

Group Portion of Final Report

Katana Guard, Lubaba Sheikh, Dawson Alexander, Max Pedroza, Deanne Savard

Github Repository Link: https://github.com/LubabaSheikh/BMEN415_Project#bmen415_project

Introduction

The following report explores multiple models corresponding to a classification dataset, regression dataset, and image input classification dataset. The regression dataset regards the use of volumetric features of the brain to predict the age of a healthy patient. The classification dataset takes into account input features such as pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function and age in order to separate patients with diabetes from those without. The image input classification dataset uses thousands of chest x-ray images of patients with pneumonia and patients without, to train a model classifying patients as having pneumonia or not.

For the Regression dataset the group came up with a Partial Least Squares model, Neural Net, Ordinary Least Squares model, 3 Random Forests, a Lasso model, Decision Tree, 2 KNN (1 with 12 neighbors and 1 with 10 neighbors), Bayesian Ridge Regression model, Linear Discriminant Analysis, Stochastic Gradient Descent, Regression Tree, and Multiple Linear Regression model.

For the Classification dataset the group used a CART model, 3 Neural Nets, Linear Discriminant Analysis, 2 Logistic Regression Classification, 2 Naive Bayes, Voting Classifier, 3 KNN (1 with 23 neighbors, 1 with 12 neighbors, and 1 with 33 neighbors), Decision Tree, and a Random Forest model. For the Image Input Classification problem we used a Convolutional Neural Network.

We hypothesize that the neural net methodology will work well with both classification and regression, because it is highly adaptable and can be optimized for many different problems. The tree based regression models will not work well, because tree methodology is intrinsically better for classification. The SGD and PLS models will also work poorly for our data, because our regression data is not collinear.

Evaluation of Models

a. Training Regime

Each dataset was split into 80% training data and 20% test data for each model in both regression and classification cases. Both the Regression dataset and the Classification dataset were preprocessed using a function called “StandardScaler”. It arranges the data into a standard normal distribution with a mean of 0 and standard deviation of 1. Ideally this should increase the accuracy of each model since it will stop features with a larger range from having more influence on the output than those with a smaller range [1]. The Regression dataset was further preprocessed by eliminating any unrelated or non-contributing data from the input features. This included “Left-WM-hypointensities”, “Right-WM-hypointensities”, “Left-non-WM-hypointensities”, and “Right-non-WM-hypointensities” all of which correspond to an

unchanging column of zeros and therefore does not impact the models. “Dataset” and “S.No” were also removed as they should be considered unrelated to the output prediction.

The image input data had three sets of data: a validation, training, and a testing set. Within these sets, the pictures are further organized into one of two groups, ‘NORMAL’ or ‘PNEUMONIA’, referred to as subsets. Upon importing the data, we perform a rescaling on the greyscale of the image to obtain a value between 0 and 1 for each pixel. We also generate new data by zooming into the pictures a little bit, and by reflecting the images horizontally. This generation makes our training more resistant to changes in angle of view and to flips of the image whether that occurred due to software or due to situs inversus.

To create the model, we use multiple layers to create convolution kernels, max pooling for 2D data, flattening of the input, and dropouts to randomly assign 0 to units. In total, this model has 23 037 505 trainable parameters. By using a convolutional neural network, we are able to take advantage of the spatial organization of the pixels.

b. Regression Models

As you can see from Table 1, the regression tree methods used had the best performances of all the model types with the simple regression tree and the random forest methods having the two highest performances. As expected, the PLS, decision tree, and stochastic gradient descent method had low R² values. In Figure 4 (right), you can see that the data does not follow a continuous distribution and is discretized into smaller blocks. This is expected from the decision tree model which is typically used for classification, but can be used for regression. The PLS model also had a low R² value, because PLS models work best with collinear data. From Figure 1 (left), you can see that the data is not strictly linear and follows a curved path.

Another problem we ran into during our simulations was overfitting. In some cases, the model overfit to the training data, but got much lower R² values for the testing data in comparison. From Table 1, you can see that the NN, Random Forest, Decision Tree, Random Forest 2, regression tree, and random forest 3 are all overfit to the training data and have an R² performance that is over 10% different between the training and testing data. The NN has the highest difference of 19%. As you can see from Figure 2, the training data is much closer to the $x=y$ line while the testing data is more dispersed around it. Overfitting is a common problem that needs to be considered when developing regression and classification models. Overall, the OLS shows the best performance with our data with no overfitting issues. The KNN models also showed good R² values with no overfitting. The KNN with the higher K value of 12 had a better performance over the KNN model with the K value of 10. However, it is important to not overfit your data by using a K value that is not too high. In this case, neither KNN models are overfit. As you can see in Figure 7, the training y predictions and testing y predictions have a very similar scatter.

Table 1. Regression performance criteria found using 15 different regression models.

		Testing Data		Training Data	
	Model Type	R2	RMSE	R2	RMSE
1	Partial Least Squares	0.77113	9.68893	0.76185	9.76358
2	Neural Net	0.80698	8.89765	0.99389	1.5641
3	Ordinary Least Squares	0.85605	7.68393	0.86049	7.47277
4	Random Forest	0.87848	7.06007	0.98028	2.80977
5	Lasso	0.8383	8.14387	0.8341	8.14921
6	Decision Tree	0.7815	9.46682	0.91291	5.90443
7	KNN (K=12)	0.84101	7.98048	0.85788	7.56734
8	Bayesian Ridge	0.84657	7.83984	0.85594	7.61881
9	Random Forest	0.85818	7.53724	0.98033	2.8151
10	LDA	0.84458	7.98417	0.8929	6.54739
11	Stochastic Gradient Descent	0.74728	10.1811	0.80346	8.86968
12	Regression Tree	0.87859	7.05654	0.98028	2.80892
13	Random Forest	0.84984	7.36719	0.97209	3.33298
14	KNN (K = 11)	0.83808	8.07177	0.86355	7.4104
15	Multiple Linear Regression	0.84924	7.81256	0.86383	7.39757

*Root Mean Square Error

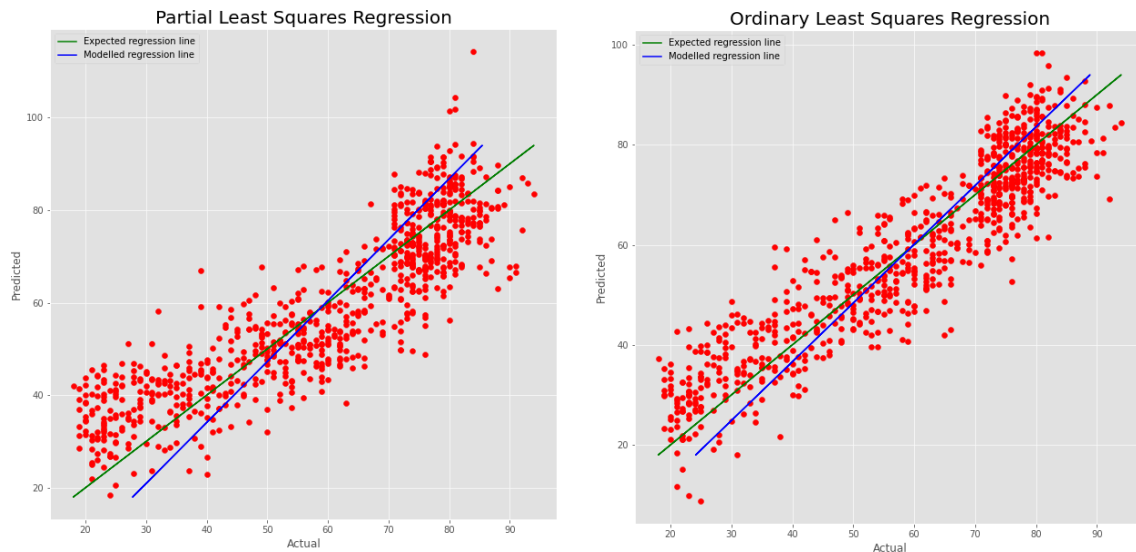


Figure 1. Model 1 (left) and Model 3 (right) showing the expected regression line plots y_{test} vs y_{test} (green line) and the modeled regression line plots y_{test} vs $y_{\text{pred_test}}$ (blue line). The blue line corresponds to the red scatter.



Figure 2. Model 2 neural network plot of $y_{\text{predicted}}$ versus y for both the testing (blue scatter) and training (orange scatter) data.

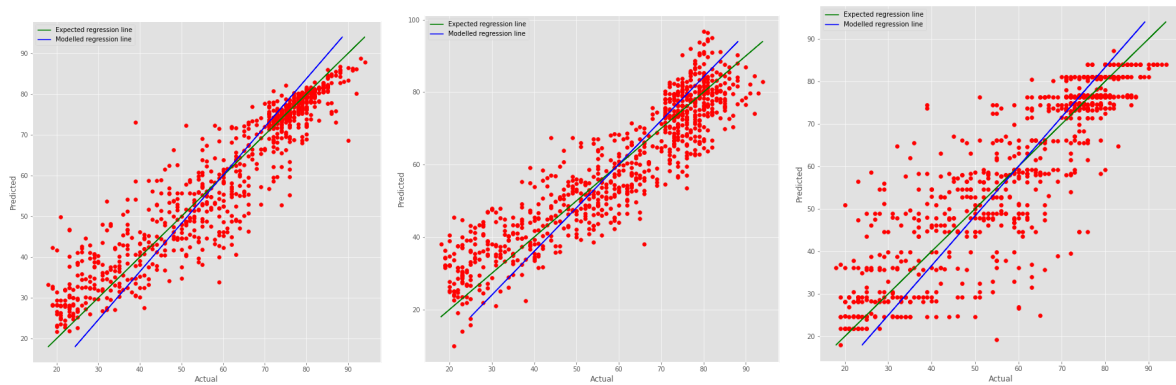


Figure 4. Expected regression line plots y_{test} vs y_{test} and the modeled regression line plots y_{test} vs $y_{\text{pred_test}}$ using random forest(left), lasso(center), and decision tree(right) regression.



Figure 5. Expected regression line plots y_{test} vs y_{test} and the modeled regression line plots y_{test} vs $y_{\text{pred_test}}$ using KNN (left), Bayesian Ridge(center), and Random Forest(right) regression.

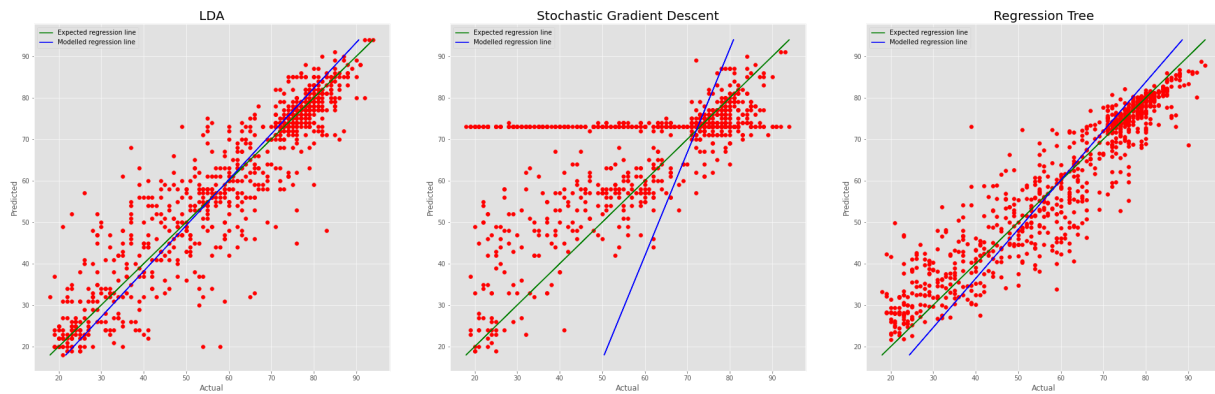


Figure 6. Expected Regression line plots and the modeled regression line plots using LDA (left), Stochastic Gradient Descent (middle), and Regression Tree (right)

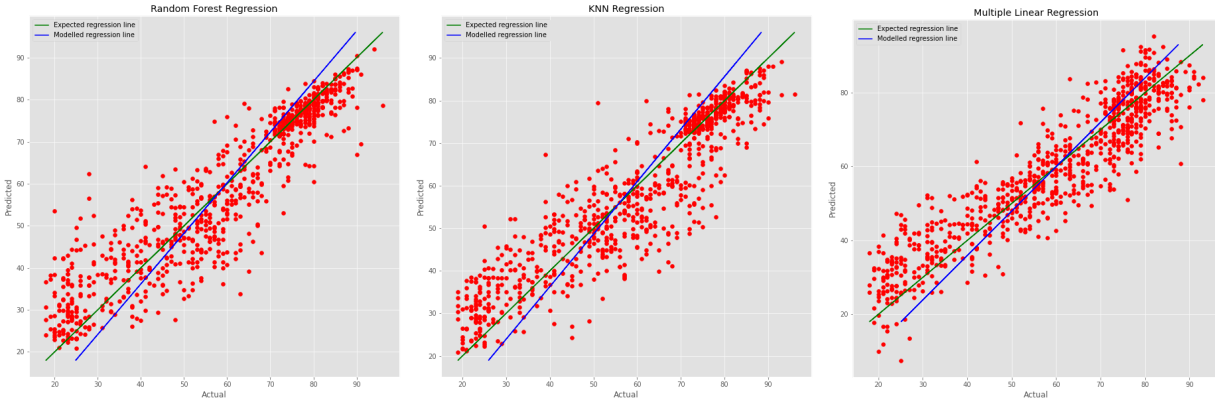


Figure 6. Expected Regression line plots and the modeled regression line plots using Random Forest Regression (left), KNN Regression (middle), and Multiple Linear Regression (right)



Figure 7. Model KNN=12 plot of $y_{\text{predicted}}$ versus y for both the testing (blue scatter) and training (orange scatter) data.

c. Classification Models

From Table 2, you can see that the CART model had the lowest accuracy of all the models. However, the CART model is still popular because it is simple to understand and gives very intelligible class split representation. In Figure 11, you can see how the flow chart for the CART model can be used by anyone even without a computer once the model is initialized. For instance, the LDA model has a higher accuracy, but is difficult to visualize without lowering the dimensionality of the problem. Histograms are a popular choice to use to visualize classification models after using eigenvector calculations to lower

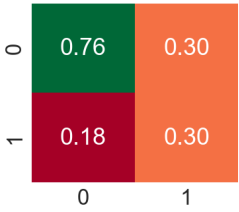
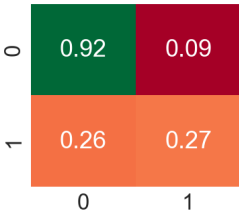
dimensionality. However, after this is done, you cannot see the features of your classification model. In Figures 12 and 13, you can see a 2D and 3D representation of the class scatters. However, you can only see 2 and 3 of your features represented.

Neural nets are a popular choice for regression and classification problems due to their adaptability. From Figures 9 and 10, you can see that the architecture for each NN is very different and can be specialized for a given problem to achieve better results. However, it takes a certain understanding about NN performance and creativity to fully optimize these problems. Here, it is good enough that the NN's are giving similar accuracy as the other models.

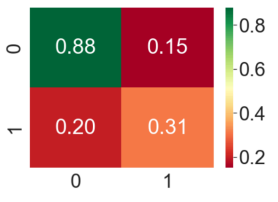
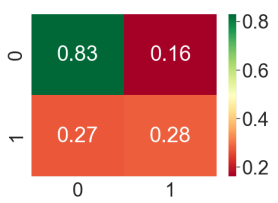

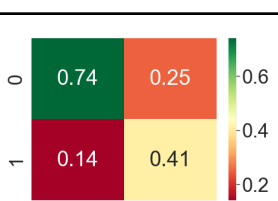
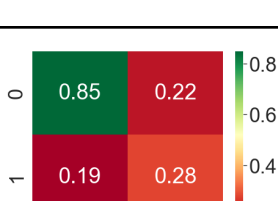
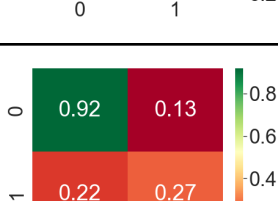
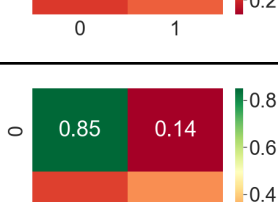
Again there are some overfitting issues with the CART, decision tree, and random forest methods used. These models have been shown to be more susceptible to overfitting than other models. The Model 13 NN is also overfit to an extent. Since the NN model used in the regression section was also overfit, it is shown to be an issue with NN's as well. It is important to look out for when modeling NN's, because it is uncertain whether your model will be overfit or not, depending on your architecture.

In this classification problem, we are looking for a high sensitivity as opposed to a high accuracy. This is because it can be dangerous if a patient tests negative for diabetes but is actually positive. It is better to have more false positives and, therefore, a lower specificity and accuracy. This is usually true for biomedical classification problems. From Figure 8, you can see that the second Neural Net model gives the highest sensitivity along with the Naive Bayes model. However, the Naive Bayes method also has a higher accuracy and specificity than the NN model, so it is better for this problem. Theoretically, we could add a bias to our NN and other models to accept more false positives and less false negatives. This would make the performance of our models better for this type of classification problem.

Table 2. Performance parameters used to access our classification models

		Testing Data		Training Data
	Model Type	Confusion Matrix	Accuracy	Accuracy
1	CART		0.68831	1
2	Neural Net		0.75974	0.76058

3	Linear Discriminant Analysis	<div><div><div>0</div><div>0.87</div><div>0.11</div></div><div><div>1</div><div>0.24</div><div>0.32</div></div><div><div>0</div><div>1</div></div></div>	0.75324	0.78339
4	Logistic Regression Classification	<div><div><div>0</div><div>0.91</div><div>0.08</div></div><div><div>1</div><div>0.25</div><div>0.30</div></div><div><div>0</div><div>1</div></div></div>	0.78571	0.77687
5	Naive Bayes	<div><div><div>0</div><div>0.85</div><div>0.14</div></div><div><div>1</div><div>0.23</div><div>0.32</div></div><div><div>0</div><div>1</div></div></div>	0.75974	0.7671
6	Voting Classifier	<div><div><div>0</div><div>0.89</div><div>0.10</div></div><div><div>1</div><div>0.24</div><div>0.31</div></div><div><div>0</div><div>1</div></div></div>	0.77922	0.82084
7	KNN (K=23)	<div><div><div>0</div><div>0.92</div><div>0.11</div></div><div><div>1</div><div>0.20</div><div>0.31</div></div><div><div>0</div><div>1</div></div><div><div>0.75</div><div>0.50</div><div>0.25</div></div></div>	0.7987	0.77524
8	Decision Tree	<div><div><div>0</div><div>0.83</div><div>0.20</div></div><div><div>1</div><div>0.19</div><div>0.32</div></div><div><div>0</div><div>1</div></div><div><div>0.8</div><div>0.6</div><div>0.4</div><div>0.2</div></div></div>	0.74675	1

9	Random Forest	 <table><tr><td>0</td><td>0.88</td><td>0.15</td></tr><tr><td>1</td><td>0.20</td><td>0.31</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.88	0.15	1	0.20	0.31		0	1	0.77273	1
0	0.88	0.15											
1	0.20	0.31											
	0	1											
10	KNN (K=12)	 <table><tr><td>0</td><td>0.83</td><td>0.16</td></tr><tr><td>1</td><td>0.27</td><td>0.28</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.83	0.16	1	0.27	0.28		0	1	0.72077	0.78338
0	0.83	0.16											
1	0.27	0.28											
	0	1											
11	Naive Bayes	 <table><tr><td>0</td><td>0.79</td><td>0.20</td></tr><tr><td>1</td><td>0.16</td><td>0.39</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.79	0.20	1	0.16	0.39		0	1	0.76623	0.75244
0	0.79	0.20											
1	0.16	0.39											
	0	1											
12	Neural Net (8, 8, 8)	 <table><tr><td>0</td><td>0.74</td><td>0.25</td></tr><tr><td>1</td><td>0.14</td><td>0.41</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.74	0.25	1	0.14	0.41		0	1	0.74675	0.7899
0	0.74	0.25											
1	0.14	0.41											
	0	1											
13	Neural Net (16,16,1)	 <table><tr><td>0</td><td>0.85</td><td>0.22</td></tr><tr><td>1</td><td>0.19</td><td>0.28</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.85	0.22	1	0.19	0.28		0	1	0.73377	0.88274
0	0.85	0.22											
1	0.19	0.28											
	0	1											
14	KNN (K=33)	 <table><tr><td>0</td><td>0.92</td><td>0.13</td></tr><tr><td>1</td><td>0.22</td><td>0.27</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.92	0.13	1	0.22	0.27		0	1	0.77273	0.77036
0	0.92	0.13											
1	0.22	0.27											
	0	1											
15	Logistic Regression Classification	 <table><tr><td>0</td><td>0.85</td><td>0.14</td></tr><tr><td>1</td><td>0.23</td><td>0.32</td></tr><tr><td></td><td>0</td><td>1</td></tr></table>	0	0.85	0.14	1	0.23	0.32		0	1	0.75974	0.78990
0	0.85	0.14											
1	0.23	0.32											
	0	1											

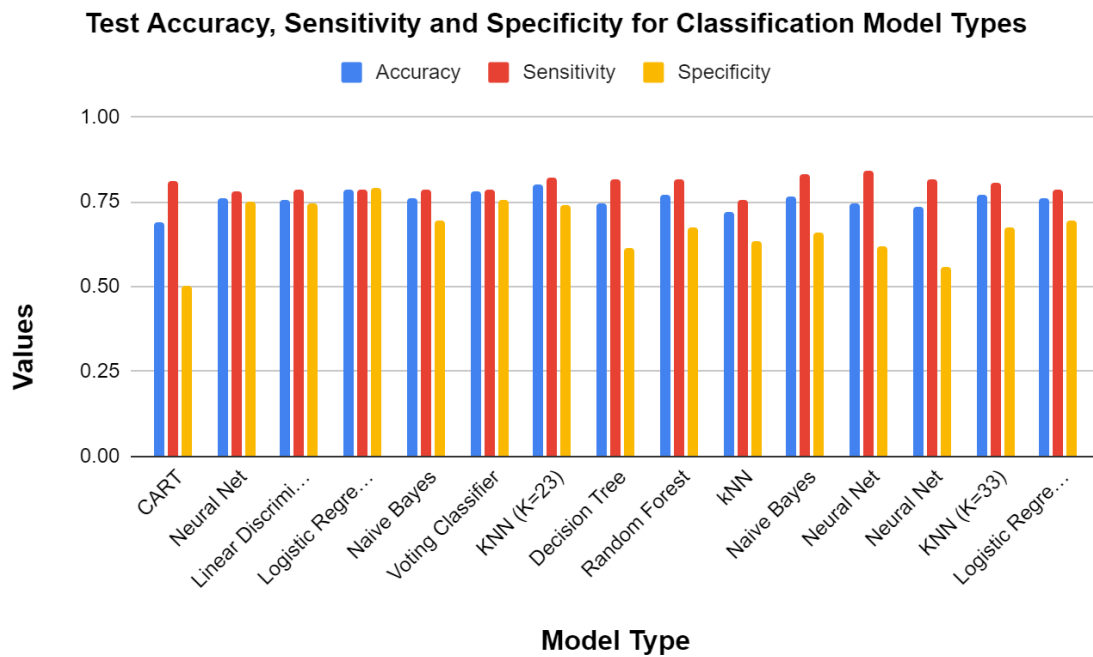


Figure 8. Bar graph showing additional performance parameters for our classification models.

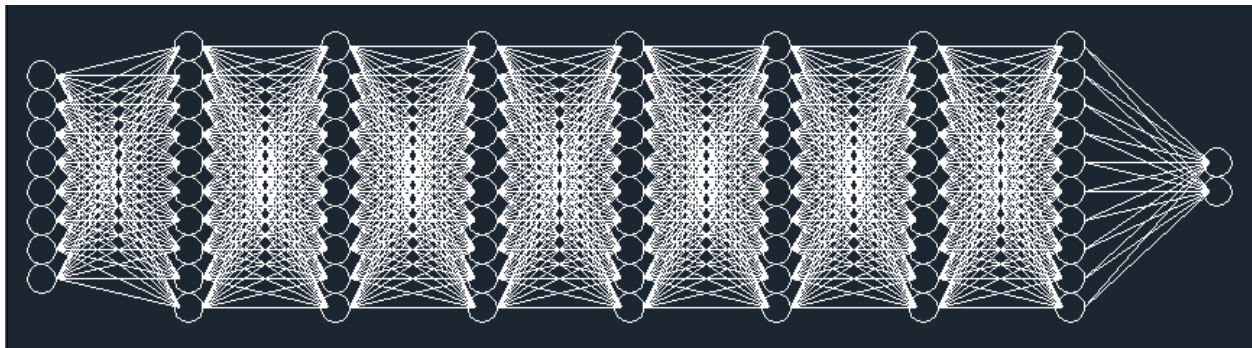


Figure 9. Model 2 neural net architecture showing 6 input features, 7 densely connected hidden layers, and 2 outputs. There are 10 neurons in each layer. A quasi-newton optimizer was used with Relu activation functions.

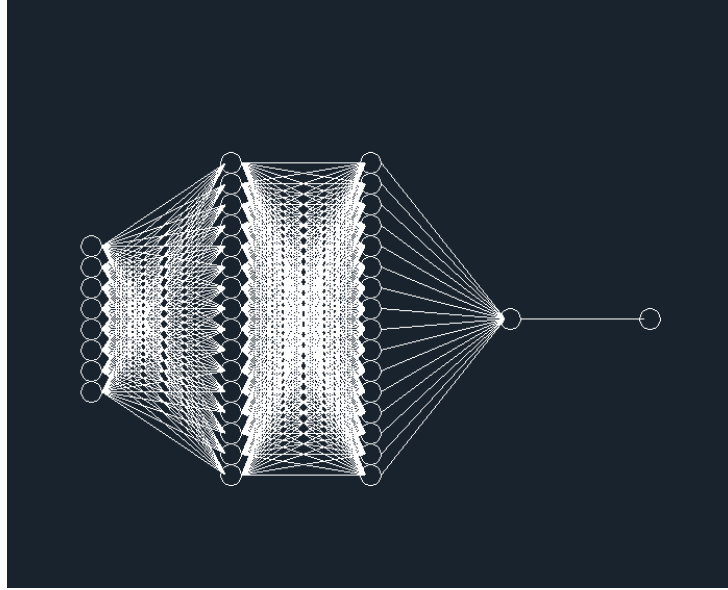


Figure 10. Model 12 neural net architecture showing 3 hidden layers [16,16,1]

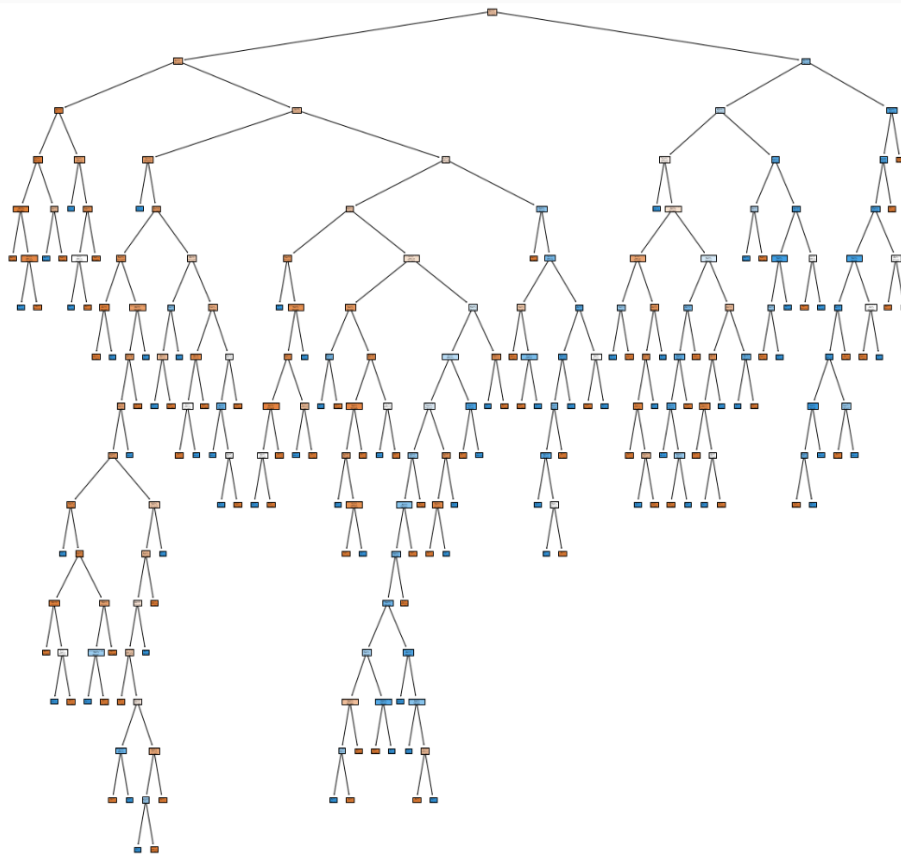


Figure 11. Model 1 diagram of the framework used in the CART classification model where the colour of the node indicates the density of the population at that node.

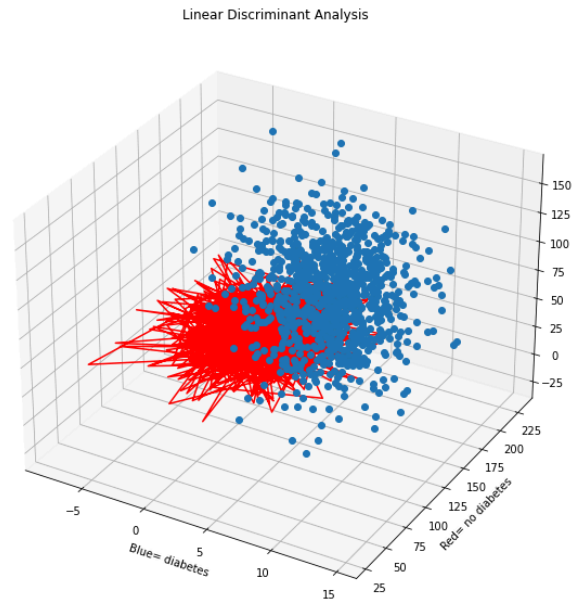


Figure 12. Model 3 diagram for LDA showing 2 classes in 3D.

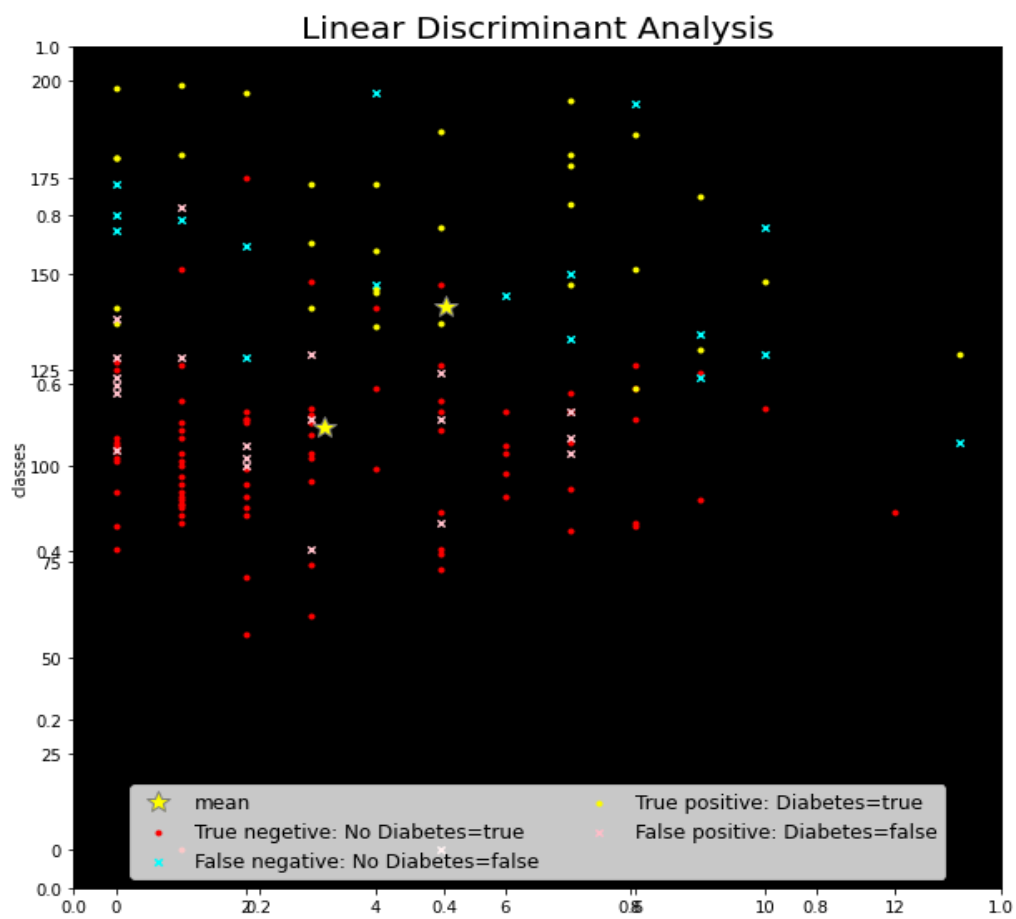


Figure 13. Model 3 scatter plot for linear discriminant analysis- 2D representation.

a. Image Classification Model

For the Image Classification Model, we adapted Pham Cao's code from Kaggle [2] to suit our needs. Due to constraints with time and hardware we had access to, we only completed 35 epochs. Despite this, our performance would likely not have improved much with additional cycles because the most dramatic improvement occurs at the beginning. Figure 16 illustrates the progression of loss and accuracy with each iteration. 35 Epochs required just over three hours to complete for a test accuracy of 90%. According to the online post, a 92% test accuracy is obtainable for 100 epochs. Dedicating time to compute those additional 65 epochs would likely require an additional 6 hours or so.

Table 3. Loss and Accuracy Results for the Image Classification CNN Model

	Loss	Accuracy
Validation Data	0.26919	0.88939
Training Data	0.20667	0.92032
Test Data	0.29909	0.89903

This dataset had class imbalances. The Pneumonia class dominated the normal class as observed in figure 14. This did not limit our ability to train the CNN model but it did lead to an uneven distribution in the confusion matrix in figure 15. Due to the fact that there are more pneumonia cases, the model develops an opposition to diagnose pneumonia. It tries to compensate for the fewer number of normal cases. This phenomenon results in proportionally more pneumonia cases to be flagged as normal. For medical purposes, this is actually disadvantageous because it means we reduce the chance of true negatives. It is preferred to further examine a healthy patient than to send a sick patient home which unfortunately this model does not do.

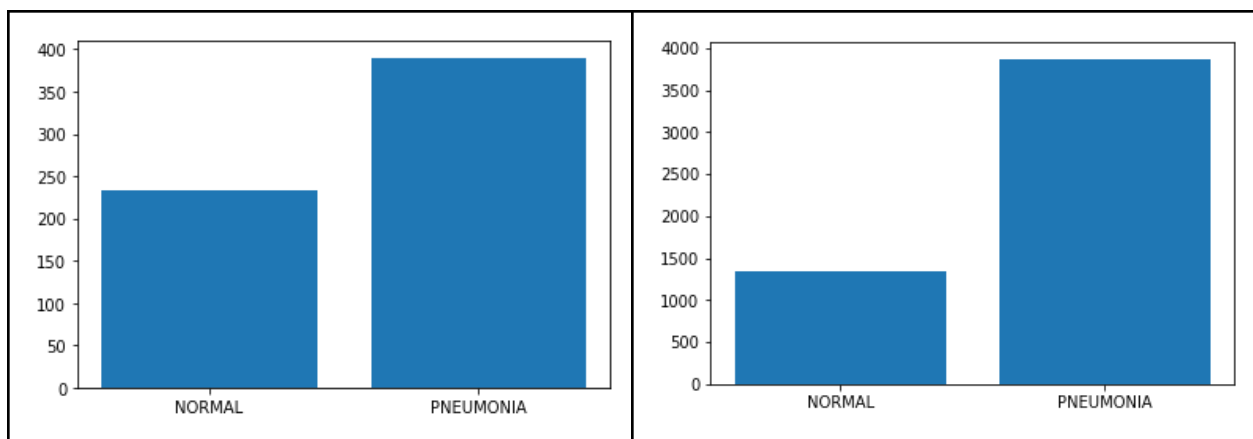


Figure 14. The Pneumonia class dominates both the testing data (left) and the training data (right)

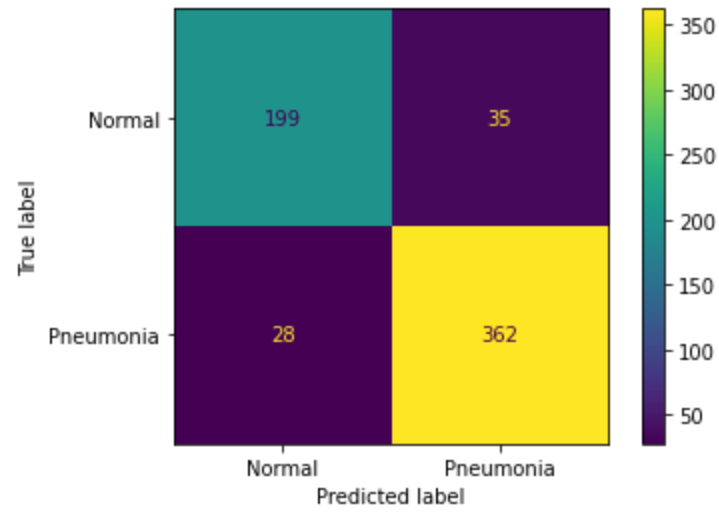


Figure 15. Confusion Matrix for the Image Classification CNN Model

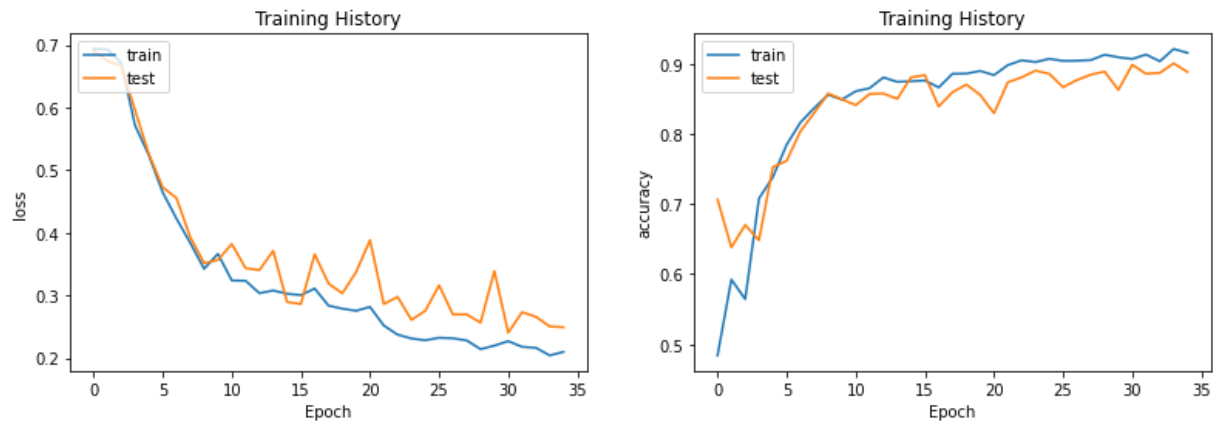


Figure 16. Loss and Accuracy Training History

Summary

Starting with the regression case study, the goal was to use regression modeling to predict the age of a patient based on the volumetric features of the brain. The regression tree methods used had the best performances of all the model types with the simple regression tree and the random forest methods having the two highest performances. Overfitting was a common problem and was a factor in 6 of the regression models.

For the classification case study, the balance between interpretability and performance was a key factor. While models such as CART were easy to see and interpret, models like the neural nets performed better.

The image processing case study was done with 35 epochs completed. The resulting 90% test accuracy was deemed satisfactory as it was approximated that an additionally 65 epochs would only lead to 92% accuracy. The imbalance in the dataset with the pneumonia class dominating lead to an uneven distribution in the confusion matrix. This inadvertently increased the accuracy of the model and reduced the likelihood of false negatives.

As a reflection, this project exposed the group to regression, classification, and image classification data sets and machine learning. The practical experience of treating the data and training the models was an invaluable learning experience and showed the complexity and versatility of machine learning applications.

Contributions:

Katana Guard: KNN model (K=12), Bayesian Ridge Regression model, and Random Forest model to the regression models, and the KNN model (K=23), Decision Tree and Random Forest model to the classification models, Regression plots, Introduction, and Training Regime.

Lubaba Sheikh: KNN (K=11) Regression, Random Forest Regression, Multiple Linear Regression, Neural Network (16,16,1), KNN (K=33) Classification, Logistic Regression, Regression plots, NN architecture, Sensitivity and Specificity graph, Classification Model analysis, image preprocessing

Dawson Alexander: Lasso, Random Forest, Decision Tree to regression models, and Logistic Regression Classification, Naive Bayes, Voting Classifier to classification models, and Summary.

Max Pedroza: Regression: Linear Discriminant Analysis, Stochastic Gradient Descent, Regression Tree Classification: K Nearest Neighbor (K=12), Naive Bayes, Neural Net (8, 8, 8)
Image model execution and adjustment and related writing as well as training regime.

Deanne Savard: Regression: Partial Least Squares, Ordinary Least Square, and Neural Net. Classification: Neural Net (model 2), Linear Discriminant Analysis, and CART. Writing: Classification, Regression, and Hypothesis.

References

[1] Robinson, S. (2021), “K-Nearest Neighbors Algorithm in Python and Scikit-Learn”. [Online]. Available at: <https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/> [Retrieved April 12th, 2022].

[2] Cao, P. (2022), “Diagnose Pneumonia by CNN (92% Testing)”. [Online]. Available at: <https://www.kaggle.com/code/caoofficial/diagnose-pneumoinia-by-cnn-92-testing/comments> [Retrieved April 12th 2022].