>0.6548</b>
Adagrad	0.6053	0.6071	<b>0.5794</b>	NaN	<b>0.5731</b>	<b>0.5746</b>	0.583
Adam	0.6051	<b>0.561</b>	<b>0.5609</b>	NaN	0.5689	<b>0.5628</b>	<b>0.5609</b>
SGD	0.7574	0.5984	<b>0.5738</b>	NaN	0.5991	<b>0.5938</b>	<b>0.577</b>
RMSprop	0.6429	<b>0.5661</b>	<b>0.5588</b>	NaN	<b>0.5706</b>	0.589	<b>0.5615</b>

Table 5.6: Precision for the model trained with 300 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	-5.160310	-5.9341843	-0.064492	0.500001	-0.042970	<b>24.806279</b>	-0.246826
Adagrad	<b>1.993469</b>	0.676687	0.437431	0.257576	0.502698	0.988253	0.438676
Adam	<b>1.956247</b>	<b>1.554202</b>	0.560954	0.378789	0.930456	0.965514	0.525380
SGD	1.6923e+05	0.848156	0.363916	0.272728	0.416539	<b>5.113142</b>	0.427093
RMSprop	<b>4.049005</b>	<b>2.884051</b>	0.902649	0.2422425	0.667206	1.317516	0.763839

Table 5.7: Precision for the model trained with 70000 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	2.905	2.168	2.969	0.2661	9.39	-15.85	2.429
Adagrad	2.68	7.847	22.79	0.2488	18.7	0.5746	25.48
Adam	2.033	0.714	13.29	0.2431	7.095	4.5	8.669
SGD	0.5409	-1.962	9.229	0.2594	10.12	-1.306	6.3672e+7
RMSprop	2.345	7.43	399.1	0.2732	72.1	16.43	6.221

Table 5.8: Recall for the model trained with 300 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	0.533379	0.535461	0.241264	0.666667	<b>2.176625</b>	0.559652	0.399609
Adagrad	<b>1.993469</b>	0.507154	0.768152	0.333333	<b>2.061743</b>	<b>2.428261</b>	0.765825
Adam	1.742845	1.709055	<b>2.092028</b>	0.545455	1.387705	<b>8.449408</b>	0.929412
SGD	<b>3.314767</b>	-0.704208	0.870329	0.348485	<b>1.138032</b>	0.637700	1.084247
RMSprop	1.69231e+	0.935572	<b>1.325643</b>	0.333333	<b>1.287244</b>	<b>1.618044</b>	1.175710

Table 5.9: Recall for the model trained with 70000 instances

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	0.6822	0.5971	0.4458	0.2561	0.4892	0.4892	0.3839
Adagrad	0.5336	0.5928	0.5402	0.2488	0.5695	0.6002	6.0592
Adam	0.7374	0.6639	0.6216	0.2431	0.6129	0.6887	0.6125
SGD	0.04722	0.3317	0.5627	0.2594	0.4474	0.3119	0.5187
RMSprop	6.707	0.6558	0.6133	0.2732	0.6003	0.6489	0.6467

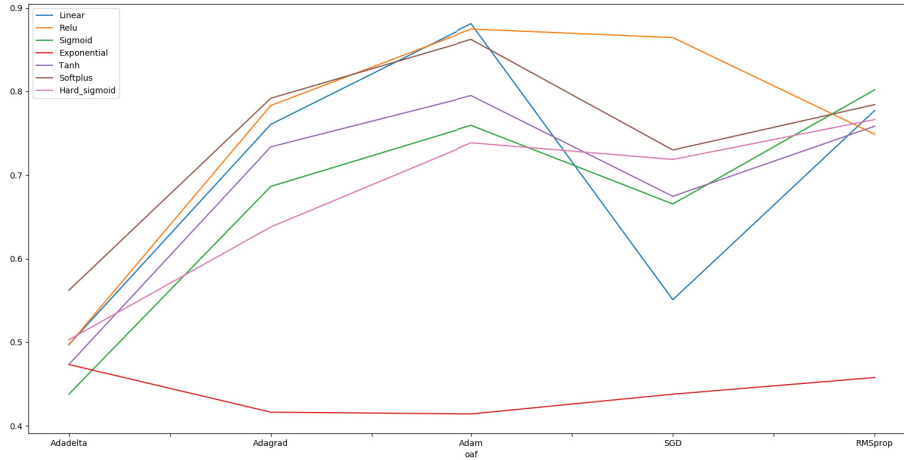


Figure 5.11: Accuracy comparison based on the data-set

## 5.2 Experiment-2: Model trained without least ranked features

In experiment 2, three least ranked features from table 5.1 are removed which are chol, fbs, restecg. The results have been obtained without these features in table 5.6, 5.7, 5.8 and 5.9.

Table 5.10: Accuracy for the combination of algorithms without 3 functionalities

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	0.426035	0.455621	0.35503	<b>0.550296</b>	0.431953	0.449704	<b>0.538462</b>
Adagrad	<b>0.745024</b>	0.671329	0.681011	0.465842	<b>0.746638</b>	0.70199	0.526627
Adam	<b>0.862829</b>	<b>0.82894</b>	0.754169	0.472835	<b>0.778375</b>	<b>0.792899</b>	0.735879
SGD	0.71759	0.7149	0.678322	0.485207	<b>0.746638</b>	<b>0.739107</b>	<b>0.724045</b>
RMSprop	<b>0.798279</b>	<b>0.794513</b>	0.777838	0.514793	<b>0.786444</b>	<b>0.796665</b>	0.743948

Table 5.11: Loss for the combination of algorithms without 3 functionalities

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelata	23.54428	2.89827	0.751857	7.07e+26	<b>0.872663</b>	15.33109	<b>0.914535</b>
Adagrad	<b>0.520074</b>	0.594971	0.597778	0	<b>0.516956</b>	0.58466	0.618799
Adam	<b>0.358377</b>	<b>0.389025</b>	0.471614	0	0.429904	0.444521	0.516095
SGD	0.574019	0.576481	0.614803	0.	<b>0.536301</b>	<b>0.566344</b>	0.58391
RMSprop	0.591658	<b>0.440211</b>	0.478875	0	<b>0.446479</b>	<b>0.406925</b>	0.542974

Table 5.12: Precision for the combination of algorithms without 3 functionalities

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelta	0.562891	<b>0.647218</b>	0.373579	0	<b>0.642952</b>	0.58273	0.551106
Adagrad	<b>1.111153</b>	0.525026	-0.41523	0.242424	<b>1.748801</b>	0.649028	0.374162
Adam	1.446865	<b>6.490123</b>	2.753029	0.363636	<b>5.566366</b>	3.874915	1.166678
SGD	0.559809	0.468253	<b>0.833564</b>	0.30303	0.647061	<b>1.410763</b>	0.899207
RMSprop	1.473218	<b>2.208972</b>	1.879263	0.303031	<b>3.181668</b>	-3.88118	1.772402

Table 5.13: Recall for the combination of algorithms without 3 functionalities

	Linear	Relu	Sigmoid	Exponential	Tanh	Softplus	Hard_sigmoid
Adadelta	-4.27944	-6.3806	0.347965	0	-0.86088	-1.47261	<b>1.942103</b>
Adagrad	0.992934	-1.89878	0.066751	0.166667	0.653065	<b>6.466366</b>	-0.20542
Adam	<b>2.31556</b>	<b>0.934882</b>	0.539524	0.272728	0.727536	0.824619	0.811082
SGD	<b>5.557139</b>	-1.95965	0.489154	0.227273	<b>3.608484</b>	0.436654	0.523297
RMSprop	<b>2.589096</b>	<b>0.905749</b>	0.75226	0.303031	0.69422	<b>1.101237</b>	0.494693

The above graph shows the performance of the activation functions

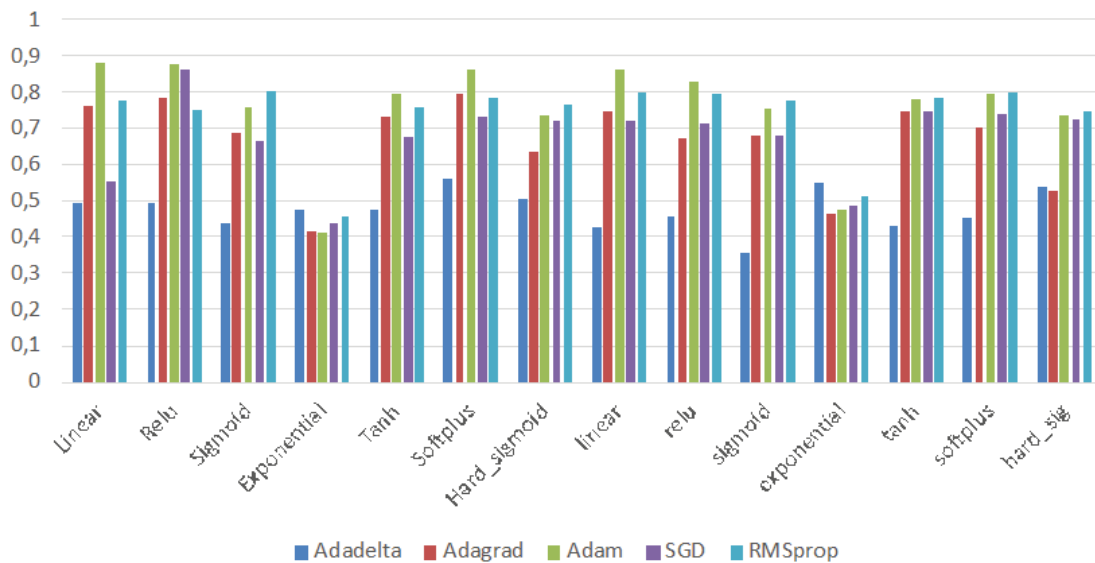


Figure 5.12: Accuracy comparison based on the modification of the data-set

### 5.3 Experiment-3: Impact of each feature of the neural networks

In experiment 3 only one of the least ranked features is removed and output is calculated by running the implementation for 12 times without each feature.

Table 5.14: Accuracy of Combination of algorithms without least ranked features

	Without FBS	Without Chol	Without Restecg
Adam/Linear	0.819258	0.797203	0.828402
Adam/relu	<b>0.899408</b>	<b>0.846154</b>	<b>0.821947</b>
Adam/Sigmoid	0.725121	<b>0.838623</b>	0.756321
Adam/Tanh	0.732652	<b>0.806348</b>	0.786444
Adam/Softplus	<b>0.835933</b>	<b>0.850995</b>	<b>0.832168</b>
SGD/Relu	0.674018	0.702528	0.713825
SGD/Softplus	0.733728	0.664874	0.670791
Adagrad/Relu	<b>0.750403</b>	0.657881	0.642281
Adagrad/Softplus	0.729962	<b>0.764927</b>	<b>0.788596</b>
Adadelat/Softplus	0.409898	0.272189	<b>0.838623</b>
Rmsprop/Linear	0.767617	0.763852	0.765465
Rmsprop/Relu	<b>0.846154</b>	<b>0.846154</b>	<b>0.846154</b>
Rmsprop/Sigmoid	0.729424	0.721356	0.754169
Rmsprop/Softplus	0.768155	<b>0.794513</b>	0.787520
Rmsprop/Tanh	0.775148	<b>0.799354</b>	0.747714
Rmsprop/Hardsigmoid	<b>0.781065</b>	<b>0.759866</b>	0.732652

Analysing the tables 5.2 and 5.3 containing the results of the accuracy that are obtained by training the MLP neural network integrated with back propagation algorithms, we can see that some of the overall better scores are obtained by combinations of the optimization algorithms such as *Adam* and *RMSprop* being on the top followed by the *SGD* joined with the activation functions namely *ReLU* and *Softplus* obtaining the top spots followed by the *Sigmoid* and *Tanh* yielded the more accuracy results.

Interesting part of the experimentation is the linear activation function which is on par with *ReLU* when combined with the *Adam* optimization algorithm in the table 5.2. Even though the fact that it is not good for multi-class classification because being a straight line function it produces constant on derivation which means the gradient has no relation with the input data stating that learning rate is not impacted by the input[12]. In table 5.2 linear function when used with the *SGD* neither improved nor gave loss result, the value surged to NaN or it was initialized value for other optimizing algorithms. For every epoch the loss produced is same as it was for the first epoch with every optimization algorithms.

Another issue concerning in case of linear function is that it defies the whole point of non-linearity which is the main purpose of the neural networks. Each layer having the linear function just calculates the weighted sum on that input and triggers signals based on the another linear function. This means having neural network having one or more hidden layer would just act as the single layer which is similar to the linear regression problem[13][12]. These facts are leading to ruling out the linear activation function for using in neural networks.

From the table 5.2 and 5.3 we can observe results of the *ReLU* and *Softplus* are very close when the model is trained with large data set and promising to be used more often with most of the optimization algorithms. Due to close similarities, the gap between the accuracy results is very minimal with *ReLU* leading with just **1.4156293%** for the model trained with least data. There is catch when coming to precision and recall because the results of the *softplus* managed to perform better in giving more positive results followed by the *Hard\_sigmoid* and *Tanh* in case of loss management which managed to minimize the loss or error in the model and which is worth noting that all the three activation functions performed better than the *ReLU* in the loss management.

Coming to the optimization algorithms we can observe in tables 5.4 and 5.5 that *Adam* is performing slightly well compared to others in all the metrics and next position is occupied by the *RMSprop* and *Adagrad*. Even though *Adadelta* is similar to *RMSprop* in terms of algorithms and formula but drawback is with initialisation of the learning rate because bad choice can increase the learning rate leading to the overfitting or underfitting of the model [33]. For adaptive learning rate methods like *Adadelta* doesn't need learning rates but good choice of initialisation of learning rate can give much better results than presented in the table 5.4 and 5.5. The most basic optimizer of all is *SGD*, outperformed the advanced learning rate algorithm *Adadelta* due to the initial learning rate value[28]. The reason for not promoting the linear activation function is its negligible contribution in the learning process when joined with the *SGD*. This *SGD* has shown the real effect on the linear activation function in all the cases because of the constant derivative of the linear activation function and avoiding the algorithm to improve the learning rate or the knowledge.

In case of the adaptive learning rate methods namely *Adam*, *Adadelta*, *Adagrad* and *RMSprop* can learn without the initial learning rate values and the reason we can see loss, precision and recall from the tables 5.4 to 5.9 metrics are much better than *SGD*. The precision should be between the range of 0 and 1 but here in these tables 5.6 and 5.7 we can observe that precision values are out of range for *Linear*, *ReLU* and *Softplus* for the model trained with least instances because output of these activation functions is continuous values but the actual output is between range 0 and 1. Through this observation we can say by using the *Sigmoid*, *TanH* and *Hard\_sigmoid* which produce the output range between the 0 and 1 would give optimal results for the precision. Using the *Sigmoid*, *Tanh* and *Hard\_sigmoid* in the output layer would give the precision value range between 0 and 1. This is the effect of activation functions on the performance of the neural network for precisely identifying the class of the input. The relation between precision and recall is inversely proportional. So, from the tables 5.6 to 5.9 we can observe that precision is high causing the recall to be lower. We can see from the table 5.8 the recall values w.r.t optimization algorithms *Adam*, *Adagrad* and *SGD* with activation functions *ReLU*, *Sigmoid* and *Hard\_Sigmoid* are stable considering that the output is continuous value. Here comes to end of the first research answer.

For the Research Question-2, the tables 5.10, 5.11, 5.12 and 5.13 in the section 5.2 are the results of the neural network trained with the data-set with only features of highest ranks and less instances i.e with 300 instances. By comparing the results with metrics presented in the section 5.1, we can see that omission of the features did affect the performance of the combinations of the algorithms and functions used in the neural network. From the above analysis explained in the previous paragraphs, there are possible best performing combinations from optimization algorithms like (*Adam*, *Adagrad* and *RMSprop*) paired with activation functions like (*ReLU*, *Softplus*, *Sigmoid*, *TanH*, *Hard\_Sigmoid*) are compared with performance results obtained from the neural network trained with the data-set without the least ranked feature from the table 5.1.

From the observations we can see that difference in the performance of the neural

network trained with the data-set with all features and deprecated features is minuscule. Here the *Adam* with most of the activation functions got effected but the *RMSprop* lead to the top with steep increase in the overall performance of neural network based on the metrics tables in the section 5.2. The decrease in the performance of *Adam* with *ReLU* function which combination performed well when original data-set with 13 attributes is used but is due to the decrease in the data for training the model. That is the reason the performance of the *Adam* decreased with every activation function with **0.052%** reduction in performance. *RMSprop* dominated with gain of **0.07794%** in performance[28][25]. This *RMSprop* is followed by the *Adam* and *Adagrad*. From this analysis we can say that though the performance of the selected combinations from the previous research answer got impacted with decreased values but did not affect the selection of the combinations. Because feature deprecation did decrease the overall performance of all the neural network model trained with different combinations of the activation functions and optimization algorithm.

From the experiment 3 in results section, we can see the affect of individual feature on the performance of the combination of the optimization algorithms and activation function. The results are analysed based on the comparison with the results table in experimentation 2 in All the three feature impacted the performance of the combination of the specific optimization algorithms namely *Adam*, *RMSprop* and *Adagrad* and the activation functions namely *ReLU*, *Softplus*, *Tanh*, *Sigmoid* and *Hard\_sigmoid*. As can be seen from the table the performance increased for the combination from the above listed optimization algorithms and activation functions. Performance of the *Adam/Relu* has increased dramatically and could be considered as the most effective combination of all when compared with the experimentation 2 results in results section.

The solution to Research Question-3 is as follows, where based on information gain, least-ranked attributes have been eliminated one at a time to check the performance of the neural network. The missing data is due to elimination of attributes. Gradually when each of the least-ranked attribute was eliminated and was experimented, the accuracy of the neural network in case of few combinations have been increased. As mentioned in Table 5.14, without the attribute FBS the combinations that outperformed the other combinations are *Adam/Relu*, *Adam/Softplus*, *Adagrad/Relu*, *Rmsprop/Relu*, *Rmsprop/Hardsigmoid*. Also, without Chol attribute, combinations like *Adam/Relu*, *Adam/sigmoid*, *Adam/Tanh*, *Adam/Softplus*, *Adagrad/Softplus*, *Rmsprop/Relu*, *Rmsprop/Softplus*, *Rmsprop/Tanh* *Rmsprop/Hardsigmoid* have performed way better than the other combinations. Similarly, without Restecg, combinations like *Adam/Relu*, *Adam/Softplus*, *Adagrad/Softplus*, *Adadelat/Softplus*, *Rmsprop/Relu* have performed well when compared to other combinations. According to the above comparisons, *Adam/Relu*, *Adam/Softplus*, *Rmsprop/Relu* have performed the best among all the comparisons. Therefore, these three combinations give their best performance in spite of having least amount of attributes in the data set.

In neural network, each attribute contributes to the outcome of the neural network. Also, the attributes have the priority hierarchy. The impact of the performance of the neural network depends on the priority of the attribute. If a high-priority attribute is removed while experimenting, then the performance of the neural network

decreases drastically. The impact of elimination of the least priority attribute in the experiment was almost negligible. Hence, the priority of the attribute is directly proportional to the performance of the neural network.

## Chapter 7

---

# Conclusions and Future Work

In this study we analysed the performance of the different combinations of the optimization algorithms and activation functions used in the neural networks trained with the structured data. We have total of 35 combinations out of which 15 combinations are selected based on the metrics obtained. Based on the analysis we can obtain possible combinations of the specific optimization algorithms namely *Adam*, *RMSprop* and *Adagrad* and the activation functions namely *ReLU*, *Softplus*, *Tanh*, *Sigmoid* and *Hard\_sigmoid*. Performance of the neural networks also depends on the no of nodes and layers count for that reason we have used loss, precision and recall that gives clear idea of how optimization algorithm and activation function are together performing on giving the best results possible.

Based on the results obtained for the neural network trained with the data-set which has been modified with omission of the features with least ranks, we can say that neural networks needs to be trained with optimal data-set. The original data-set with all the features is the optimal one and gave less loss to the model when compared to the customized data-set. This variation in the performance of the neural network trained with the original and customized one occurred because the features are ranked based on the *InfoGainAttributeEal* and *CoorelationAttributeEval* methods which evaluate the feature based on the co-relation to all the attributes. This is the reason behind the sudden decrease in the performance of the neural networks in which the learning rate depends on the features. The overall performance of the combination decreased but the selection of the combination did not get affected by training the model with feature deprecated data-set.

From the experimentation 3 we can observe the performance variance of the combination of the optimization algorithm and activation functions used in the neural network. Based on the analysis of the results we conclude that omission of individual features will affect the performance of the combination used in the neural networks either by increasing or decreasing the performance based on the information gain of the features.

As for future work, we can use the deep learning neural networks and use multiple combinations instead of just pairs. Grasping the inference time for the neural network would help in knowing time consumed for generating the model. We have used only structured data but this analysis can be done on the various data types like time series, statistical etc,



---

## References

- [1] Heart Disease UCI. Library Catalog: [www.kaggle.com/ronitf/heart-disease-uci](http://www.kaggle.com/ronitf/heart-disease-uci),[Online accessed, 2020-05-09].
- [2] Matplotlib: Python plotting — Matplotlib 3.2.1 documentation. <https://matplotlib.org/>,[Online accessed, 2020-05-09].
- [3] Neural Networks - What are they and why do they matter? [https://www.sas.com/en\\_us/insights/analytics/neural-networks.html](https://www.sas.com/en_us/insights/analytics/neural-networks.html),last accessed on 2020-05-05,.
- [4] NumPy — NumPy. <https://numpy.org/>,[Online accessed, 2020-05-09].
- [5] pandas - Python Data Analysis Library. <https://pandas.pydata.org/>,[Online accessed, 2020-05-09].
- [6] Prediction of heart disease using neural network - IEEE Conference Publication.
- [7] scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation. <https://scikit-learn.org/stable/>,[Online accessed, 2020-05-09].
- [8] TensorBoard. Library Catalog: [www.tensorflow.org](http://www.tensorflow.org).
- [9] Welcome to Python.org. <https://www.python.org/>,[Online accessed, 2020-05-09].
- [10] Scikit-Learn In Python - Important Machine Learning Tool, January 2015. <https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/>,[Online accessed, 2020-05-09].
- [11] NumPy in Python | Set 1 (Introduction), January 2017. <https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/>,[Online accessed, 2020-05-09].
- [12] Why you shouldn't use a linear activation function, June 2019. Library Catalog: [www.machinecurve.com](http://www.machinecurve.com).
- [13] Activation Functions | Fundamentals Of Deep Learning, January 2020. Library Catalog: [www.analyticsvidhya.com](http://www.analyticsvidhya.com).
- [14] Anaconda (Python distribution), April 2020. [https://en.wikipedia.org/w/index.php?title=Anaconda\\_\(Python\\_distribution\)&oldid=953681663](https://en.wikipedia.org/w/index.php?title=Anaconda_(Python_distribution)&oldid=953681663),[Online accessed, 2020-05-09].

- [15] Artificial neural network, January 2020. [https://simple.wikipedia.org/w/index.php?title=Artificial\\_neural\\_network&oldid=6764482](https://simple.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=6764482), [Online; accessed 2020-05-04].
- [16] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, March 2020. Google-Books-ID: tZnSDwAAQBAJ.
- [17] Jun Chin Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed. Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5):971–989, September 2016.
- [18] Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, and Hermann Ney. Empirical Investigation of Optimization Algorithms in Neural Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):13–25, June 2017.
- [19] Jason Brownlee. How to Perform Feature Selection With Machine Learning Data in Weka, July 2016. <https://machinelearningmastery.com/perform-feature-selection-machine-learning-data-weka/>, [Online accessed, 2020-05-10].
- [20] Sriramakrishnan Chandrasekaran. A Machine Learning Implementation of Predicting the Real Time Scenarios in a better way. page 12.
- [21] Bhaskar DasGupta and Georg Schnitger. The Power of Approximating: a Comparison of Activation Functions. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 615–622. Morgan-Kaufmann, 1993.
- [22] Sankaran Gnanambal, Muthuraman Thangaraj, V. T. Meenatchi, and V. Gayathri. Classification Algorithms with Attribute Selection: An Evaluation Study using WEKA, 2018. Library Catalog: [www.semanticscholar.org](http://www.semanticscholar.org), [Online accessed, 2020-05-11].
- [23] Tülay Karayılan and Özkan Kılıç. Prediction of heart disease using neural network. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 719–723, October 2017.
- [24] Bekir Karlik and A Vehbi Olgac. Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. page 12.
- [25] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. arXiv: 1412.6980.
- [26] Ilias G. Maglogiannis. *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*. IOS Press, 2007. Google-Books-ID: vLiTXDHr\_sYC.

- [27] Arpita Nayak and Kaustubh Dutta. Impacts of machine learning and artificial intelligence on mankind. In *2017 International Conference on Intelligent Computing and Control (I2C2)*, pages 1–3, June 2017.
- [28] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, June 2017. arXiv: 1609.04747.
- [29] Mohamed A. Shahin, Mark B. Jaksa, and Holger R. Maier. *Artificial Neural Network Applications in Geotechnical Engineering*.
- [30] SAGAR SHARMA. Activation Functions in Neural Networks, February 2019. <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>, [Online accessed, 2020-05-11].
- [31] P Sibi, S Allwyn Jones, and P Siddarth. ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS. . *Vol.*, 47:5, 2005.
- [32] Avinash Sharma V. Understanding Activation Functions in Neural Networks, March 2017. <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>, [Online accessed, 2020-05-10].
- [33] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*, December 2012. arXiv: 1212.5701.





