

BMEN 415  
Sensor Systems and Data Analytics

Group 11

Date of Submission: April 12, 2022

Professor: Dr. Ethan MacDonald

Akanksha Bhargava (30088153)

Zaid Mujtaba (30095352)

Phuong Phung Vu Hoang (30087108)

Sayma Haque (30093666)

Ethan Le (30093338)

# Group Section

## Introduction

The goal of this project is to perform classification, regression and a challenging case study on image data by implementing various machine learning models.

The evaluation of the classification models was conducted by comparing accuracy and confusion matrices. The classification problem focuses on classifying fetal health in order to prevent child and maternal mortality, hence classes of health is our target variable. The target variable is recorded as a binary number with 1, 2, or 3 which describes fetal health as normal, suspect, or pathological, respectively, and overall there are 21 features with 2126 samples. The evaluation of models for regression will be completed through comparison of the  $r^2$  value and scatter plots.

The regression problem will consist of predicting the target variable “Age” which represents brain age measured in years. There are overall 138 features including volumetric data, which consist of continuous values. The data is also separated into different numbers and datasets specified from 1 to 9. If the data were to be modelled without randomizing the order of the data before splitting, the model would be trained on specific subsets, reducing performance. To avoid this problem and reduce inconsistencies across datasets, only the first dataset was used in training and validation.

The image classification dataset used for this project was a set of human faces distinguished by ages in years from ages 1 to 110, spanning 9778 photos overall, and focused on identifying the age of a face based on the characteristics of an image’s face through regression. The target variable will be the age of the individual in the picture and the features are based on the images themselves.

## Evaluation of Models

### Classification Case Study

In developing appropriate models for classification with the fetal health dataset, all group members preprocessed the data by first using feature selections. In preprocessing, an extra-trees classifier is first used to fit 20 randomized decision trees onto a minimum of two sub-samples, then filters out the data for the top 9 features which impacts the outcome the most. Feature selection reduces the training time of the machine learning models and also reduces the potential for overfitting. After feature selection, the data is then scaled to normalize its value to reduce outliers. After preprocessing the data, the data was split randomly with 20% of the data being used for testing. Once the data had been split, all group members used various models with differing parameters to compare the effectiveness of each model as described by Table 1, where the accuracy and confusion matrix is displayed for each model.

**Table 1: Classification Case Study Metrics and Visualizations**

Member	Model	Metrics and visualizations	
		Accuracy	Confusion matrix
Phuong	Logistic Regression	86.9	[306 14 6 ] [23 33 2 ] [3 8 31]
	Naive Bayes	78.9	[261 49 16] [8 48 2 ] [2 13 27]
	Decision Tree (entropy criterion)	92.7	[317 8 1 ] [14 42 2 ] [3 3 36]
Akanksha	Linear Discriminant Analysis	87.09	[305 13 3] [ 24 39 3] [ 5 7 27]
	Decision Tree (gini criterion)	91.31	[305 14 2] [ 13 50 3] [ 2 3 34]
	Support Vector Machines	89.43	[309 11 1] [ 20 42 4] [ 4 5 30]
Sayma	Ridge	87.09	[326 4 3] [ 31 32 1] [ 5 11 13]
	Gradient Boosting	90.61	[317 11 5] [ 13 50 1] [ 4 6 19]
	kNN with 5 Neighbours	91.08	[323 9 1] [ 16 44 4] [ 4 4 21]
Ethan	kNN with 10 Neighbours	90.38	[332 3 1] [ 28 23 1] [ 2 6 30]
	Random Forest	96.24	[335 1 0] [ 12 40 0] [ 0 3 35]
	Multi-Layer Perceptron	93.90	[329 6 1]

	Neural Network (1 Hidden Layer with 100 Neurons)		[ 14 38 0] [ 1 4 33 ]
Zaid	Logistic Regression	84.51	[315 10 6] [ 25 27 3] [ 14 8 18]
	Multi-Layer Perceptron Neural Network (2 Hidden layers with 100 neurons and 50 neurons)	88.50	[334 9 2] [ 17 32 1] [ 1 1 29]
	Decision Tree (Criterion = Entropy and Max Features = log2)	91.31	[328 13 3] [ 14 36 2] [ 2 3 25]

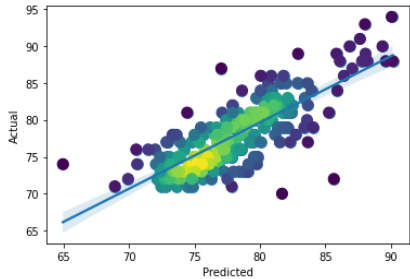
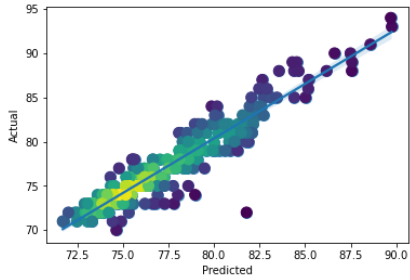
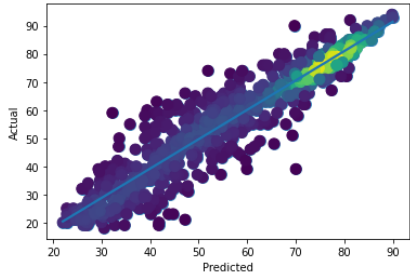
The model that is the most accurate in predicting the outcome of fetal health is the random forest model with 96.24% accuracy, which is expected to perform well since there are more than 3 outputs, meaning the dimensionality is higher. The model with the lowest accuracy is the naive bayes model which could be due to the assumption that all features are independent of one another, which is not true. The model that contains the most true outcome for normal health is the random forest, most likely due to the previous advantage stated earlier. Both gradient boosting and decision tree were able to accurately determine the suspect fetal health better than the random forest with 50 true values. This potentially implies that the random forest does not perform as well in predicting for smaller amounts of an outcome class compared to the two other models. For determining the pathological fetal health, the decision tree is again able to predict the most true outcomes compared to the other models, with the difference being that the gradient boosting model performs significantly worse than before. It can also be noted that the k-nearest neighbour model with 5 neighbours performs slightly better than the k-nearest neighbour model with 10 neighbours, since the increase of neighbours can lead to overfitting the data. In comparing the three decision tree models, the decision tree nodes are split differently. The maximum features considered before splitting is equal to the total number of features in the first two models (which is the default) and log2 of the total in the last model. The effect this has on the result seems negligible. With the two multi-layer perceptron neural network models, there is a difference in the number and size of the hidden layers. The first model uses the default setting of one hidden layer with 100 neurons, while the second model uses two hidden layers with 100 and 50 neurons respectively. The first, even though it is the default, outperforms the second model.

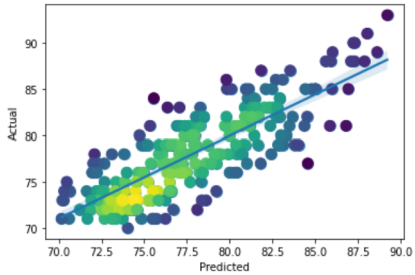
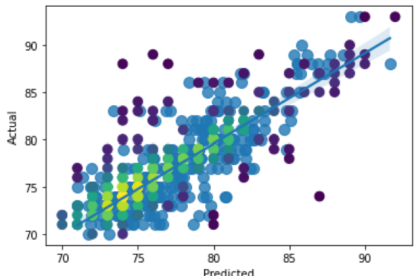
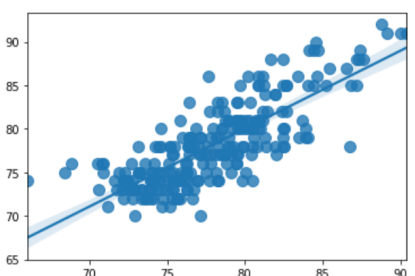
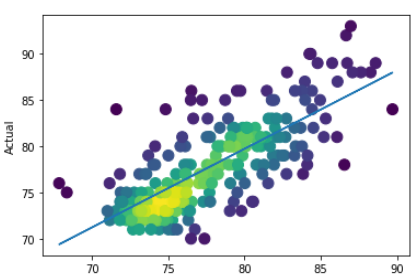
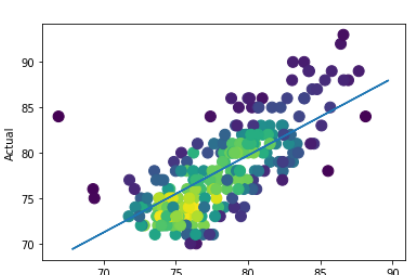
## Regression case study

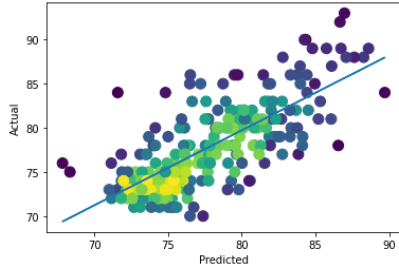
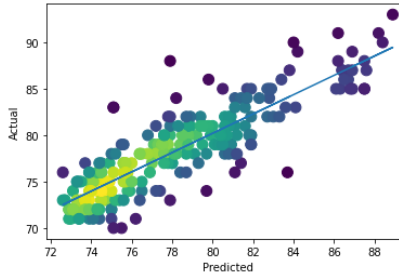
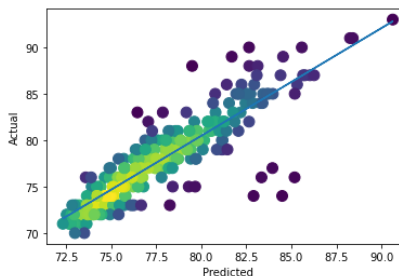
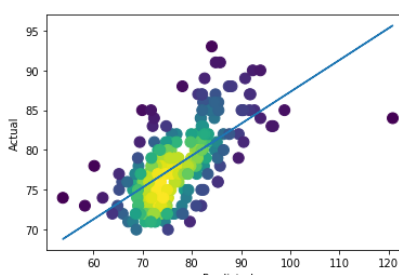
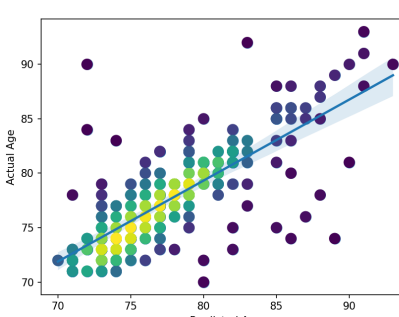
The dataset for regression case study was taken from [1] which is based on predicting brain age from volumetric features. For this case study, only the first data set (data set 1) was used, because if the data were to be modelled without randomizing the order of the data before splitting, the model would be

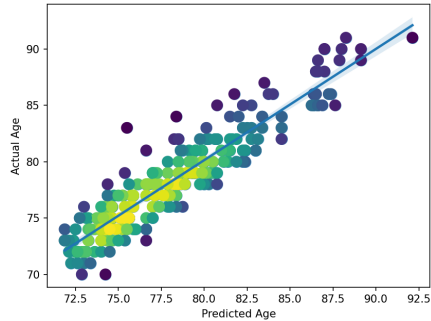
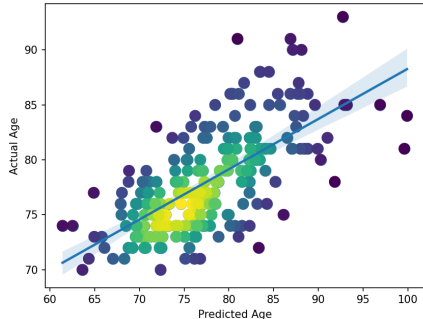
trained on specific subsets, reducing performance. For some of the models it was found that scaling and/or principal component analysis (PCA) improved the  $r^2$  value, however, this was not the case for all the models. These inconsistencies may have stemmed from problems in the implementation of PCA and it was determined that the dimensionality was already adequate and did not need to be reduced. The metric and visualization used to evaluate the case study are  $r^2$  and density scatter plots, which are provided below in Table 2. Models with a higher  $r^2$  can better fit datasets. A train/test split of the entire dataset was decided to be 80% of data for training and 20% for testing.

**Table 2: Regression Case Study Metrics and Visualizations**

Member	Model	Metrics and visualizations	
		$r^2$	Density Scatter Plot
Phuong	Multiple linear regression (MLR)	0.61	
	K-nearest Neighbours Regression (KNN - 3 neighbors)	0.68	
	Random Forest	0.84	

Akanksha	Linear Regression	0.61	
	Decision Tree	0.59	
	Partial Least Squares	0.62	
Sayma	Support Vector Machine	0.53	
	Gaussian Process Regressor	0.56	

	Stochastic Gradient Descent	0.61	
Ethan	kNN with 10 Neighbours	0.752	
	Random Forest	0.739	
	Multi-Layer Perceptron Neural Network (1 Hidden Layer with 100 Neurons)	-0.536	
Zaid	Decision Tree (Criterion: Friedman MSE)	0.519	

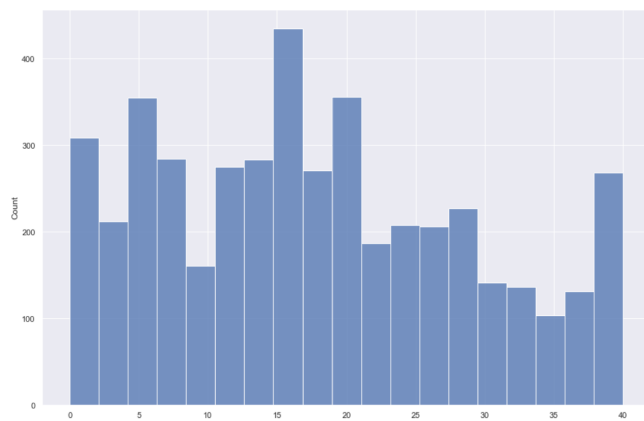
	kNN with 8 neighbors	0.859	
	Multi-Layer Perceptron Neural Network (Hidden layers: 150, 100)	-0.124	

With this split set, the best performance model is kNN with 8 neighbors and random forest, whose testing accuracy is 85.9% and 84%, respectively. It can be seen from kNN with 8 neighbors scatterplot, that a higher accuracy model will generate a better data fit, in comparison with partial least squares. Other models'  $r^2$  values, except the two Multi-Layer Perceptron Neural Networks, ranges from .5 to .7, in which little difference between them was observed. With the multi-layer perceptron neural networks, the model with the additional layer and overall greater number of neurons performed better than the other, which contains a negative  $r^2$  value of -0.536. This negative number itself implies that the neural network cannot properly train and test the data with its current parameters, most likely due to the reduced number of layers and neurons.

## Image Case Study

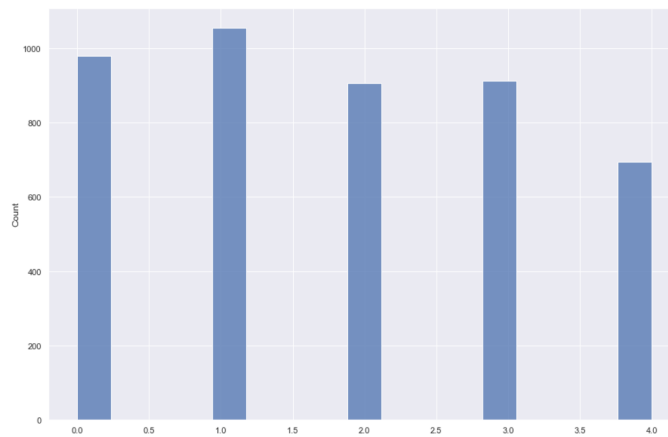
The image input used for this project is a set of human faces distinguished by ages in years. The target variable is the age of the individual in the picture and the features are based on the images themselves. The data set consists of 99 folders of images for the ages 1-100 with 9778 files. The image data was very unbalanced because there are more images for ages 1-10 and very few for ages 80-100. In order to balance the data, the age range that was used was restricted from 10-50. This removes the number of classes required for classification and ensures that the data is more balanced. The data set was reduced to the age classes 10-50 by directly editing the data source folder. Next, the image data was imported with the age data saved as the labels of the data (ie. age labels). After importing the images, a histogram was plotted to see the distribution of the data. Reducing the classes of the original data set ensured that the image data was more balanced. Figure 1 shows the class distribution for the edited data set. Even after cutting down the data set, a class imbalance is still visible. This will reduce the accuracy of the results obtained from the model implemented.





**Figure 1:** Class Distribution of Trimmed Data Set

The images from the data set were analyzed to determine the size of the images, which was 200 x 200 pixels. The image size is used as an input in the model. In order to further increase the accuracy of classification, the original data set was reduced to a smaller number of classes through binning. For example classes 0-7, corresponding to ages 10-17 was assigned to a class '0'. Through re-categorizing the data, the class imbalance is reduced, as seen in Figure 2. Hence, this data can now be processed by the model.



**Figure 1:** Updated Class Distribution of Trimmed Data Set after Re-categorization

The model that was implemented for the image case study is a convolutional neural network (CNN) made up of many layers of artificial neurons. The image data used for this model generates activation functions in each layer and passes them on to the next. In the model itself, training and testing data was used to predict and evaluate the outcomes of the model. The model used keras layers that had a specified number of hidden layers and activation functions, specifically Relu and softmax. The loss function that was run was Sparse Categorical Cross Entropy. For the model itself, epochs indicate when the dataset is passed forward and backward through the neural network once. 10 epochs were run for the model. While more computationally expensive, running more epochs would have improved accuracy as it reduces the amount of error for each time the dataset is passed through the neural network. A validation accuracy of **23.21%** was achieved. The accuracy of the model is very low indicating poor performance.

A number of factors may have resulted in the low performance of the model, with many of the issues stemming from the data set. First, there are very clear class imbalances in the data set. Even with binning, which reduces the categories and improves the class distributions, the accuracy was reduced. In order to improve accuracy, a further step that could be taken is to complete more pre-processing; for example data augmentation could be implemented to reduce class imbalances. Second, there are a lot of different features in the image data such as race and gender, these may have affected the predictions made by the model. Diversifying the images used in the data set (for features such as race) and increasing the number of images would likely improve the performance of the model. Overall, the performance of the model was limited by the data set used. It should be noted that in earlier iterations, the model was implemented as an image regression model, since there are a large number of classes. However, the accuracy achieved was below 5% and so a new classification approach using binning was implemented.

## Summary

In order to compare our Classification models, an accuracy score was computed for each model which was compared between all 15 models in order to rank the algorithms. The accuracy reflects a ratio of the correctly predicted values to the total observations. A confusion matrix was also determined for each model to breakdown and compare errors. The classification model that performed the best was the Random Forest model with an accuracy of 96.24%. This can be attributed to the simplicity of the model, its use of cross validating to prevent overfitting, and how little tuning it requires to achieve its efficiency. It should also be noted that the data being classified result in 3 outputs (higher dimensionality), which is ideal for a Random Forest Model. The worst algorithm of the ones we tested seemed to be Naive Bayes with an accuracy of 78.9%. This can be attributed to the fact that the Naive Bayes algorithm assumes that all features are independent of one another, which is not true in this case study.

In order to compare our Regression models, the  $r^2$  was computed for each model which was compared between all 15 models in order to rank the algorithms. The  $r^2$  reflects how much variation in the dependent variable that can be explained by the independent variable, this is represented by a value between 0 and 1.0. A density scatter plot is used additionally to compare the distribution of data. The regression model that performed the best was the kNN model with 8 neighbours which has an  $r^2$  value of 0.859. This can be attributed to the highly sensitive nature of the KNN model. Though it has the risk of overfitting with too many neighbors, a higher amount is preferred to increase efficiency. The worst algorithm of the ones we tested seemed to be the Multi-Layer Perceptron Neural Network with both versions having  $R^2$  values of less than 0. This result can indicate that the algorithm cannot properly train and test the data with its current parameters, most likely due to the reduced number of layers and neurons.

The final part of this report covers the Image Case Study which aims to identify ages of humans based on an image of their face. From this case study both regression and classification were explored, however, the best results were determined from classification through reducing the number of classes of the data. The results of the image case study in terms of accuracy was low, exemplifying limitations of the data set utilized. In addition, more areas of improvement such as data augmentation were identified.

Overall, our group feels that we have learned a tremendous amount about machine learning through this course and this project. The biggest challenge our group overcame was making connections between the

technical knowledge explored in the class and connecting it to the implementation of our models. We found it very helpful to do research online about the models we were implementing and connecting it with the concepts we learnt in class. Furthermore, difficulty was greatly offset by the presence of multidisciplinary team members, specifically those with experience in programming and machine learning. This was a new course for most of us, and as a result the learning curve was quite high. Nonetheless, we consider this course to be one of the most important courses in this minor program. The focus on data analytics and current programming methods to achieve efficiency in processing/predicting information is both extremely relevant and vital to our future work in this program.

## Individual Section For Sayma Haque

### Hypothesis:

When comparing the three classification algorithms: Gradient Boosting, k-Nearest Neighbours with 5 neighbours, and Ridge, I hypothesize that the kNN model will perform the best and therefore have the highest accuracy while the Ridge algorithm will perform the worst of the three and have the lowest accuracy. My hypothesis is a result of in-class exploration of the algorithms as well as general knowledge found in research about how the algorithms classify data. When looking at the kNN model, we know that the model can be quite sensitive. Though in regression this is not ideal, it has the potential to do quite well in classification case studies despite being so simple. There is always a risk of overfitting, which is why I chose to keep the neighbours at a mid-sized number (5). In contrast, Ridge Algorithms are best used in regression case studies as it adds bias to fit a model. It also works best when there are less training values than predictor values, which is the opposite in this classification case study.

When comparing the three regression algorithms: Support Vector Machine, Gaussian Process Regressor, and Stochastic Gradient Descent. I hypothesize that Stochastic Gradient Descent will perform the best while Support Vector Machine will perform the worst. With SGD, larger datasets are known to converge much faster than other algorithms as parameters are updated more frequently. This makes it easier to escape local minimums of loss functions and in turn increases efficiency. With these considerations, I hypothesize the  $r^2$  value will be higher than the other two models. In contrast, SVM perform worse with large datasets and higher dimensionality, both of which are present in this case study. In this case, the  $r^2$  value will be lower than the rest.

### Reflection and Contributions:

After running my classification models, I was able to conclude that the results supported my initial hypotheses. As hypothesized, the kNN model outperformed the other two models with an accuracy of 91.08% while the Ridge algorithm underperformed with an accuracy of 87.09%. In general, the accuracy for all three models was between 87% and 92% as seen in Table 1. After running my regression models, I was able to conclude that the results supported my initial hypotheses. As hypothesized, the SGD model outperformed the other two models with an  $r^2$  value of 0.61 while the SVM underperformed with an  $r^2$  value of 0.53. In general, the  $r^2$  values for all three models was between 0.50 and 0.65 as seen in Table 2.

Personally, I found this project quite challenging despite my background in programming. I had never used python before and moreover I have no experience in Machine Learning. The learning curve was quite big, specifically with using python to implement machine learning in a way I had never done before. Although, I found it satisfying by the end of the course. Despite having to commit many hours to reading the textbook to understand assignments/quizzes/content, I enjoyed the course. In terms of improvements, I would recommend more emphasis on in-class lectures in the future rather than textbook readings to ensure students get the chance to ask questions in class and solidify their understanding. I found the textbook not only draining to read, but confusing at times. The content in this course is truly interesting (and very important in my opinion), I just wish it was presented in a more digestible manner. Nevertheless, the skills I have learned in this course will follow me throughout my program (both major and minor) and I'm thankful to have had the opportunity to indulge in it all.

My contributions in this project include: implementing 3 different algorithms each for classification and regression case studies, assisting group members in programming, specifically editing scatter plots to display lines best fit and debugging with the image study, writing the summary or the project report, editing/formatting the document and information, as well as helping put together the github and assisting group members in committing to the repository.

## References and Repository

GitHub Repository: <https://github.com/KatheerZM/BMEN415-Project>

1. Larxel, "*Fetal Health Classification*" Kaggle.com. [Online]. Available: <https://www.kaggle.com/andrewmvd/fetal-health-classification>
2. F. Rabbi, "*Facial Age*" Kaggle.com. [Online]. Available: <https://www.kaggle.com/frabbisw/facial-age>
3. E. MacDonald, "*Volumetric Features*" d2l.ucalgary.ca. [Online]. Available: <https://d2l.ucalgary.ca/d2l/le/content/426815/viewContent/5226532/View>

In [1]: *# Imports for file and from sklearn*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from scipy.stats import gaussian_kde
```

In [2]: *#Reading, Splitting, and Setting data*

```
data = pd.read_csv(r'Volumetric_features.csv')

X = data.iloc[0:1221,1:139] # X = prediction values
y = data.iloc[0:1221,139] # y = target values

#splits 20% data to test set
xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size = 0.20, random_st

# PCA Scaling
sc = StandardScaler()
X1 = sc.fit_transform(X) #scales data
pca1 = PCA(n_components = 100)
pca1_data = pca1.fit_transform(X1) #fits training data to rest

sc = StandardScaler()
xTrain = sc.fit_transform(xTrain) #fits training data to rest
xTest = sc.transform(xTest) #scales data
```

In [3]: *# Linear Support Vector Machine Algorithm*

```

from sklearn.svm import LinearSVR
from sklearn.pipeline import make_pipeline
svm = make_pipeline(StandardScaler(), LinearSVR(random_state=0, tol=1e-3))
svm.fit(xTrain, yTrain)
yPred1 = svm.predict(xTest)

# Print Results
print ("R^2 score:", r2_score(yTest, yPred1))
print("Mean Squared Error:", mean_squared_error(yTest, yPred1))
print("Mean Absolute Percentage Error:", mean_absolute_percentage_error(yTest, yPred1))

# Print Scatter Plots
sns.scatterplot(yPred1, yTest)
sns.regplot(yPred1, yTest, fit_reg=True, scatter_kws={"s": 100})

XY = np.vstack([yPred1, yTest])
z = gaussian_kde(XY)(XY)
fig, ax = plt.subplots()
ax.scatter(yPred1, yTest, c=z, s=100)
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
a, b = np.polyfit(yPred1, yTest, 1)
plt.plot(yPred1, a*yPred1+b)
plt.show()

```

R^2 score: 0.5323418660089911

Mean Squared Error: 10.88399610115058

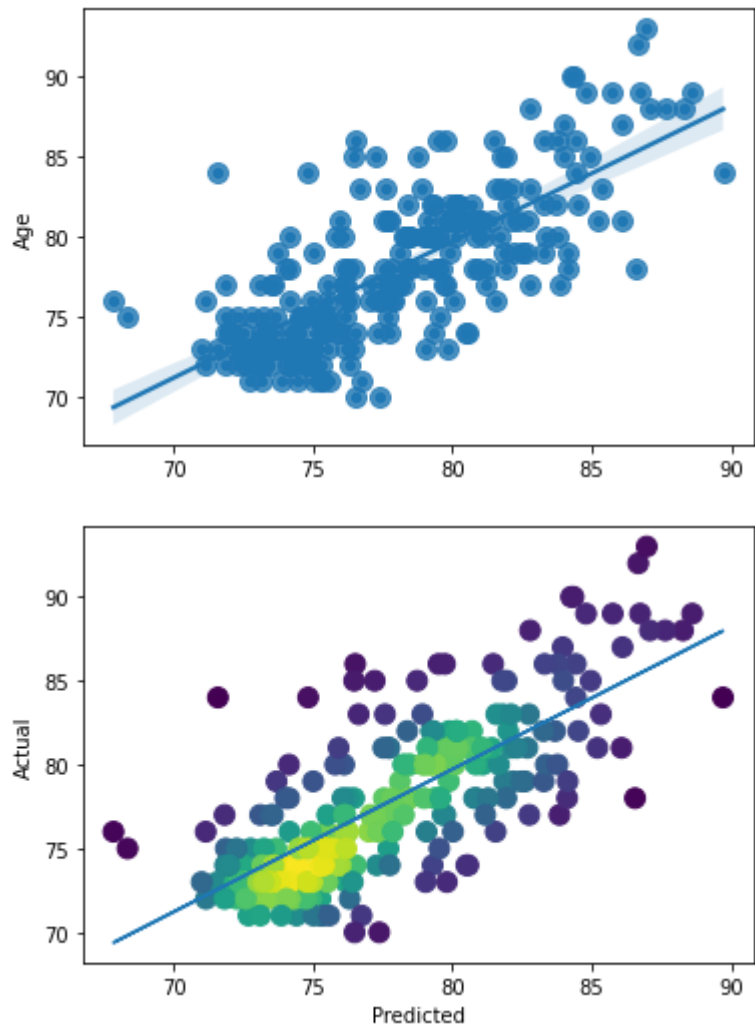
Mean Absolute Percentage Error: 0.03202445314297302

C:\Users\sayma\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

C:\Users\sayma\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



In [4]: *# Stochastic Gradient Descent Algorithm*

```

from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
sgd = make_pipeline(StandardScaler(),SGDRegressor(max_iter=1000, tol=1e-3))
sgd.fit(xTrain, yTrain)
yPred2 = sgd.predict(xTest)

# Print Results
print ("R^2 score:", r2_score(yTest, yPred2))
print("Mean Squared Error:", mean_squared_error(yTest, yPred2))
print("Mean Absolute Percentage Error:", mean_absolute_percentage_error(yTest, yPred2))

# Print Scatter Plots
sns.scatterplot(yPred2, yTest)
sns.regplot(yPred2, yTest, fit_reg=True, scatter_kws={"s": 100})

XY = np.vstack([yPred2,yTest])
z = gaussian_kde(XY)(XY)
fig, ax = plt.subplots()
ax.scatter(yPred2,yTest, c=z, s=100)
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
c, d = np.polyfit(yPred2, yTest, 1)
plt.plot(yPred2, c*yPred2+d)
plt.show()

```

R^2 score: 0.5924729103526027

Mean Squared Error: 9.484542088432468

Mean Absolute Percentage Error: 0.031372683289653557

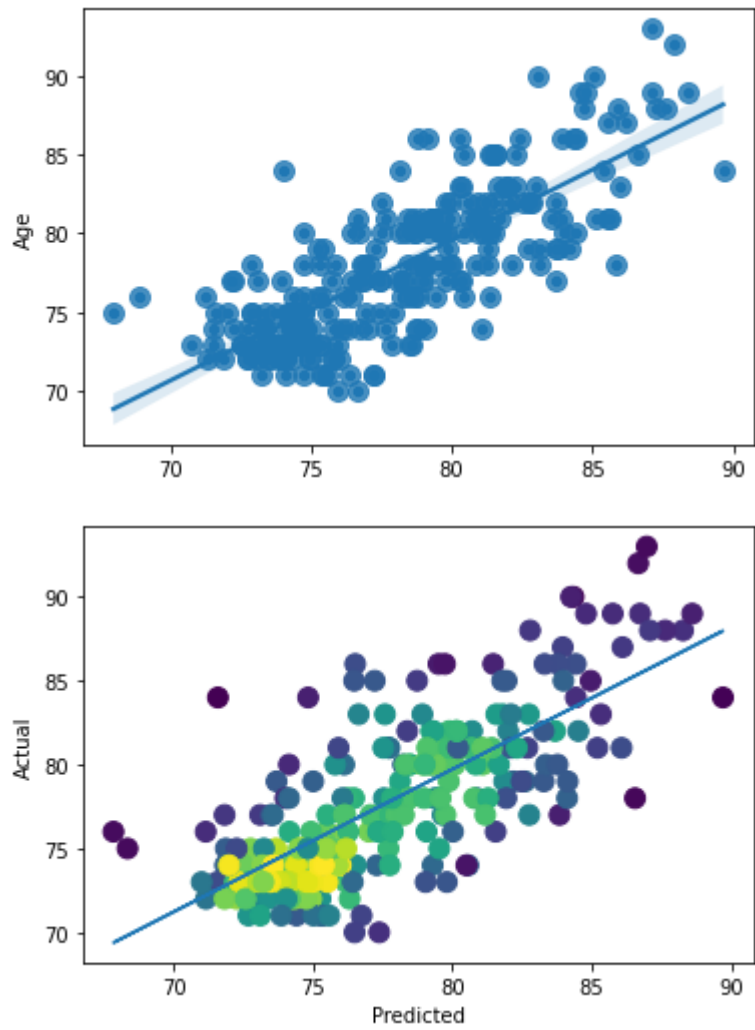
C:\Users\sayma\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

C:\Users\sayma\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(





In [5]: *# Gaussian Process Regressor Algorithm*

```

from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import DotProduct, WhiteKernel
kernel = DotProduct() + WhiteKernel()
gpr = GaussianProcessRegressor(kernel=kernel, random_state=0).fit(xTrain, yTrain)
yPred3 = gpr.predict(xTest)

# Print Results
print ("R^2 score:", r2_score(yTest, yPred3))
print("Mean Squared Error:", mean_squared_error(yTest, yPred3))
print("Mean Absolute Percentage Error:", mean_absolute_percentage_error(yTest, yPred3))

# Print Scatter Plots
sns.scatterplot(yPred3, yTest)
sns.regplot(yPred3, yTest, fit_reg=True, scatter_kws={"s": 100})

XY = np.vstack([yPred1, yTest])
z = gaussian_kde(XY)(XY)
fig, ax = plt.subplots()
ax.scatter(yPred3, yTest, c=z, s=100)
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
e, f = np.polyfit(yPred1, yTest, 1)
plt.plot(yPred1, e*yPred1+f)
plt.show()

```

R^2 score: 0.555657043878693

Mean Squared Error: 10.341372576427823

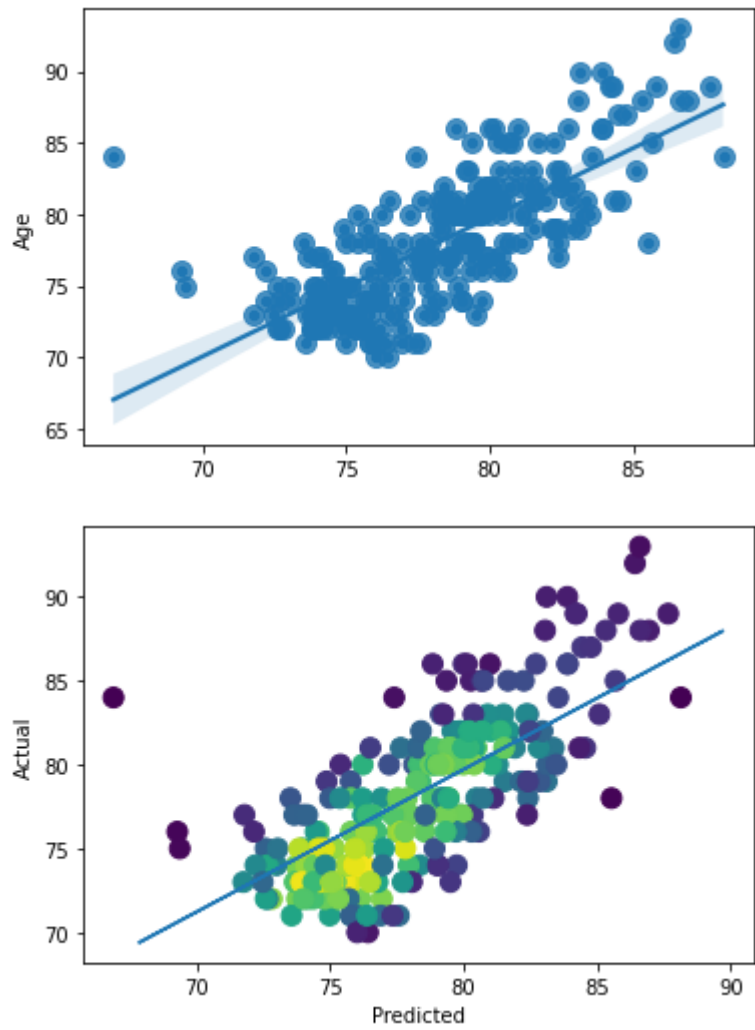
Mean Absolute Percentage Error: 0.03241889670553171

C:\Users\sayma\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

C:\Users\sayma\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



In [1]: *# Imports for file and from sklearn*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier

from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
```

In [2]: *#Reading, Splitting, and Setting data*

```
data = pd.read_csv('fetal_health.csv') #read csv file
target = ['fetal_health']
pred = list(set(list(data.columns)) - set(target)) #set predictions to all data n
data[pred] = data[pred]/data[pred].max()

X = data[pred].values # X = prediction values
y = data[target].values # y = target values

#splits 25% data to test set
xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size = 0.20, random_st
```

In [3]: *#Feature Selection*

```
X.shape
FeatureSelection = ExtraTreesClassifier(n_estimators = 20)
FeatureSelection = FeatureSelection.fit(X,y)
FeatureSelection.feature_importances_
TheModel = SelectFromModel(FeatureSelection,prefit = True)
X_new = TheModel.transform(X)
X_new.shape
```

C:\Users\sayma\AppData\Local\Temp\ipykernel\_19032\3627454629.py:5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().

```
FeatureSelection = FeatureSelection.fit(X,y)
```

Out[3]: (2126, 10)

In [4]: *# Gradient Boosting Algorithm*

```
from sklearn.ensemble import GradientBoostingClassifier
boost = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=3)
boost.fit(xTrain, yTrain[:,0])
yPred1 = boost.predict(xTest)

# Print Results
print("Accuracy:", metrics.accuracy_score(yTest, yPred1),"\n")
print("Confusion Matrix \n",confusion_matrix(yTest, yPred1),"\n")
print("Classification Report \n",classification_report(yTest, yPred1))
```

Accuracy: 0.9061032863849765

Confusion Matrix

```
[[317  11   5]
 [ 13  50   1]
 [  4   6  19]]
```

Classification Report

	precision	recall	f1-score	support
1.0	0.95	0.95	0.95	333
2.0	0.75	0.78	0.76	64
3.0	0.76	0.66	0.70	29
accuracy			0.91	426
macro avg	0.82	0.80	0.81	426
weighted avg	0.91	0.91	0.91	426

In [5]: *# Nearest Neighbours Algorithm (n = 5)*

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(xTrain, yTrain[:,0])
yPred2 = knn.predict(xTest)

# Print Results
print("Accuracy:", metrics.accuracy_score(yTest, yPred2),"\n")
print("Confusion Matrix \n",confusion_matrix(yTest, yPred2))
print("Classification Report \n",classification_report(yTest, yPred2))
```

Accuracy: 0.9107981220657277

Confusion Matrix

```
[[323  9  1]
 [ 16 44  4]
 [  4  4 21]]
```

Classification Report

	precision	recall	f1-score	support
1.0	0.94	0.97	0.96	333
2.0	0.77	0.69	0.73	64
3.0	0.81	0.72	0.76	29
accuracy			0.91	426
macro avg	0.84	0.79	0.82	426
weighted avg	0.91	0.91	0.91	426

In [6]: *# Ridge Algorithm*

```

from sklearn.linear_model import RidgeClassifier
ridge = RidgeClassifier()
ridge.fit(xTrain, yTrain[:,0])
yPred3 = ridge.predict(xTest)

# Print Results
print("Accuracy:", metrics.accuracy_score(yTest, yPred3),"\n")
print("Confusion Matrix \n",confusion_matrix(yTest, yPred3))
print("Classification Report \n",classification_report(yTest, yPred3))

```

Accuracy: 0.8708920187793427

Confusion Matrix

```

[[326  4  3]
 [ 31 32  1]
 [  5 11 13]]

```

Classification Report

	precision	recall	f1-score	support
1.0	0.90	0.98	0.94	333
2.0	0.68	0.50	0.58	64
3.0	0.76	0.45	0.57	29
accuracy			0.87	426
macro avg	0.78	0.64	0.69	426
weighted avg	0.86	0.87	0.86	426

In [ ]: