Age classificaton of Abalone Dataset using Neural Network

Assessment 3 Report of ZZSC6836 - Data Mining and Machine Learning

Rushmila Islam

2023-12-03

Contents

1 Abstract

Many machine learning methods are applied to solve classification tasks, such as Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbour (k-NN). Artificial Neural Network (ANN) has been proved to be a powerful methodology to build a model to solve classification problem. Neural Networks (NN), a promising dynamic network algorithm, works efficiently and effectively on both regression and classification tasks. We first build a feedforward network for this classification task of Abalone age prediction. Then, we study the performance of the model by adjusting the number of hidden neurons, by comparing the optimizer Stochastic gradient descent (SGD) and Adam, by adding different number of hidden layers to the network. Finally, we evaluate, identify, and explain the best NN model using the accuracy score, confusion matrix and Receiver Operating Characteristic (ROC) and Area Under the ROC Curve (AUC) of different classes of this multi-class problem.

2 Introduction

During the mid 1980s multiple research works independently rediscovered the Backpropagation algorithm (@rumelhart1986learning) - a way to solve chains of mathematical operations using gradient-descent optimisation as a novel approach to train large NN. Backpropagation is a way to efficiently calculate the gradient of a loss function with respect to the weights and biases of a neural network. The gradient can then be used to update the weights and biases using gradient descent which is an optimisation algorithm that iteratively moves the weights and biases in the direction of decreasing loss. Early neural networks showed great success for image recognition problems such as classifying handwritten digits. However, due to the lack of large training dataset and computational constraints neural networks eventually sent back to oblivion. With the availability of large dataset and the advent of modern GPUs since 2010s deep neural networks (DNN) have become the go-to algorithm to solve perceptual machine learning problems - a task that involves understanding and interpreting images, sounds, videos, and natural languages (@krizhevsky2012imagenet). DNN is nowadays effectively used to solve a broad range of supervised machine learning problems using structured and labelled train dataset and using multilayer perceptron (MLP) as a building block. In its most simplistic form a DNN i.e., a MLP consists of an input layer, activation function, hidden layers, output layer, gradient descent, and iterations till network convergence. For the purpose of this project, we are tasked to predict the age of Abalones that are marine snails that live in rocky coastal waters around the world. We use simple to complex DNN for this task and analysis the performance of the NN at different settings to determine the best performing model.

The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. By predicting abalones' age, researchers can understand the environment better and protect this species. The dataset used in this analysis is obtained from the UCI Machine Learning Repository (@abalone2023dataset). The dataset contains measurements of abalones, which are large sea snails. The measurements include the length, diameter, height, whole weight, shucked weight, viscera weight, and shell weight of each abalone. It is a popular choice for machine learning tasks such as regression and classification. The target variable is the age of the abalone, which is determined by counting the number of rings in its shell. The dataset is challenging because the age of an abalone is difficult to predict from its physical measurements. However, the dataset is well-suited for machine learning algorithms that can learn complex relationships between features.

In this project, we perform exploratory data analysis of the Abalone dataset and then perform neural network modelling to predict the age of the abalone using the input features and applying different techniques. Eight physical measurements are taken as input features. The ages are outputs, uniquely labelled from 1 to 29. For this study we are considering it as a multi class classification problem and classifying ring age into four age groups - class 1 (0-7 years), class 2 (8-10 years), class 3 (11-15 years), and class 4 (Greater than 15 years). There are three primary challenges in this task. Firstly, there is the lack of instances for some classes e.g., Abalones aging at 29 years are very rare, and in the given dataset, there is only one instance of it. Second, the difference among abalones which have similar age is very small. Finally, the instances are not uniformly distributed among all the ages e.g., there are more instances for ages 10 and 11 years. We start with basic data exploration and pre-processing, then perform both univariate and bivariate analysis to understand the data distributions, relationships, and patterns within the features. We start with a simple MLP model of a single layer NN with five neurons and then gradually increase the number of neurons to 10, 15, and 20 while using a SGD optimiser (@bottou2010large). Later, we investigate the effect of learning rate and different number of hidden layers i.e., one and two. We also investigate the effect of Adam (@kingma2014adam) - another popular optimiser - and SGD on both training and test performances. Finally, we evaluate the best NN model using classification accuracy and explain our analysis using confusion matrix and ROC/AUC for different age groups. In conclusion, our results suggest that determining an appropriate set of input features for such regression model is challenging and indicates more elaborate dataset containing additional features might have been helpful in determining the abalone age.

3 Exploratory Data Analysis

In this section, we perform basic data analysis to understand the nature of the data, check for missing values. We also observe the standard data descriptions like range, min-and-max values, percentile, mean, standard deviations, and the overall spread and variety of the individual column values.

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 4177 entries, 0 to 4176
## Data columns (total 9 columns):
##  #   Column          Non-Null Count  Dtype
## ---  ------          --------------  -----
##  0   Sex             4177 non-null   object
##  1   Length          4177 non-null   float64
##  2   Diameter        4177 non-null   float64
##  3   Height          4177 non-null   float64
##  4   Whole_weight    4177 non-null   float64
##  5   Shucked_weight  4177 non-null   float64
##  6   Viscera_weight  4177 non-null   float64
##  7   Shell_weight    4177 non-null   float64
##  8   Rings           4177 non-null   int64
## dtypes: float64(7), int64(1), object(1)
## memory usage: 293.8+ KB
```

```
##             Length     Diameter  ...  Shell_weight        Rings
## count  4177.000000  4177.000000  ...   4177.000000  4177.000000
## mean      0.523992     0.407881  ...      0.238831     9.933684
## std       0.120093     0.099240  ...      0.139203     3.224169
## min       0.075000     0.055000  ...      0.001500     1.000000
## 25%       0.450000     0.350000  ...      0.130000     8.000000
## 50%       0.545000     0.425000  ...      0.234000     9.000000
## 75%       0.615000     0.480000  ...      0.329000    11.000000
## max       0.815000     0.650000  ...      1.005000    29.000000
##
## [8 rows x 8 columns]
```

In the Abalone dataset, nine physical measurements describe an abalone and the last measurement, the number of rings, represents an Abalone's age. The number of instances is 4177.

We observed from the abalone dataset is all the features except 'Sex' and 'Age' have values all continuous float numbers. Additionally, we perform basic data pre-processing tasks to check for missing values and found there are no missing values in the dataset. We also observe the standard data descriptions like range, min-and-max values, percentile, mean, standard deviations, and the overall spread and variety of the individual column values. We see that there is no missing data in the dataset.

The range of an Abalone's age is an integer between 1 to 29. Thus, we take it as a classification task. For this project we are classifying age groups in 4 different groups. Eight physical measurements are input features and they are mapped to 4 classes. We plot out all the instances in Fig.1. Fig.1 shows that the distribution of Abalone's age approximates to a normal distribution. For some ages, the instances are rare. For example, there is no abalone aged 28 in the dataset.

3.0.1 Correlation matrix

Figure 2 shows. the correlation matrix with all the input features. From the diagram we see Whole Weight is almost linearly varying with all other features except Rings. Height has least linearity with remaining

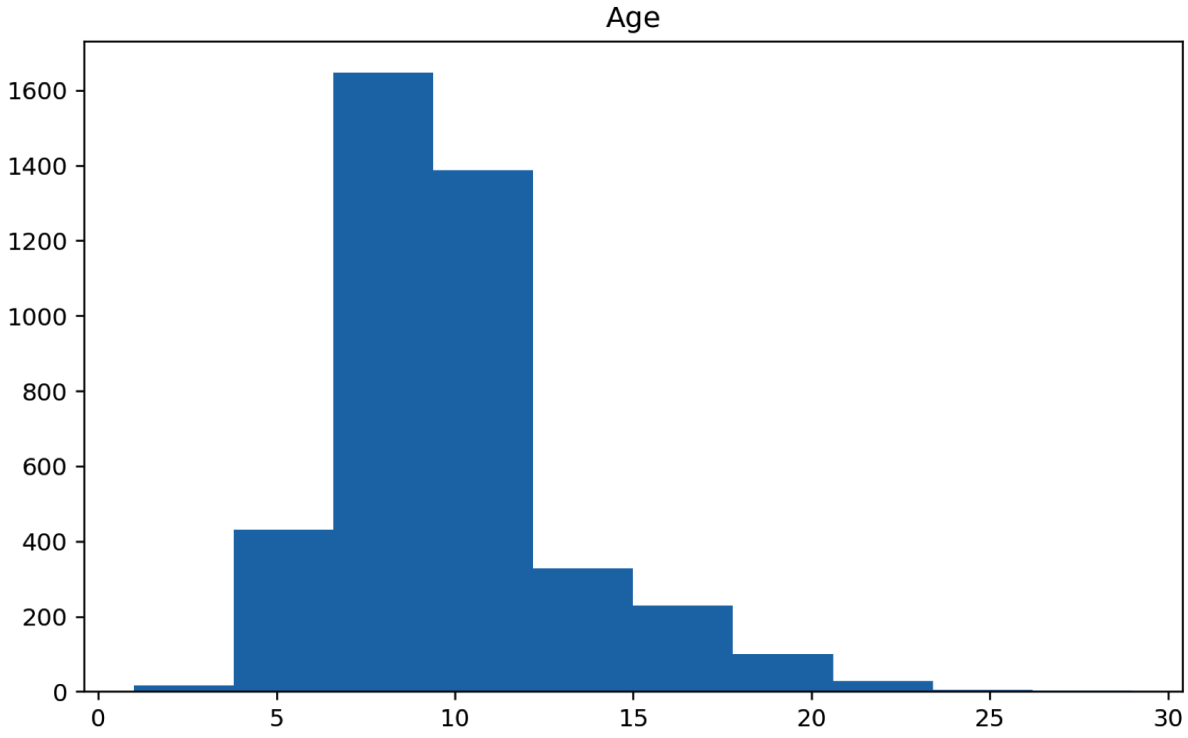


Figure 1: Age distribution

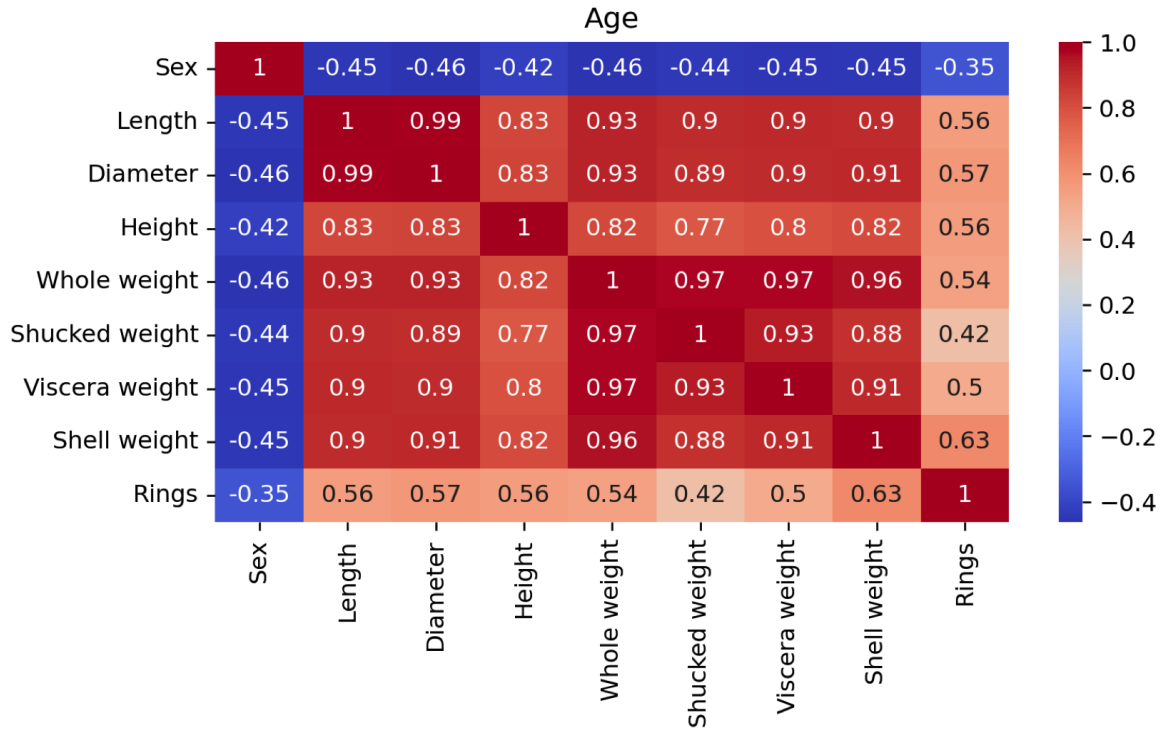


Figure 2: Correlation matrix

features. Number of rings is most linearly proportional with Shell weight, Diameter and Length. Rings is least correlated with Shucked weight. We observe from the correlation matrix that Length - Diamete' and Whole weight - Shucked weight has positive correlation as well as Sex - Height (-0.42) and Sex - 'Shucked Weight'(-0.44) have least correlation.

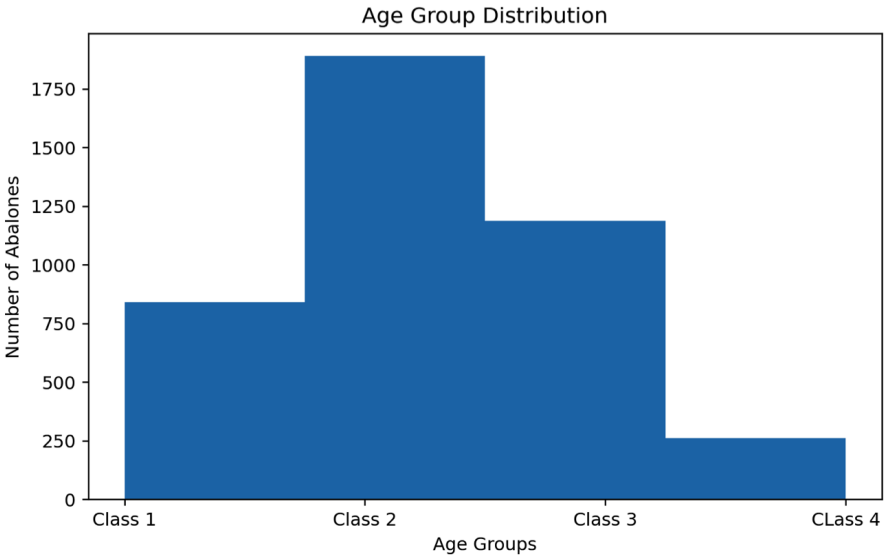


Figure 3: Age class distribution

3.0.2 Feature Analysis

Fig 4 shows the distribution of all the features along with the target - age group in the dataset. We can say from that all the features has normal distribution. We also learn that most most data belongs to age class 2 and age class 3. In summary the age class data are not evenly distributed in the groups.

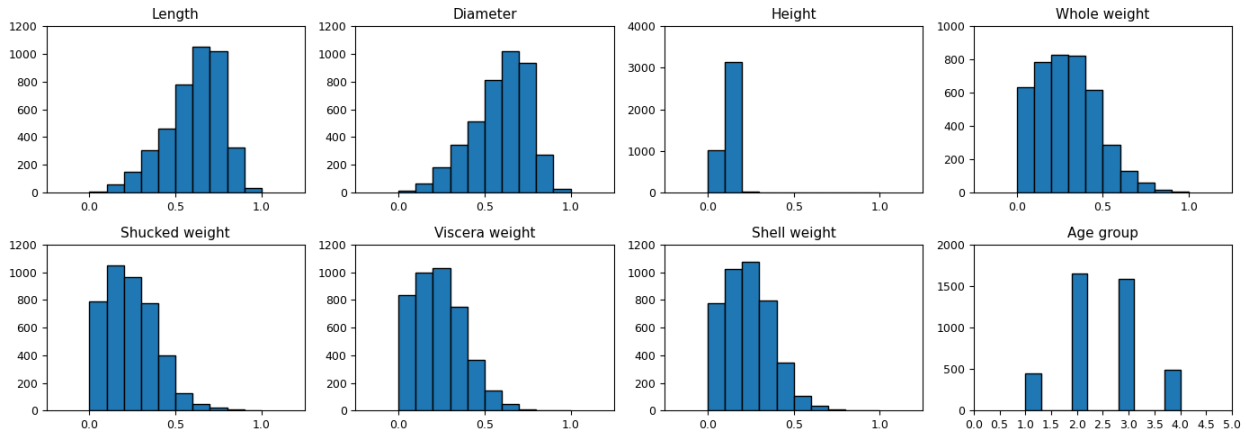


Figure 4: Univariate histogram

TBD

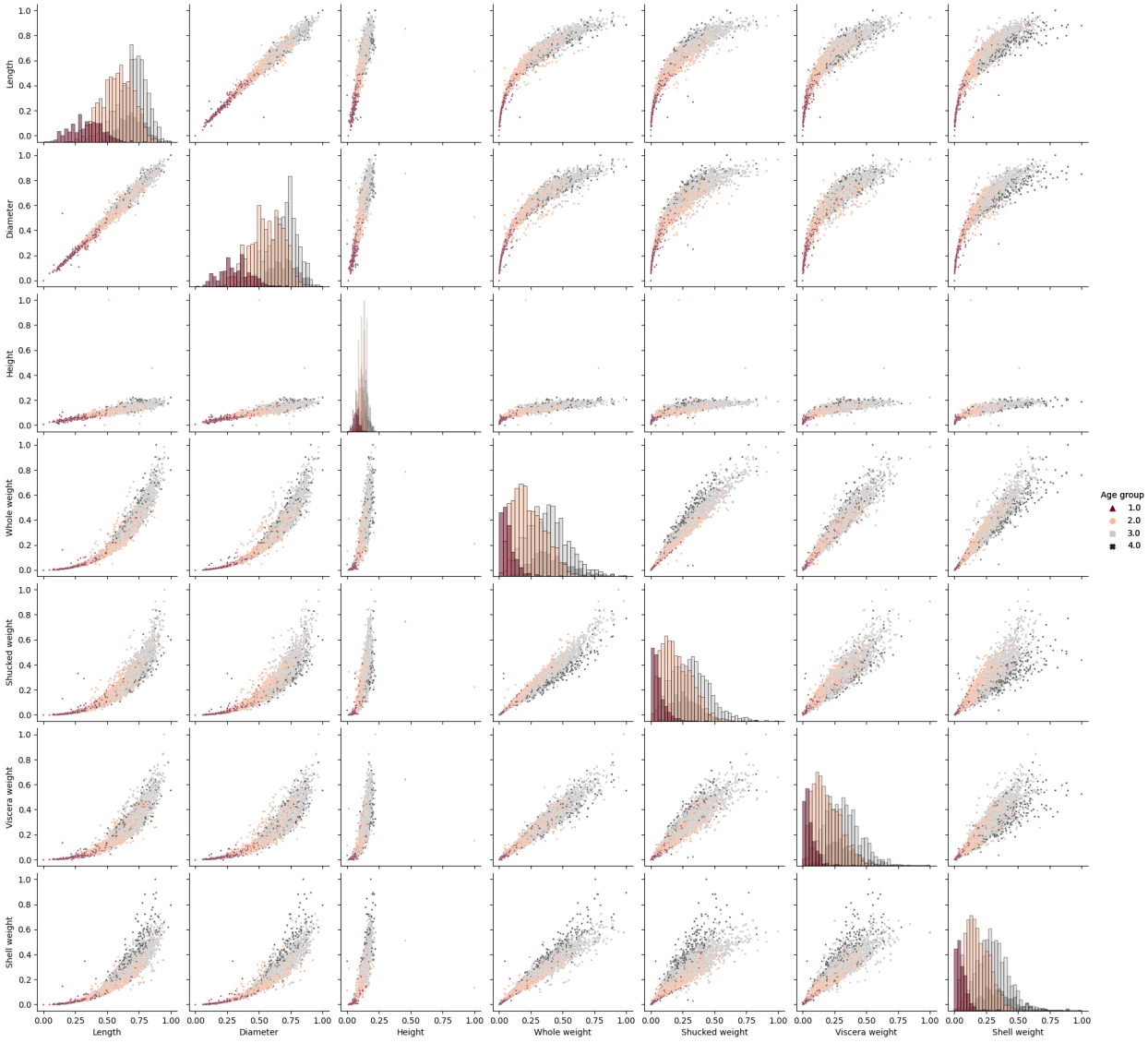


Figure 5: Bivariate pair plot by Age Class

4 Preparing the data

For the preparation of NN modelling, we first need to prepare the dataset. At the very begging we rename our columns for more readability. We remaned the column accordingly using the site reference - https://archive.ics.uci.edu/dataset/1/abalone. Now we need to separate the feature and target dataset. We use the Rings column and convert it to Age Class as the target and other columns as features. We now mark the age of the abalone with appropriate age groups. We are splitting data into four age groups - class 1 (0-7 years), class 2 (8-10 years), class 3 (11-15 years), and class 4 (Greater than 15 years).

4.0.1 One Hot Encoding

In the dataset the feature 'Sex' feature has three categories - M, F and I. For the modelling purpose we mapped this to numerical values - 0, 1 and 2 respectively. The main reason to encode 'Sex' in this way because this feature should have similar magnitude with other features. This is called integer encoding. We categorise the age group in four classes and treating it as four-category classification task, we add a new column age_class and assign the corresponding age group to the class number. Age 0 to 7 are mapped to class 1, age 8-10 to class 2, age 11-15 to class 3 and age greater than 15 years to class 4 respectively. The target feature and variable contains different categorical values.The target variable represents multi-class values. As we are modelling multi-class classification problems using neural networks, it is good practice to reshape the output attribute from a vector that contains values for each class value to a matrix with a Boolean for each class value and whether a given instance has that class value or not. This technique is called one-hot encoding or creating dummy variables from a categorical variable. In preparation for the NN modelling, we separate out feature and target variable and apply one hot encoding to the categorical columns - Sex and Age which should improve our model performance (@onehot2023jason).

5 Experimental Setup

Before setting up the NN for experiment, we need to split the dataset for taining and testing. Here we are using 60/40 percent of train and test split in the dataset. We are using fixed random state (42) for the splitting so the data split reamains same across all the experiment. Also we using 'startify' to ensure that the split maintains the same proportion of classes in the target variable in both the training and testing sets.

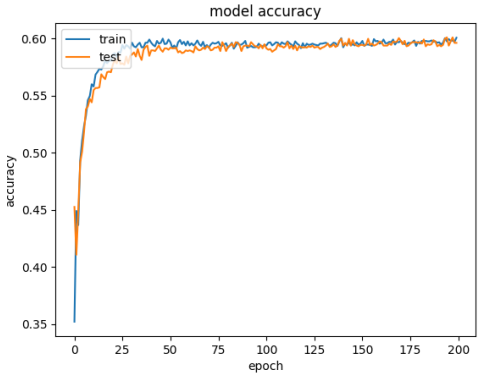
To run this experiment we need to setup a NN model with 10 input neurons and 4 output neurons. The intermediate layers will use 'relu' (rectified linear unit) as their activation function and the final layer will use 'softmax' activation. The 'relu' activation turns. the negative values into zeros and only work with the positive values. 'Softmax' is often used in the output layer of a NN model for multi-class classification problems. It predicts the class with the highest probability after applying the softmax activation. Now we need to choose a loss function and an optimizer. Because we are facing a multi-class classification problem, it's best to use 'categorical_crossentropy' loss function. There's another loss function used for classification problem, that is - 'sparse_categorical_crossentropy'. We will be using 'categorical_crossentropy' as our target vector we have applied one-hot-encoding, for 'sparse_categorical_crossentropy' it does not need the encoding. For the experiments we setup 5 NN models - 3 of the models with SGD optimizer and 2 using Adam optimizer to analyse the effect of different type of optimizers. We also vary the learning rate from 0.01 to 0.001 to analyse the effect of learning rate in the model. We have also used models with multiple hidden layers to understand the effect on performance. We have executed 10 experimental run while model evaluation. Initially to assign weight for each experimental run we have used 'he_uniform' as kernel inititalizer. We also considered using 'he_uniform' and 'random_normal' as optimizer. As we are using 'relu' activation for the hidden layer, it is recommended to use either 'he_uniform' or 'he_normal' optimizer for better performance. Now between 'he_uniform' or 'he_normal', we have done experiments and result shows that 'he_uniform' gives better result, hence we are using 'he_uniform'.

6 Result Analysis

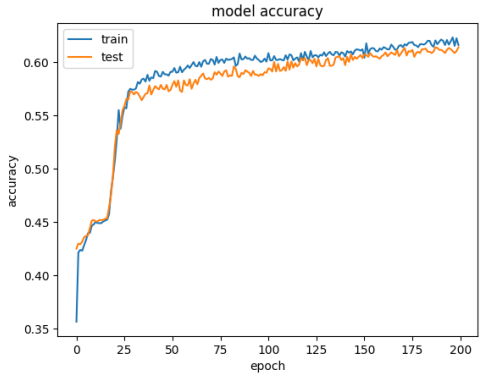
6.1 The Effect of Number of Hidden Neurons for a single hidden layer (using SGD)

For this analysis we start with 5 hidden neurons and consider as a baseline to compare others. We can observe from the graph is with 5 hidden neurons the model performance is growing over time, which means the model accuracy is improving. We also see it grows at the beginning, but over time it reaches a plateau, meaning it's not able to learn any more. In Table 1, we see the models accuracy when there are with various number of hidden neurons. The accuracy curve shows a good fit as the training loss and the validation loss both are decreasing to point of stability and the validation loss curve has small gap with the training loss. In Table 2, we observe that when the network is smaller the model accuracy takes longer epochs to train the model and get the higher accuracy, whereas bigger networks with 10, 15 and 20 hidden neurons reach the higher accuracy level much faster. The confidence interval of accuracy for a 5 hidden neurons model is (0.58 to 0.624). We are 95% confident that the true accuracy of the model falls within the range of 0.58 to 0.624. When the network has more hidden neurons validation loss gets lower quickly comparing to the base model. Smaller network starts over fitting later than the bigger network. The bigger the network the training loss gets near zero very quickly. The more capacity the network has, the more quickly it can model the training data.

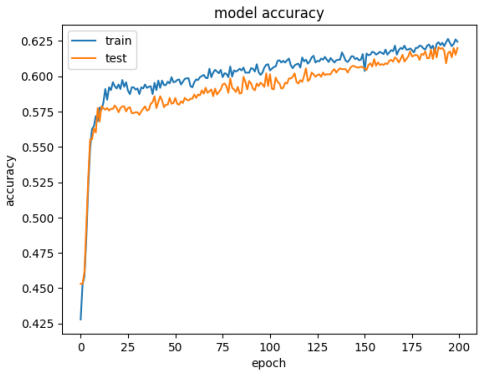
Table 1: Model Accuracy for models with single layer and 5, 10, 15 and 20 hidden neurons



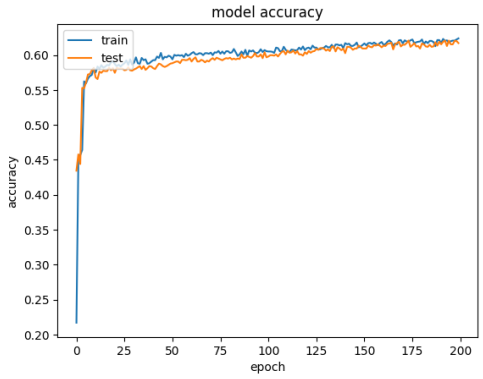
Accuracy with 5 neurons



Accuracy with 10 neurons

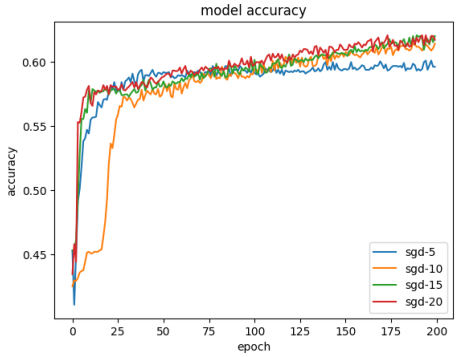


Accuracy with 15 neurons

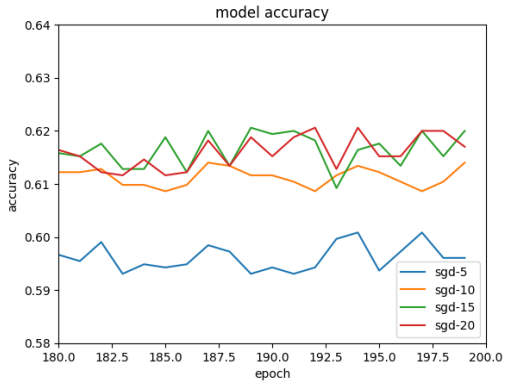


Accuracy with 20 neurons

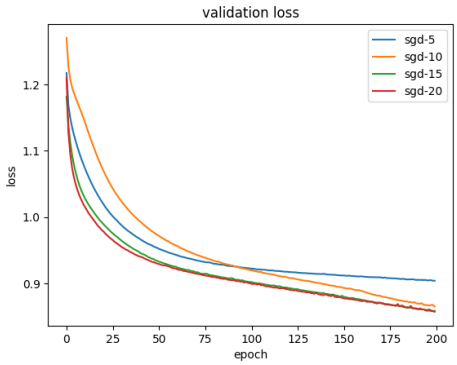
Table 2: Model accuracy and validation loss comparison for models with single layer and 5, 10, 15 and 20 hidden neurons



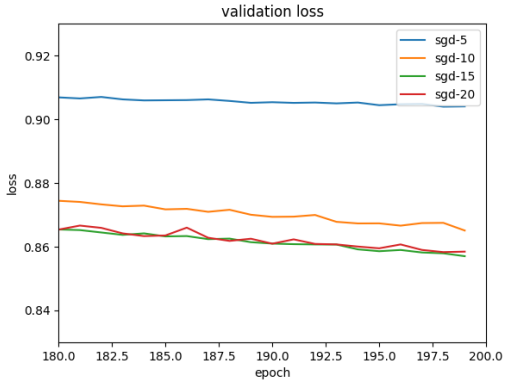
Accuracy Curve with multiple numbers of hidden neurons



Extended Accuracy Curve with multiple numbers of hidden neurons



Loss Curve with multiple numbers of hidden neurons



Extended Accuracy Loss with multiple numbers of hidden neurons

According to the Table 2, we observe that model accuracy is higher when we use more hidden neurons. But adding more hidden neurons doesn't always imply better model. The models with 15 neurons seems to result better accuracy than 20 neurons. It also reflects the loss curve where we can see that model with 15 and 20 neurons shows better result than other models but the one with 15 neurons is much consistent.

The below table shows the confidence interval of accuracy and the mean accuracy of the model.

Table 3: Mean and Confidence Interval of accuracy for a single hidden layer varying neurons

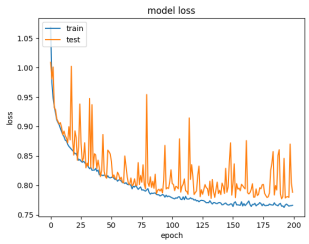
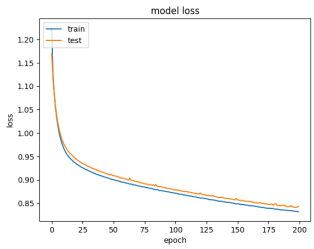
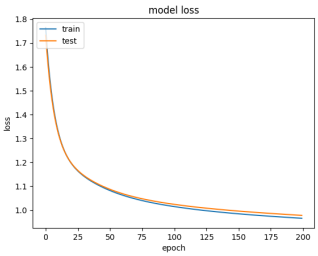
No of neurons	Mean Test Accuracy	95% Confidence Interval of Accuracy
5	0.61	(0.57, 0.62)
10	0.61	(0.59, 0.63)
15	0.62	(0.60, 0.64)
20	0.62	(0.59, 0.64)

Table 3, also shows that model with 15 hidden neurons 60% to 64% correct predictions, which is also higher than other models.

6.2 The Effect of Learning Rate using SGD

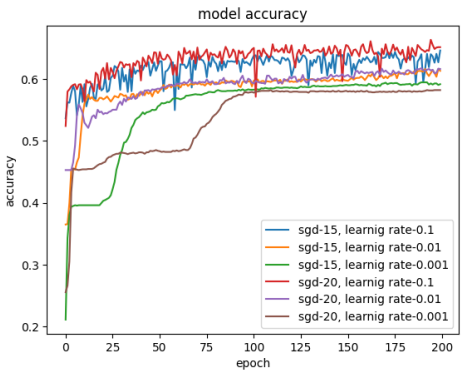
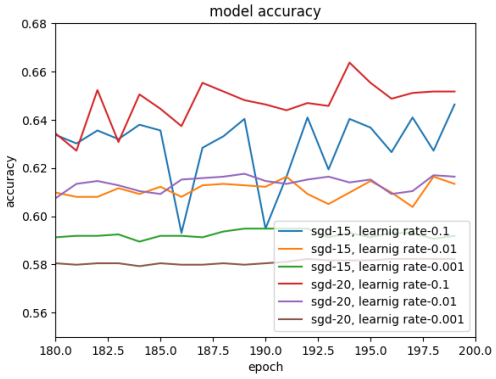
As in the previous section we found that using 15 and 20 hidden neurons in a single layer model gives better performance, using this information we carry our next experiment varying the learning rate to fine tune the model for better performance. As in previous model we used the default learning rate for SGD optimizer which is 0.1, 0.01 and 0.001, we will vary the learning rates and compare the result.

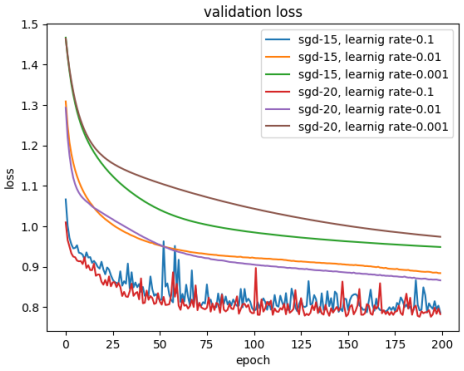
Table 4: Validation loss curve for models with 20 hidden neurons with different learning rates

		
Model with 20 hidden neurons and learning rate 0.1	Model with 20 hidden neurons and learning rate 0.01	Model with 20 hidden neurons and learning rate 0.001

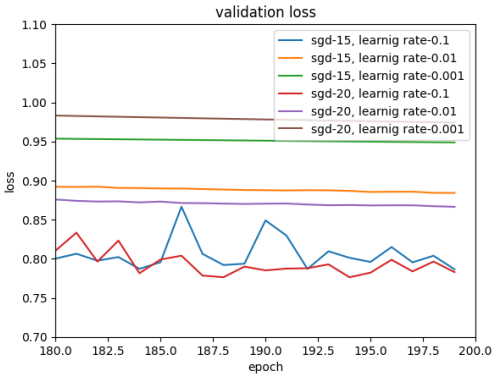
We can observe from the graphs in Table 4, that the first model using 0.1 learning rate showing an overfit model, the fast learning rate model curve also indicating that the validation dataset may be too small relative to the training dataset. As our dataset is relatively small (4177 rows), considering that the learning rate 0.1 is not giving ideal result. Whereas we can see that learning rate with 0.01 and 0.001 provides better result and also good fit.

Table 5: Accuracy and loss curve comparison for models using SGD optimizer with different learning rates

	
Model accuracy curve	Expanded accuracy curve



Loss curve



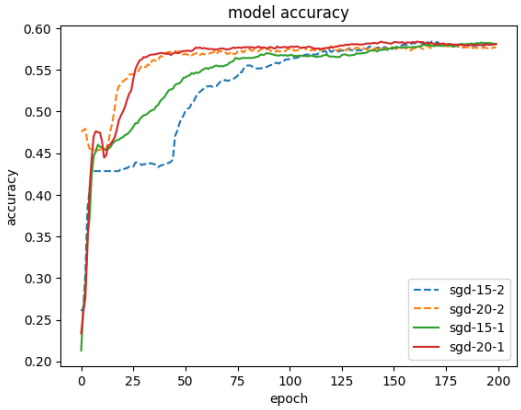
Expanded Loss curve

Based on the analysis, the models with 15 and 20 neurons with learning rate 0.01 will the optimal options to choose to build a better performing model.

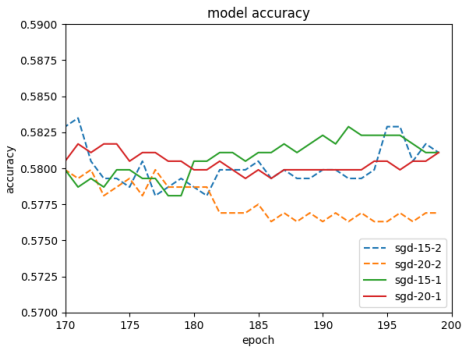
6.3 The Effect on Different Number of Hidden Layers

In the earlier section we saw that models with 15 and 20 neurons with a single hidden layer is a good option for building a better model. Keeping this in our mind we have done the next experiment with 2 hidden layers with 15 and 20 neurons each layers and compared with single layer models also compared with different learning rates to understand better model.

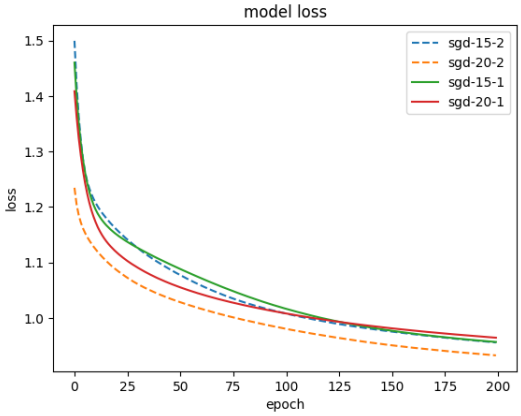
Table 6: Comparison of models with 1 and 2 hidden layers with SGD optimizer using 15 and 20 hidden neurons



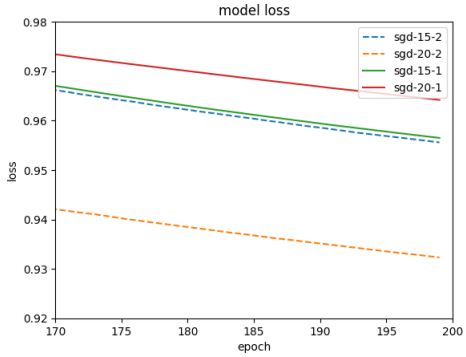
Accuracy curve for models with multiple hidden layers



Expanded accuracy curve with multiple hidden layers



Loss curve with multiple hidden layers



Expanded loss curve with multiple hidden layers

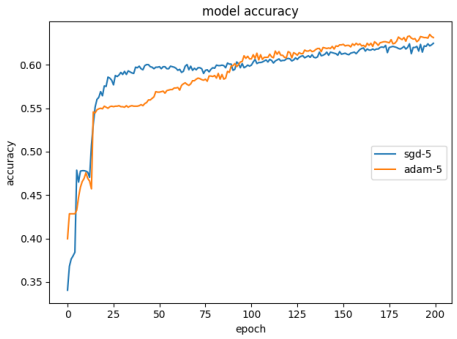
We can observe from the Table 6, learning curves that for the models with 2 hidden layers with 15 neurons and 20 neurons on each comparing with a single hidden layer. The model with 20 neurons and 2 hidden layers gets the peak accuracy level faster than other models and also the loss the shows that it is a validation loss is much smaller than other models.

6.4 Effect of Adam and SGD on training and test performance

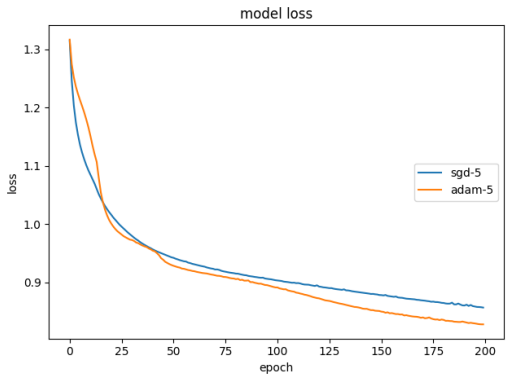
In this section we are evaluate the two optimization algorithms - Adam and SGD. All the experiments we have done so far using SGD optimization, now we will be also using Adam as optimizer an observe the performance of our models. While using SGD we have applied various learning rates (0.1, 0.01, 0.001) to evaluate the performance. In case of Adam optimizer, it adapts the learning rates for each parameter individually and dynamically adjust the learning rates during training.

We have perform below comparison with 5 hidden neurons single layer model using SGD and Adam optimizer where the no manual learning rate is applied.

Table 7: Comparison between SGD and Adam (5 hidden neurons)



SGD vs Adam Accuracy



SGD vs Adam Validation Loss

Table 7, showing learning curves for models using SGD and Adam optimizer. We can clearly observe from the accuracy curve that Adam optimizer reaches better accuracy than SGD and also in the loss curve shows that model loss of Adam is reaching towards zero faster than the SGD one.

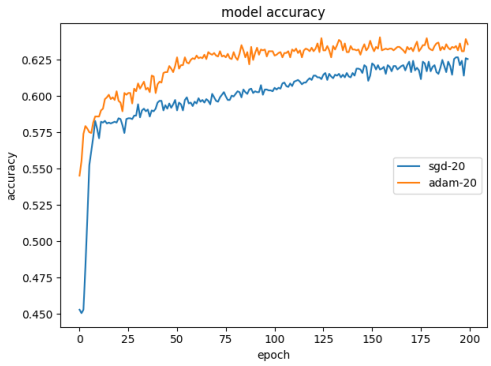
Table 8, shows the performance comparison of the models with single layer 5 hidden neurons using SGD and Adam optimizer.

Table 8: Adam vs SGD single hidden layer with 5 neurons

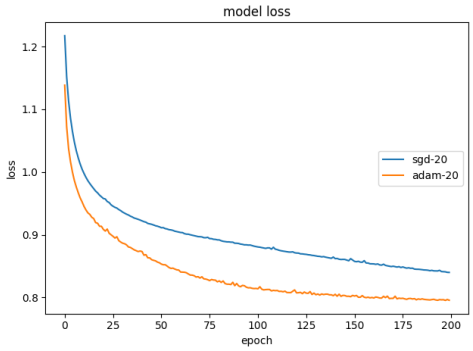
	SGD	Adam
Average Train Accuracy	0.61	0.64
Average Train Loss	0.88	0.81
Average Test Accuracy	0.60	0.63
Average Test Loss	0.89	0.81
95% confidence interval accuracy	(0.60, 0.65)	(0.61, 0.65)

Table 8, shows the average train and test accuracy and loss. We can average test and train accuracy rate is for Adam is bit higher than SGD whereas achieves better loss score.

Table 9: Adam vs SGD single hidden layer with 20 neurons



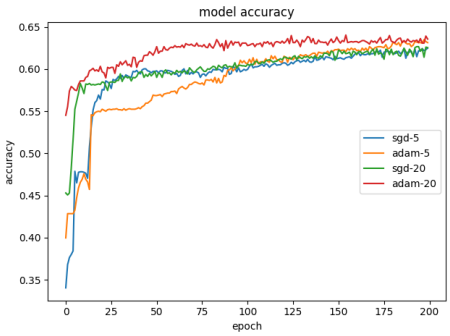
Adam vs SGD Accuracy Curve



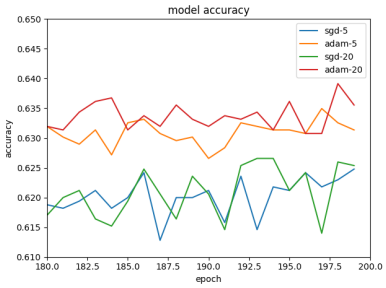
Adam vs SGD Loss Curve

In summary if we compare all the result together, and the table Table 9, clearly states that using Adam optimizer gives much better performance. Accuracy curve of Adam reaches to peak earlier than others and gives higher accuracy level than all others. Also the loss is much less than model using SGD. For the model using Adam optimizer with 20 neurons with one hidden layer, the 95% confidence interval accuracy is (0.61, 0.66), which means this model can predict 61% to 65% data correctly.

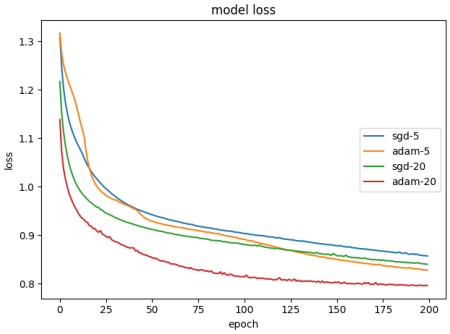
Table 10: Comparison of Adam and SGD with 5 and 20 hidden neurons for a single layer model



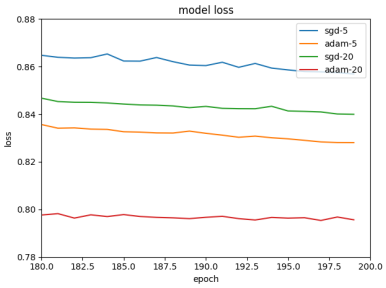
Adam vs SGD Accuracy Curve



Adam vs SGD Accuracy Curve Expanded



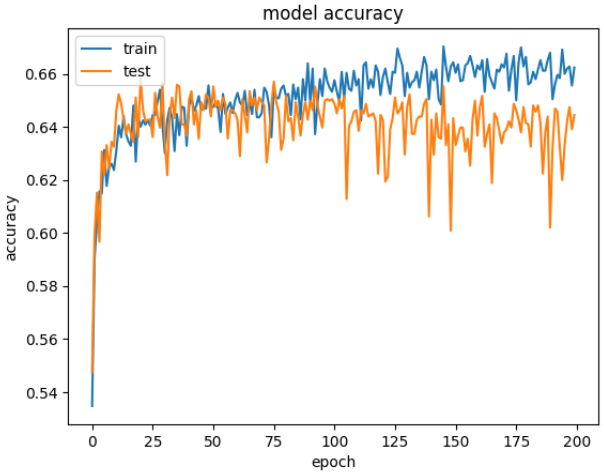
Adam vs SGD Loss Curve



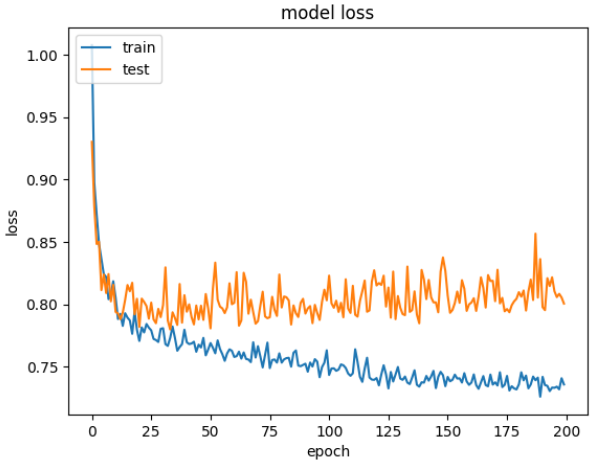
Adam vs SGD Loss Curve Expanded

Now if we consider adding another hidden layer in the models using Adam optimizer to see if the performance gets better. The result shows in Table 11.

Table 11: Models with Adam using multiple hidden layers



Accuracy curve for 2 hidden layers with 20 neurons



Loss curve for 2 hidden layers with 20 neurons

We can see that in our case adding multiple layers overfit the model hence 2 hidden layers with 20 neurons

using Adam optimizer would not a better fit for this model.

6.5 The Best Model: Evaluate The Best Model Based On The Test Dataset

6.5.1 Accuracy Curve and loss Curve

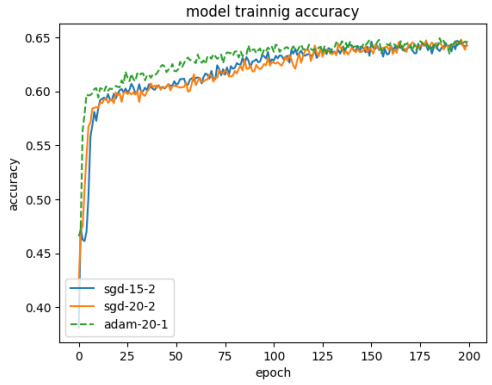
To evaluate the best model lets start evaluating the accuracy curve and loss curve for the three models that performed well based on our earlier experiments - (1) model with 2 hidden layers and 15 hidden neurons in each layer with SGD optimizer, (2) model with 2 hidden layers and 20 hidden neurons in each layer with SGD optimizerand (3) model with single hidden layer and 20 hidden neurons with Adam optimizer.

Table 12, shows the training accuracy and the training loss for all the three models stated above. It states that the model with Adam optimizer training accuracy is faster and higher than the other two models with SGD optimizer. The training accuracy curve, represents the model's training accuracy over time.

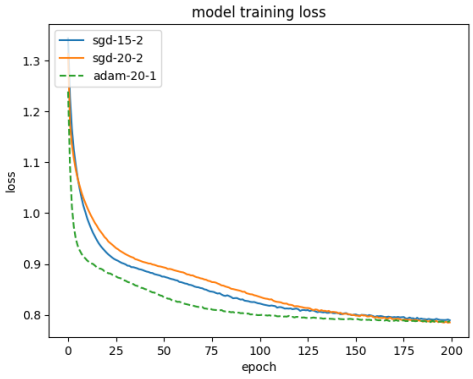
Table 13, shows both the loss curve and the corresponding accuracy curve. The accuracy curve, represents the model's accuracy of correctly classified instances over time. As the accuracy curve rises, it signifies that the model is making more correct predictions, thereby enhancing its overall performance. Hence from the curve we can see that the model with Adam optimizer is performing better.

The loss curve shows the values of the model's loss over time. Initially, the loss is high and gradually decreases, indicating that the model is improving its performance. A decrease in the loss value suggests that the model is making better predictions, as the loss represents the error or misclassification between the predicted output and the true output. Therefore, a lower loss indicates better performance. So from the curves we can see that the model with Adam optimizer with 20 hidden neurons with a single layer model showing better performance.

Table 12: Training curve for multi layer model with SGD vs single layer model with Adam optimizer

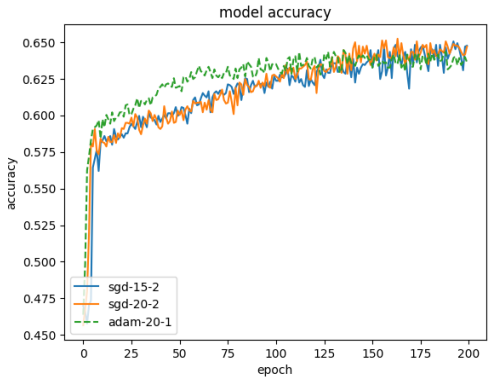


Training accuracy curve for single layer Adam and multi layer SGD models

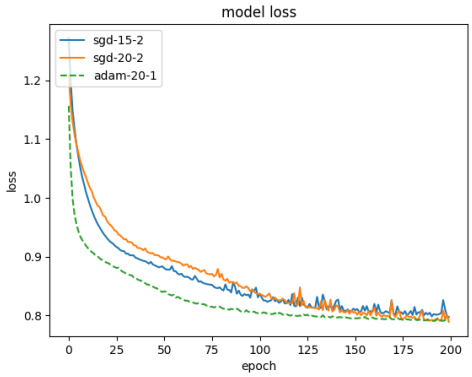


Training loss curve for single layer Adam and multi layer SGD models

Table 13: Accuracy and loss curve for multi layer model with SGD vs single layer model with Adam optimizer



Accuracy curve for single layer Adam and multi layer SGD models



Loss curve for single layer Adam and multi layer SGD models

Comparison for the best model:

Table 14: Best model comparison

Optimizer	**Adam**	**SGD**	**SGD**
Number of hidden layers	1	2	2
Number of hidden neurons (layer 1 and 2)	20	15	20
Model parameter	304	**469**	**724**
Mean test accuracy	0.64	0.65	0.64
Mean test loss	0.80	0.80	0.80
95% confidence interval accuracy	(0.61, 0.66)	(0.62, 0.67)	(0.62, 0.67)

In Table 14, we see a very close comparison among the three models. Models with SGD optimizer with 2 hidden layers and 15/ 20 neurons getting 62% ~ 67% prediction correctly where as the model with Adam optimizer single layer with 20 neurons predicting very close to 61% ~ 66% data correctly.

6.5.2 F1 Score Comparison

The F1 score is a metric used to evaluate the performance of a classification model. It is a combination of precision and recall and provides a balance between these two metrics. F1 score provides a balance between precision and recall, making it suitable for imbalanced datasets. There is often a trade-off between precision and recall because increasing one may result in a decrease in the other, and the F1 score helps to set a balance.

Table 15: Performance comparison of models

Model: Optimizer = Adam, Hidden layer = 1, Hidden neurons = 20

	precision	recall	f1-score	support
0	0.83	0.70	0.76	336
1	0.61	0.79	0.69	757
2	0.56	0.42	0.48	474
3	0.57	0.27	0.37	104
accuracy			0.64	1671
macro avg	0.64	0.55	0.57	1671
weighted avg	0.64	0.64	0.62	1671

Model: Optimizer = SGD, Hidden layer = 2, Hidden neurons = 30

	precision	recall	f1-score	support
0	0.79	0.72	0.76	336
1	0.63	0.77	0.70	757
2	0.57	0.48	0.52	474
3	0.67	0.25	0.36	104
accuracy			0.65	1671
macro avg	0.67	0.56	0.58	1671
weighted avg	0.65	0.65	0.64	1671

Model: Optimizer = SGD, Hidden layer = 2, Hidden neurons = 40

	precision	recall	f1-score	support
0	0.79	0.75	0.77	336
1	0.64	0.77	0.70	757
2	0.56	0.47	0.51	474
3	0.56	0.24	0.34	104
accuracy			0.65	1671
macro avg	0.64	0.56	0.58	1671
weighted avg	0.64	0.65	0.64	1671

From the above result tables we see model with SGD optimizer with 2 hidden layers 30 and 40 neurons both got same F1 score of 0.58 whereas for the model with Adam optimizer we got F1 score of 0.57. The accuracy score is also same for the models with SGD optimizer - 0.65 but the one with Adam got accuracy score of 0.64. Now if we consider precision then model with 30 hidden neurons giving us average precision of 0.67 which is higher than other models. Comparing the results if we consider both F1 score and precision, we can say the model with 2 hidden layers, 30 hidden neurons and with SGD optimizer is performing better.

6.5.3 Confusion Matrix

Fig 6, shows the confusion matrix for the model using SGD optimizer and 2 layers with 20 hidden neurons each. From the matrix data we can see that - 241 instances that belong to class 0 and are correctly predicted. There are 61 instances of class 1, 6 instance of class 2 are incorrectly predicted as class 0. 621 instances are correctly predicted as class 1, where as 95 instances of class 0, 275 instances of class 2 and 25 instances of class 3 are incorrectly predicted as class 1. For class 2, 183 instances are true positive and 73 instances of class1, 56 instance of class 3 are false positive. Lastly, for class 3, 23 instances are correctly classified and 2 instances of class 1 along with 10 instances of class 2 are incorrectly classified as class 3.

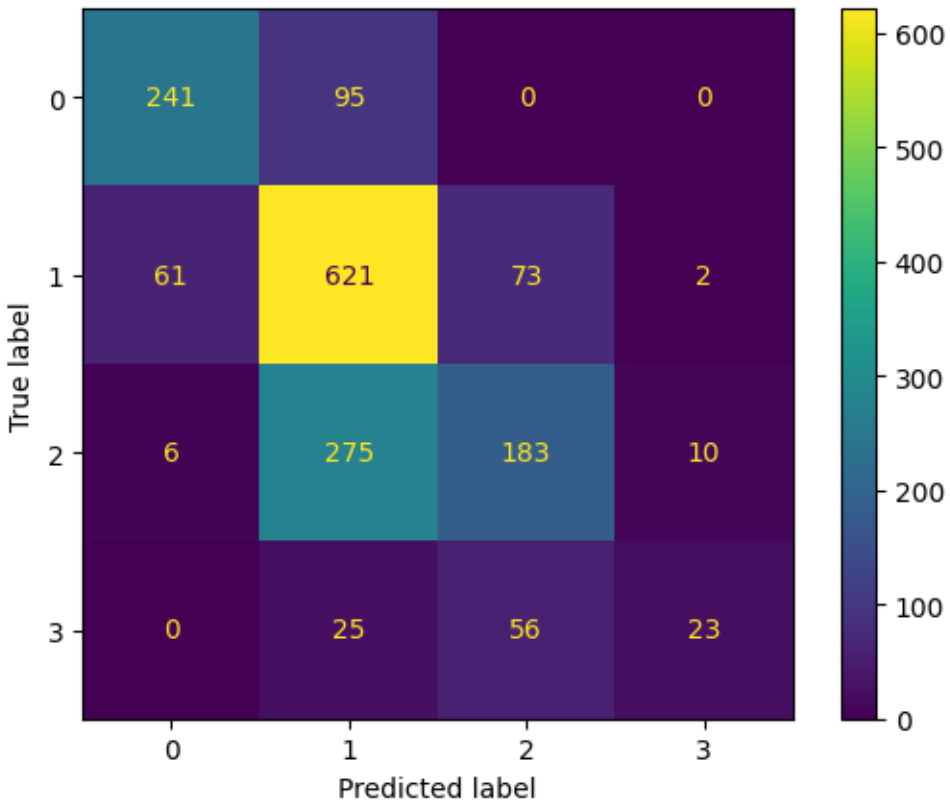


Figure 6: Confusion matrix for the model using SGD optimizer and 2 layers with 20 hidden neurons each

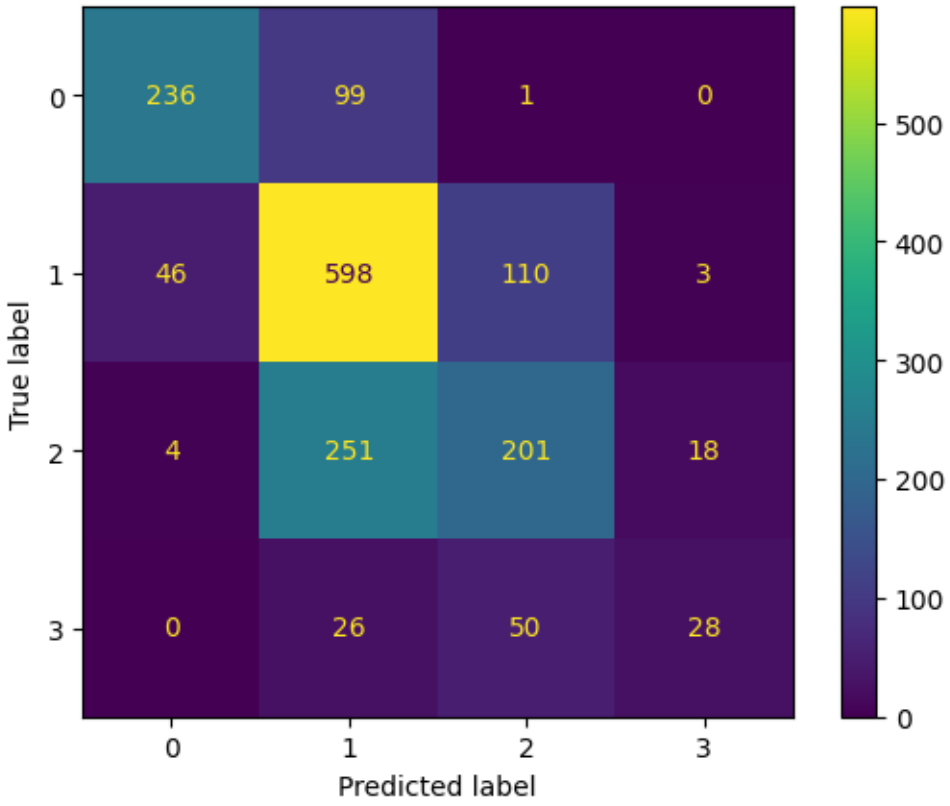


Figure 7: Confusion matrix for the model using Adam optimizer with single layer and 20 hidden neurons

6.5.4 ROC/AUC Curve

The ROC curve is a plot of the true positive rate (sensitivity or recall) against the false positive rate for different threshold values. Each point on the ROC curve corresponds to a different threshold, and the curve provides a visual representation of the trade-off between sensitivity and specificity. The AUC is the area under the ROC curve. It is a single scalar value that summarizes the overall performance of the model across different threshold. The ROC/AUC provides a comprehensive view of a model's ability to discriminate between classes without being sensitive to the specific threshold chosen for classification, higher ROC/AUC values are desirable (@roc2023jason). We have gathered the ROC/AUC score for each class among the three comparing models in Table 16.

Table 16: Comparison of ROC AUC OvR Score

Target Class	Class Label	ROC AUC OvR Score Optimizer: Adam Hidden neurons: 40 Hidden Layer: 1	ROC AUC OvR Score Optimizer: SGD Hidden neurons: 30 Hidden Layer: 2	ROC AUC OvR Score Optimizer: SGD Hidden neurons: 40 Hidden Layer: 2
Age Class 1	0	0.9482	0.9442	0.9435
Age Class 2	1	0.7761	0.7721	0.7725
Age Class 3	2	0.7900	0.7874	0.7905
Age Class 4	3	0.8887	0.8953	0.8879
Avg ROC AUC OvR		**0.8507**	**0.8498**	**0.8486**

Comparing the scores in Table 16, tells us that the model using Adam optimizer with single hidden layer and 20 hidden neurons in each, scores better. Fig 8 and Fig 9, shows the plotting for the ROC/AUC curve for different classes.

7 Conclusion

In this project we examine the Abalone dataset to determine the abalone age using linear regression machine learning technique using different sets of input features. In conclusion we notice that with more input features and appropriate normalization it would have been possible to determine more accurate age. However, our preliminary investigation indicates that linear regression is only suitable for a small set of input features and therefore the error estimates can be higher in comparing to use other advance machine learning techniques that we aim to use in the future.

8 Reference

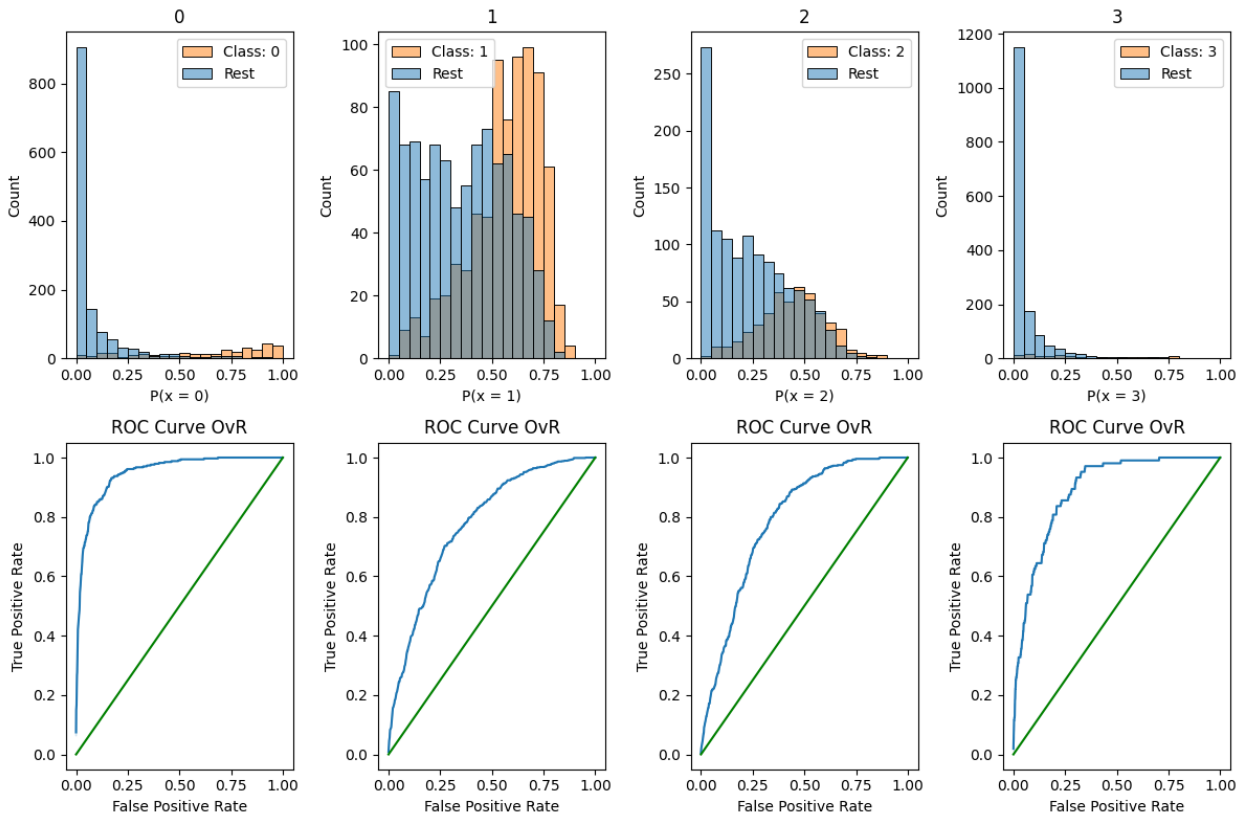


Figure 8: ROC/AUC curve for the model with Adam optimizer with single layer and 20 hidden neurons

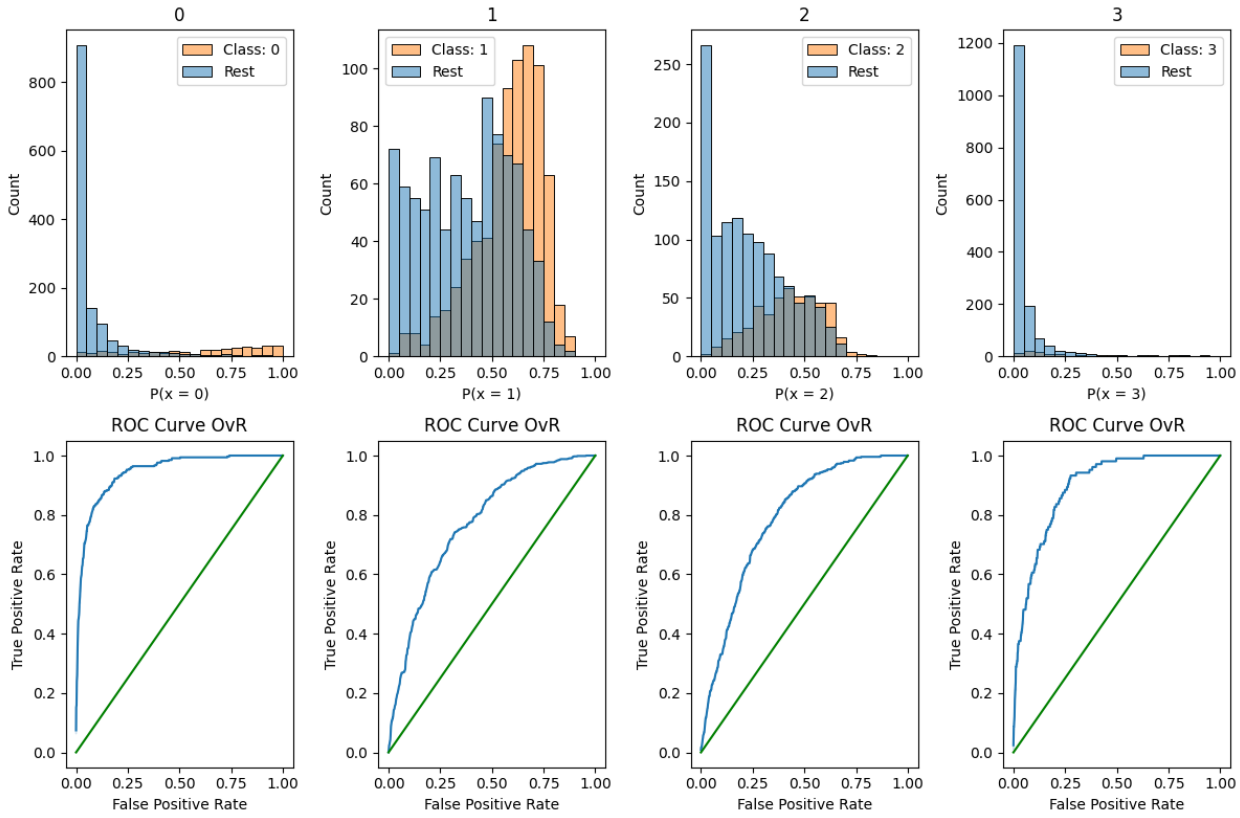


Figure 9: ROC/AUC curve for the model with SGD optimizer with 2 layers and 15 hidden neurons