



# **Department of Biomedical Engineering**

## **Senor Systems and Data Analytics (BMEN 415) Winter 2022**

### **Final Project Report Group 12**

**Submitted: April 12<sup>th</sup>, 2022**

#### **Individual:**

**Mitchell Rhead 10185289**

#### **Team Members:**

**Carter McIlwraith 30091230**

**Davina Dang 30087063**

**Sadia Khan 30090271**

**Marc Vincent Raranggor 30090829**

## **Introduction and Image Classification Hypothesis**

For the regression dataset [1], we are using melanoma tumor size prediction. Melanoma is a type of skin cancer that occurs within the pigment-making cells called melanocytes. Melanoma tumors can be found in various different sizes, shapes, and colours. Using the mass of the area, size of the area, the ratio of normal malign, the size of the damaged area, total exposed area, the standard deviation of malign skin, and the error in malign skin measurements, an accurate prediction of tumor size can be determined. The target of this dataset is to predict the size of a melanoma tumor using the attributes above.

To complete the classification dataset [2], we are using the data from a study that predicts diabetes occurrence. Features of the data include the number of times one was pregnant, plasma glucose concentration, diastolic blood pressure (mm Hg), triceps skinfold thickness (mm), 2-hour serum insulin ( $\mu\text{U/ml}$ ), body mass index ( $\text{kg/m}^2$ ), diabetes pedigree function and age (years). The target of this data is to classify if an individual has diabetes or not.

In the image classification dataset [3], we are using a dataset of various ocular diseases. This dataset is meant to represent a “real-world” application of image classification. The attributes of the dataset consist of Normal (N), Diabetes (D), Glaucoma (G), Cataract (C), Age-related Macular Degeneration (A), Hypertension (H), Pathological Myopia (M), and Other (O). The target of the dataset is to accurately diagnose the patients' ocular disease based. To classify the images, we decided to use TensorFlow and develop our own Convolution Neural Network model (CNN). CNNs have a wide range of applications, however, are mostly used for image analysis as they have the ability to develop a representation of the 2D images being studied. This feature allows for CNNs to identify the position and the scale in various different structures in the images. This is very useful for our purposes as the identifiers for the different ocular diseases are in various different positions and scales. This makes CNN an optimal choice for this task. In theory, using a convolutional neural network model should allow us to be able to accurately predict and identify the different ocular diseases within the data set.

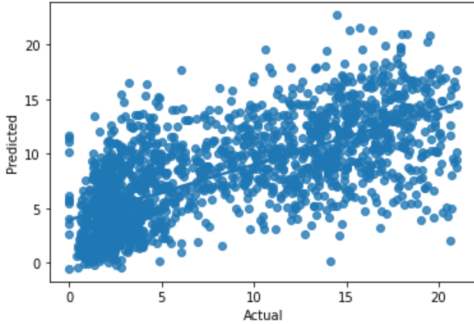
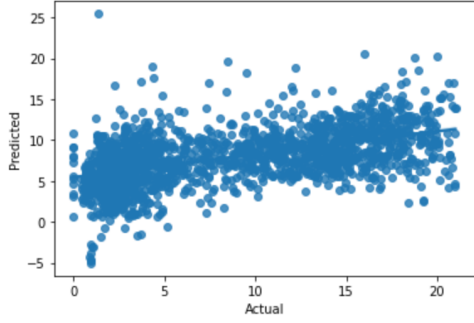
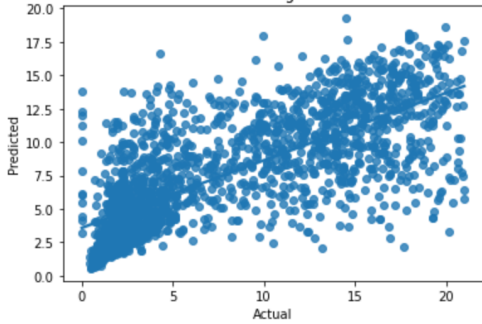
## **Individual Hypotheses**

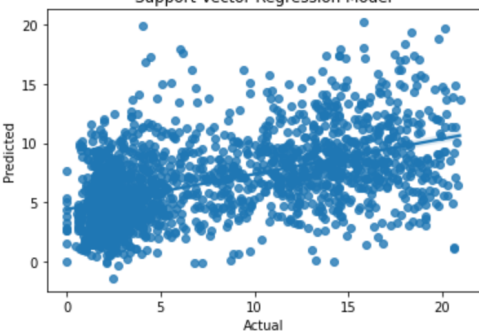
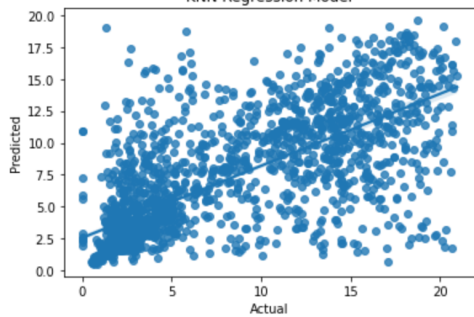
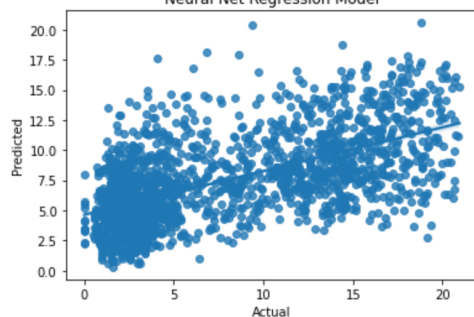
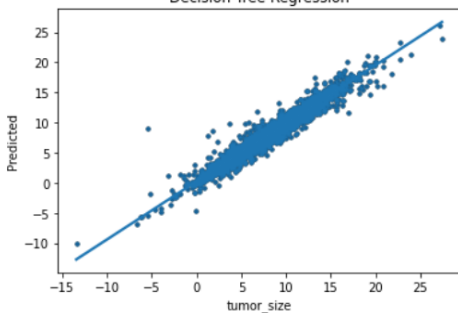
For the classification dataset, I have chosen to implement Decision Tree, KNN, and Naive Bayes models. Decision trees tend to require little data preparation, but are poor at extrapolating information [4]. I may have some issues with overfitting, but with 9 features I don't believe this will be a major issue. Regarding KNN, the classification dataset is not too large, so KNN may be ideal for the size of data [5]. There are 9 total features to the data; KNN performs best with small dimensions, so with 9 features I may see a decline in its performance. With Naive Bayes, it is able to handle higher dimensions [6], however it also assumes that all the features are independent. I anticipate the Naive Bayes to perform best, followed by KNN.

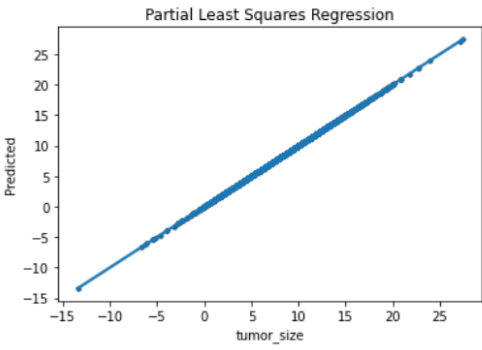
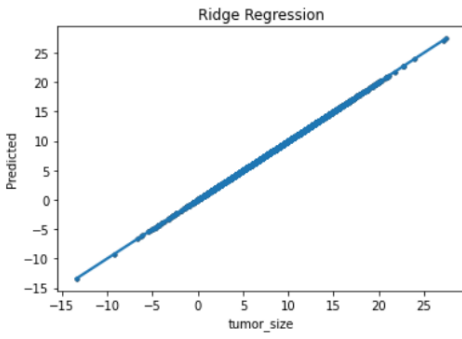
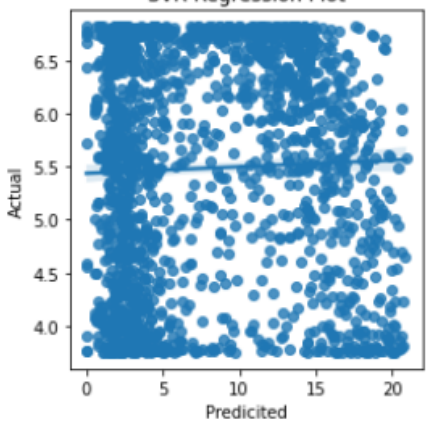
Regarding the regression dataset, I have chosen to implement Decision Tree, Partial Least Squares (PLS), and Ridge regression models. The decision tree will have similar pros and cons to the one used for classification; I think it will perform the worst as it is better suited to non-continuous variables. PLS has the advantage of being able to consider the variability of the dependent variables. Ridge regression then is

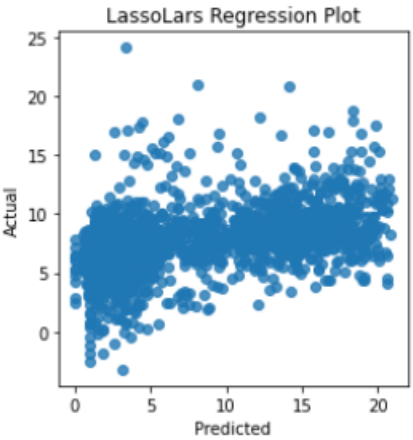
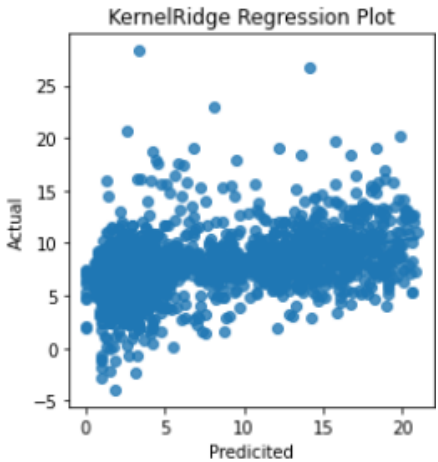
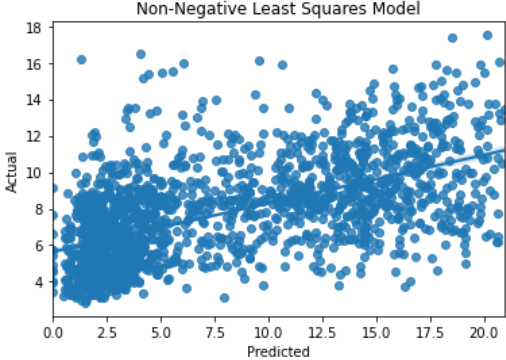
similar to a linear model, and its effectiveness is entirely dependent on whether the data can fit a linear regression method.

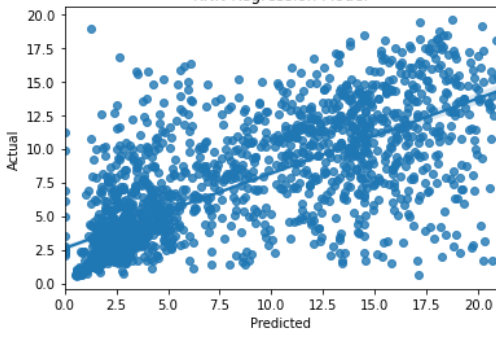

### Evaluation of Models

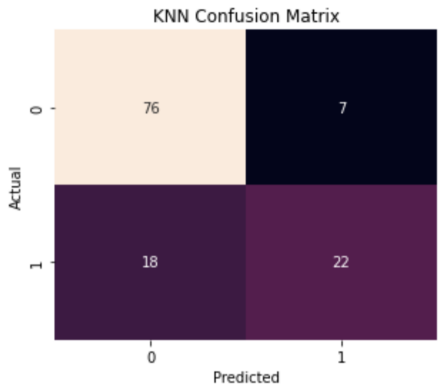
Regression Model Comparison				
Group Member	Model	Regplot	Mean Squared Error	R <sup>2</sup> Score
Marc	Neural nets		20.60	0.47
	Multiple linear regression		27.60	0.28
	Random forest		17.85	0.53

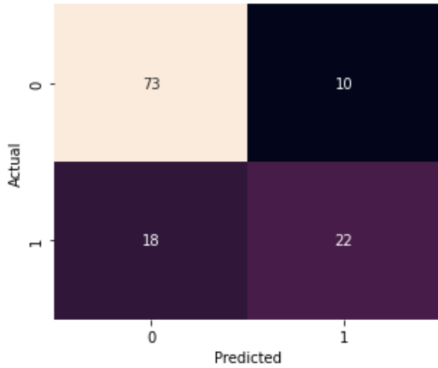
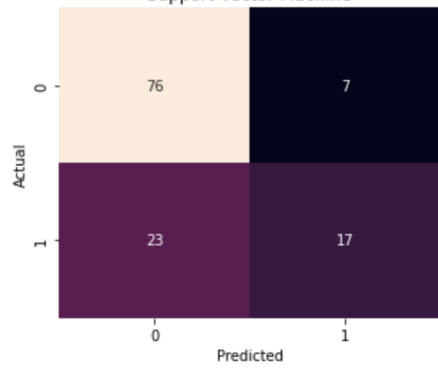
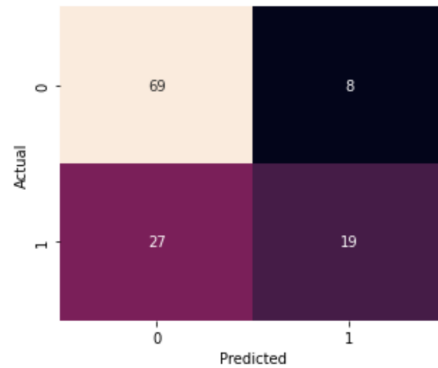
Sadia	Support vector machines (SVM)		27.57	0.23
	KNN with 4 neighbours		18.51	0.49
	Neural nets		22.84	0.36
Mitchell	Decision tree		0.4786	0.9543

	Partial least square (PLS)		0.0000	1.0000
	Ridge		0.00000012	0.99999999
Carter	Support Vector Regression (SVR)		41.3	-0.15

	LARS Lasso	 <p>A scatter plot titled 'LassoLars Regression Plot' showing 'Actual' values on the y-axis (0 to 25) versus 'Predicted' values on the x-axis (0 to 20). The data points are blue dots, showing a positive correlation with some scatter.</p>	29.18	0.19
	Kernal ridge	 <p>A scatter plot titled 'KernelRidge Regression Plot' showing 'Actual' values on the y-axis (-5 to 25) versus 'Predicted' values on the x-axis (0 to 20). The data points are blue dots, showing a positive correlation with some scatter.</p>	30.13	0.17
Davina	Non-Negative Least Squares	 <p>A scatter plot titled 'Non-Negative Least Squares Model' showing 'Actual' values on the y-axis (4 to 18) versus 'Predicted' values on the x-axis (0.0 to 20.0). The data points are blue dots, showing a positive correlation with some scatter. A regression line is visible.</p>	29.18	0.19

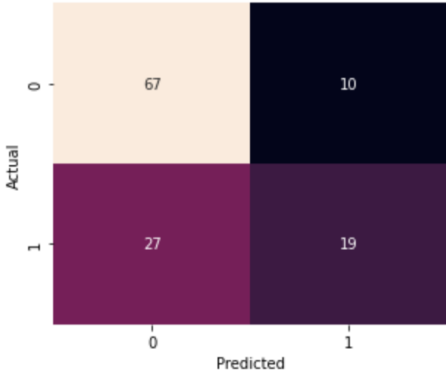
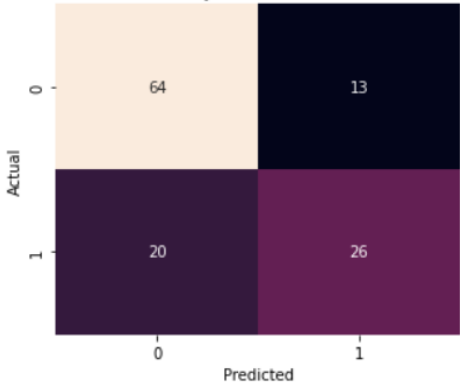
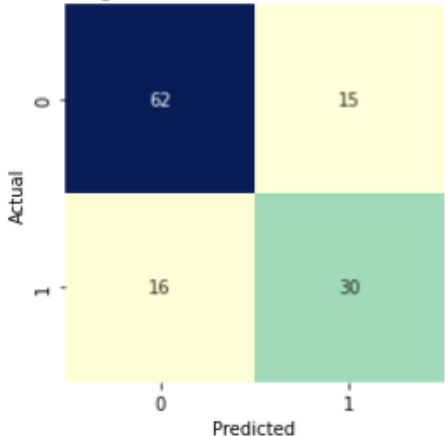
	K-Nearest Neighbours		17.93	0.50
	Gradient Boosting		24.18	0.33

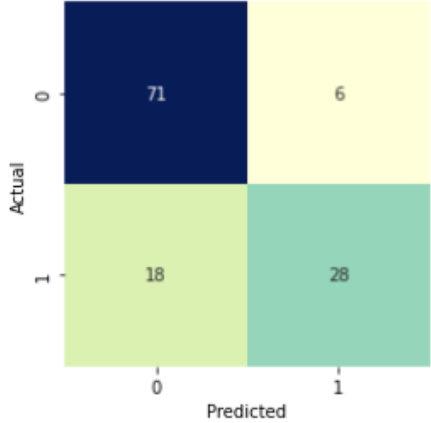
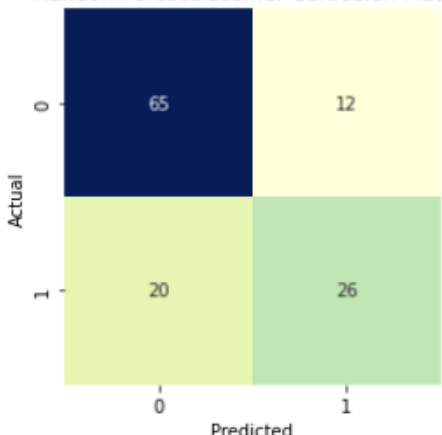
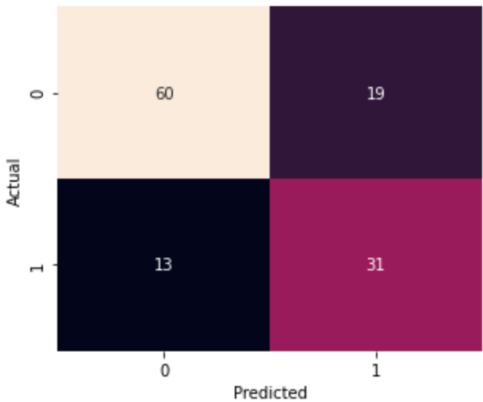
Classification Model Comparison			
Group Member	Model	Confusion Matrix	Accuracy Score (%)
Marc	KNN with 11 neighbours		79.67

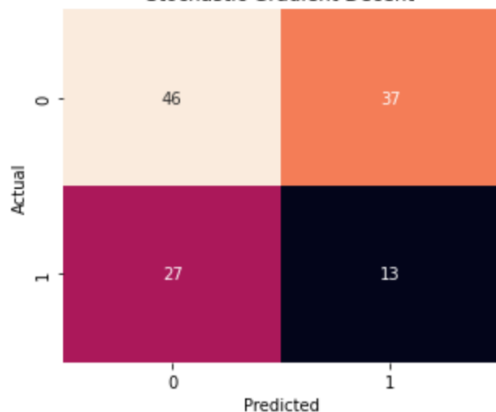
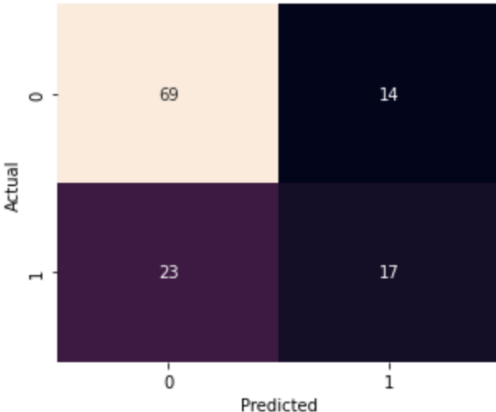
	Decision tree	<div><p>Decision Tree</p><table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>73</td><td>10</td></tr><tr><th>1</th><td>18</td><td>22</td></tr></table></div>	Actual \ Predicted	0	1	0	73	10	1	18	22	77.24
	Actual \ Predicted	0	1									
0	73	10										
1	18	22										
	Support vector machine (SVM)	<div><p>Support Vector Machine</p><table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>76</td><td>7</td></tr><tr><th>1</th><td>23</td><td>17</td></tr></table></div>	Actual \ Predicted	0	1	0	76	7	1	23	17	75.61
Actual \ Predicted	0	1										
0	76	7										
1	23	17										
Sadia	KNN with 4 neighbours	<div><p>KNN Confusion Matrix</p><table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>69</td><td>8</td></tr><tr><th>1</th><td>27</td><td>19</td></tr></table></div>	Actual \ Predicted	0	1	0	69	8	1	27	19	71.54
Actual \ Predicted	0	1										
0	69	8										
1	27	19										



	Neural nets	<div>Neural Net Confusion Matrix</div> <table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>68</td><td>9</td></tr><tr><th>1</th><td>11</td><td>35</td></tr></table>	Actual \ Predicted	0	1	0	68	9	1	11	35	83.74
	Actual \ Predicted	0	1									
0	68	9										
1	11	35										
	Random forest	<div>Random Forest Confusion Matrix</div> <table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>71</td><td>6</td></tr><tr><th>1</th><td>18</td><td>28</td></tr></table>	Actual \ Predicted	0	1	0	71	6	1	18	28	80.49
Actual \ Predicted	0	1										
0	71	6										
1	18	28										
Mitch	Decision Tree	<div>Decision Tree Confusion Matrix</div> <table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>67</td><td>10</td></tr><tr><th>1</th><td>22</td><td>24</td></tr></table>	Actual \ Predicted	0	1	0	67	10	1	22	24	73.98
Actual \ Predicted	0	1										
0	67	10										
1	22	24										

	KNN with 9 neighbours	<div>KNN Confusion Matrix</div>  <table><thead><tr><th></th><th>0</th><th>1</th></tr></thead><tbody><tr><th>0</th><td>67</td><td>10</td></tr><tr><th>1</th><td>27</td><td>19</td></tr></tbody></table>		0	1	0	67	10	1	27	19	69.92
	0	1										
0	67	10										
1	27	19										
	Naive Bayes	<div>Naive Bayes Confusion Matrix</div>  <table><thead><tr><th></th><th>0</th><th>1</th></tr></thead><tbody><tr><th>0</th><td>64</td><td>13</td></tr><tr><th>1</th><td>20</td><td>26</td></tr></tbody></table>		0	1	0	64	13	1	20	26	73.17
	0	1										
0	64	13										
1	20	26										
Carter	KNN with 3 neighbours	<div>KNeighborsClassifier Confusion Matrix</div>  <table><thead><tr><th></th><th>0</th><th>1</th></tr></thead><tbody><tr><th>0</th><td>62</td><td>15</td></tr><tr><th>1</th><td>16</td><td>30</td></tr></tbody></table>		0	1	0	62	15	1	16	30	74.80
	0	1										
0	62	15										
1	16	30										

	Logistic regression	<div>LogisticRegression Confusion Matrix</div>  <table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>71</td><td>6</td></tr><tr><th>1</th><td>18</td><td>28</td></tr></table>	Actual \ Predicted	0	1	0	71	6	1	18	28	80.49
	Actual \ Predicted	0	1									
0	71	6										
1	18	28										
	Random forest	<div>RandomForestClassifier Confusion Matrix</div>  <table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>65</td><td>12</td></tr><tr><th>1</th><td>20</td><td>26</td></tr></table>	Actual \ Predicted	0	1	0	65	12	1	20	26	74.80
Actual \ Predicted	0	1										
0	65	12										
1	20	26										
Davina	Decision Tree	<div>Decision Tree Confusion Matrix</div>  <table><tr><th>Actual \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>60</td><td>19</td></tr><tr><th>1</th><td>13</td><td>31</td></tr></table>	Actual \ Predicted	0	1	0	60	19	1	13	31	73.98
Actual \ Predicted	0	1										
0	60	19										
1	13	31										

	Stochastic Gradient Descent	<div><p>Stochastic Gradient Decent</p><table><tr><th></th><th>Actual 0</th><th>Actual 1</th></tr><tr><th>Predicted 0</th><td>46</td><td>27</td></tr><tr><th>Predicted 1</th><td>37</td><td>13</td></tr></table></div>		Actual 0	Actual 1	Predicted 0	46	27	Predicted 1	37	13	71.54
	Actual 0	Actual 1										
Predicted 0	46	27										
Predicted 1	37	13										
	Neural Networks	<div><p>Neural Net Confusion Matrix</p><table><tr><th></th><th>Actual 0</th><th>Actual 1</th></tr><tr><th>Predicted 0</th><td>69</td><td>23</td></tr><tr><th>Predicted 1</th><td>14</td><td>17</td></tr></table></div>		Actual 0	Actual 1	Predicted 0	69	23	Predicted 1	14	17	69.92
	Actual 0	Actual 1										
Predicted 0	69	23										
Predicted 1	14	17										

## Image Classification Model Evaluation

	precision	recall	f1-score	support
<b>N</b>	0.641361	0.606436	0.623410	404.0
<b>D</b>	0.558394	0.733813	0.634197	417.0
<b>G</b>	0.866667	0.188406	0.309524	69.0
<b>C</b>	0.854545	0.635135	0.728682	74.0
<b>A</b>	0.428571	0.039474	0.072289	76.0
<b>H</b>	0.500000	0.151515	0.232558	33.0
<b>M</b>	0.629630	0.326923	0.430380	52.0
<b>O</b>	0.500000	0.111940	0.182927	268.0
<b>micro avg</b>	0.603261	0.478105	0.533440	1393.0
<b>macro avg</b>	0.622396	0.349205	0.401746	1393.0
<b>weighted avg</b>	0.596417	0.478105	0.485405	1393.0
<b>samples avg</b>	0.539127	0.502883	0.512768	1393.0

Figure 1: Precision, Recall, f1-Score, and Support Values of the Model for each Image Category

### Individual Reflection

Over the course of this assignment I learned a lot about machine learning, and the approach that should be taken to these sorts of problems. Looking at the regression dataset, my models were all considerably more accurate than my other group members. I can understand one ML algorithm/method being more accurate than the others, but my performance was still wildly higher than anyone else's. Considering why this may have been, I noticed that the output dataset has some negative values for the tumor size. Of course, it is impossible to have units of negative size, however I decided not to remove these rows - perhaps the negative was instead trying to suggest a shrinkage of the tumor by that size. In any case, I decided not to manipulate the dataset, and I came out with highly accurate regression models at the expense of having a negative-valued output.

Observing my classification dataset, my output accuracy was not as high as the regression problem. I believe the main reason for this is the difference in the size of the datasets - the regression problem had several thousand entries, while the classification dataset was sub-1000. The most accurate models for me were Decision Trees, and then Naive Bayes. While the Tree did have slightly higher accuracy, I also noticed that it was more likely to declare a false positive, where the prediction is that the person does not have a disease when they really do. Therefore, if this model were to be utilized in a real health setting I actually prefer the Bayes approach since it makes it less likely to declare a false negative outcome.

Finally, I'll mention briefly that this was my first class working on data science, but also with code-collaboration over Git. I enjoyed learning it, and if given the opportunity I will definitely find it useful in my internship, which I am completing this next year.

## Report Summary

To compare our **regression** models, our group evaluated the mean squared error (MSE) and  $R^2$  score. The most accurate model was the Partial Least Squares (PLS) model, with perfect accuracy and zero error. This model did not have any manipulation performed to the original dataset, so the model was trained/tested with all the data; some values in the output *tumor\_size* column were negative, and so the model was built to allow negative outputs. Otherwise, PLS has the advantage of being able to consider the variability of dependent variables, perhaps contributing to its effectiveness. The least accurate model was Support Vector Regression (SVR) with an  $R^2$  score of -0.15, making it less accurate than the mean of the data. MSE was 41.3, which was also extremely poor. The low performance of SVR could be attributed to the noise present in the data set. For datasets containing lots of noise or overlapping target classes SVR is known to underperform. Additionally, SVR is not very accurate for large datasets as overfitting becomes a problem. These can lead to the model not accurately predicting the target.

In order to directly compare the individual **classification** models, each group member calculated for the accuracy score. The accuracy is a simple ratio of the correctly predicted values to the total observations (predictions). In addition to the accuracy score, we each determined the confusion matrix of each classification, an effective way to depict the predicted results and class distribution in the data, along with the breakdown of error types. These calculations allow us to directly compare the efficiency of each of our models numerically. The classification model that performed the best is the neural networks due to the fact that it is an efficient model for non-linear data with a large number of inputs, providing an accuracy score of 83.74%. Whereas the classification model that was observed to perform the worst was K-nearest neighbors with nine neighbors, giving us an accuracy score of 69.92%. This is due to the fact that KNN does not work well with high dimensionality, as it complicates the calculating process for distance between each dimension, in addition to the required scaling in each dimension.

The classification model used in the **image input** problem was classification through convolutional neural networks (CNN). The model runs very smoothly, however, the time for it to run all the way through could vary from 10 minutes to 13 hours. The cause of this running time difference is the number of epochs inputted to train. Each epoch runs five to 10 minutes, thus, if one inputs 100 epochs to train, the model could run for more than 12 hours. From our results, we determined that training 83 epochs would give the best results due to it having an accuracy of 86.1% and only having a loss of 10.7%. However, due to time constraints, the epochs inputted to train were 100 epochs. As was learned in class, an excessive amount of epochs could also decrease the accuracy of the model. In our case, when testing the model, the accuracy received was 55.9% with a loss value of 35.8%. This low value of accuracy could be traced back to the number of epochs selected to be trained. Figure 1 displays the precision percentage, recall percentage, f-1 scores, and support values for each category of the images.

In comparison to other models like Dense Neural Network (DNN), CNN would still be a better choice for image classification. If it took around 80 epochs for the model with more than 12,000 input images to

converse using CNN, it would take more than twice the number of epochs to be trained if DNN was used. CNN uses a method called convolution, hence the name, which applies a filter on the image input to filter information to create a feature map. This feature map is on a layer called the convolutional layer which autonomously recognizes the features of the image. The convolutional layer increases the accuracy of the model recognition because, unlike DNN, the neurons are not densely connected to every neuron in the next layer [7].

### **Group Report Contributions:**

- Carter: Built 3 regression models (SVR, LARS Lasso, Kernal ridge), 3 classification models (KNN with 3 neighbors, Logistic regression, Random Forest), built the CNN for image classification, wrote the chosen hypothesis for the image classification case study
- Davina: Built 3 classification models (Decision tree, Stochastic Gradient Descent, Neural networks), 3 regression models (Non-negative least squares, K-nearest neighbors, Gradient boosting descent) Wrote the comparison of the classification models.
- Sadia: Built 3 regression models (SVM, KNN with 4 neighbors, Neural Nets), 3 classification models (KNN with 4 neighbors, Neural Nets, Random Forest), wrote and modified introduction, the evaluation model section for both the regression and classification model comparison, created the GitHub and organized the folders
- Marc: Built 3 regression models (Neural Networks, MLR, Random Forest), 3 classification models (kNN with 11 neighbors, Decision Tree, SVM), built the CNN for image input classification and wrote the summary section for the results of the image input problem model.
- Mitchell: Built 3 regression models (Decision tree, PLS, Ridge), 3 regression models (Decision Tree, KNN, Naive Bayes), writing regression summary and drawing conclusions.

### **Group Git Repository**

[https://github.com/sadiatasneemkhan/BMEN\\_415\\_Project](https://github.com/sadiatasneemkhan/BMEN_415_Project)

## References

1. A. Kumar, "Machine Hack: Melanoma Tumor Size Prediction." Kaggle.com. [Online]. Available: [https://www.kaggle.com/datasets/anmolkumar/machine-hack-melanoma-tumor-size-prediction?select=sample\\_submission.csv](https://www.kaggle.com/datasets/anmolkumar/machine-hack-melanoma-tumor-size-prediction?select=sample_submission.csv). [Accessed: 11-Apr-2022].
2. "Diabetes Classification." Kaggle.com. [Online]. Available: <https://kaggle.com/competitions/diabetes-classification>. [Accessed: 11-Apr-2022].
3. Larxel, "Ocular Disease Recognition." Kaggle.com. [Online]. Available: <https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k>. [Accessed: 11-Apr-2022].
4. "Pros and Cons of Decision Tree Regression in Machine Learning," upGrad blog, 24-Dec-2020. [Online]. Available: <https://www.upgrad.com/blog/pros-and-cons-of-decision-tree-regression-in-machine-learning/>. [Accessed: 11-Apr-2022].
5. A. Singh, "Implement Knn Pros & Random Forest In Python," *DataSpoof*, 13-Jan-2021. [Online]. Available: <https://www.dataspoof.info/post/implementation-of-k-nearest-neighbors-from-scratch-in-python/>. [Accessed: 12-Apr-2022].
6. A. Soni, "Pros and Cons Of Naive Bayes Classifier :," *Medium*, 04-Jul-2020. [Online]. Available: <https://medium.com/@anuuz.soni/pros-and-cons-of-naive-bayes-classifier-40b67249ae8>. [Accessed: 12-Apr-2022].
7. Y. Gavrilova, "Convolutional Neural Networks for Beginners," *Serokell Software Development Company*, 03-Aug-2021. [Online]. Available: <https://serokell.io/blog/introduction-to-convolutional-neural-networks>. [Accessed: 12-Apr-2022].