MACHINE LEARNING ALGORITHMS FOR PREDICTING THE PROGRESSION OF SCOLIOSIS

SRIJA PIRATLA **(1651539)**

RITHVIK RAMESH **(1654067)**

Project submitted in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING

DEPARTMENT: ELECTRICAL AND COMPUTER ENGINEERING

COURSE	**ECE 710**
COURSE NAME	**WEARABLE DEVICES TECHNOLOGIES AND IoT**
DATE OF SUBMISSION	**APRIL 9TH, 2021**
COURSE INSTRUCTOR	**EDMOND LOU**
PROJECT WORK	**PROJECT II**

DECLERATION

We hereby declare that investigatory project entitled “MACHINE LEARNING ALGORITHMS FOR PREDICTING THE PROGRESSION OF SCOLIOSIS” has been carried out by our own efforts and fact arrived at our observation under the guidance and motivation of course instructor “Dr. Edmond Lou”.

Signed by,

Rithvik Ramesh
Piratla Srija

CONTENTS

CHAPTER **Page No.**

ABSTRACT

This paper discusses the modelling of four machine learning algorithms for predicting the progression of scoliosis. This project aims to identify the progression or non-progression of spine curvature for those with scoliosis. The cause of spine deformity is unknown, and hence screening this abnormality becomes tougher. We perform data analysis on a given dataset of 155 patients with input features to predict scoliosis progression. The input features are curve magnitude, ultrasound curve change, reflection coefficient, apical axial vertebral rotation (AVR), and torsion. This paper also includes data preprocessing and standardization of the dataset to improve the models' results and performance. We use the python programming language in the jupyter notebook software to perform the data analysis.

The scoliosis curve progression prediction after few visits with the doctor may or may not be accurate. To get efficient prediction result of the progression of scoliosis, we implement supervised learning classification type algorithms. Seven algorithms are present that will help in the data analysis of the given problem. Our paper uses only four algorithms: logistic regression, decision tree, k-nearest neighbours, and random forest. This project compares the confusion matrix results of each algorithm using only optimal hyperparameter from each model. The logistic regression and k-nearest neighbours produce the best and efficient result. However, between these two k-nearest neighbours gives higher accuracy. Our dataset's size is small, and the data set is imbalanced, due to which results are not 100% accurate.

Keywords: Machine learning, scoliosis, data analysis, jupyter notebook, logistic regression, decision tree, k-nearest neighbours, random forest, confusion matrix, hyperparameter, and accuracy.

1. BACKGROUND

1.1 Introduction

A Normal human spine consists of multiple curves from the neck to the pelvis. Scoliosis is an abnormality of the spine curves that occurs during the growth before adolescence. Scoliosis is usually a hereditary abnormality that sometimes occurs in children for unknown reasons. [1]

Scoliosis is a complex abnormality that involves three planes, namely the Coronal plane, Sagittal plane, and Axial plane. The coronal plane separates the body into ventral and dorsal sections, a vertical plane from head to foot and parallel to the shoulders. The sagittal plane bisects the body into two identical halves: right and left. The plane parallel to the plane of ground and right angles to the latter planes is known as the axial plane. The cervical, lumbar, and thoracic regions in the sagittal plane is where the spine curve is normal. Scoliosis is a spinal malformation that occurs in the coronal plane. [2]

Scoliosis occurring due to unknown reasons is known as idiopathic scoliosis. Idiopathic scoliosis occurs in both boys and girls, but the spine curvature increases in girls, requiring immediate treatment. During the fetus's development in the womb, if spine formation is improper, it is known as Congenital Scoliosis. The deterioration or breaking of the spine discs and bones is known as Neuromuscular Scoliosis. Neuro-muscular scoliosis occurs when the muscle supports the spine unevenly. [1]

There are many symptoms of identifying the occurrence of scoliosis, which is listed below.

- Shoulders look uneven
- The shape of the chest is asymmetric
- The body will lean to one side
- Head is non-centric
- The rib cage and waist look uneven
- The skin will have different textures due to spine deformity

Physical examinations such as X-ray, CT scan, MRI scan or spinal radiograph helps to determine scoliosis. The various angular degree of the curve determines the severity of the treatment. When the curve's angular degree exceeds 45 to 50 degrees, it is considered severe and requires immediate attention and treatment. [2]

After diagnosis, if scoliosis is confirmed, various issues need assessment to determine different treatment options. The issue that needs assessment are: [2]

- Check for the spinal growth of the patient, which is also known as spinal maturity. [2]
- Assess the severity of the curve and also check the way it affects the lifestyle of a person [2]
- Assess the location of the curve because the thoracic curves will progress faster than other spinal curves [2]
- Assess the possibility of curve progression because patients who have curves larger than those during their adolescence will experience severe curve progression [2]

After the assessment of the above issues, three treatment processes are:

- **Observation:** If scoliosis is mild and does not require treatment, the doctor will examine the child throughout adolescence every four to six months. For adults, once in five years, X-rays is taken, and the doctor examines the severity. [2]
- **Bracing:** The doctor recommends braces if the curve angle is between 25 and 40 degrees to prevent further progression of the spine curve. Every day for 16 to 23 hours, the brace is worn to stop the curve's progression to 40 degrees and above. The bracing technique is highly effective and successfully stops the curve's progression in 80% of children with spinal deformity. [2]
- **Surgery:** When the patient has the spinal curve angle above 40 degrees, and the curve's progression is continual, surgery will prevent scoliosis. Surgery is of four types: Posterior approach, Anterior approach, Decompressive laminectomy, and Minimally Invasive Surgery. Each type of surgery is wholly dependent on the region of occurrence of scoliosis. [2]

1.2 Motivation

Adolescent Scoliosis is an abnormality of the spine curve that occurs primarily in children ageing between 10 to 15 years. 80% of the people diagnosed with scoliosis do not know the reason for the occurrence. [3] Girls' ratio to boy's ratio with scoliosis is 8:1 meaning the girls are the most affected by this abnormality. [4] This project's primary purpose is to develop a model that predicts scoliosis progression from the given dataset. An early prediction of scoliosis's progression or non-progression would be a better idea because it can provide an early ailment to this spine deformity. The early prediction will help the doctors take precautionary action before the spine curve angle reaches 45 to 50 degrees. When the spine curve angle or the cobb angle reaches 45 to 50 degrees, it might require immediate surgery because the spine discs or bones will begin to crack. [5]

1.3 Accurate Prediction Models

Two research papers use machine learning and deep learning techniques to predict scoliosis progression. These papers are:

Junlin Yang et al. proposes the development and validation of deep learning algorithms for scoliosis screening using back images. In this paper, they validate the deep learning algorithms for screening Scoliosis to reduce costs for screening. They have developed deep learning algorithms to automate scoliosis screening by using back images. The accuracy of the algorithms was higher than human screening specialists in detecting scoliosis. The deep learning algorithm detects scoliosis before it becomes severe and also in the pre-treatment process. [6]

Hongfa Wu et al. proposes a prediction model of scoliosis progression in time series using a hybrid learning technique. This paper proposes a prediction model to predict the scoliosis progression at six and 12-month intervals with successive spinal indices and hybrid learning technique. The identification of progression patterns of cobb angle was possible by the fuzzy c-mean clustering algorithm. The trained artificial neural network will help to predict the progression of scoliosis. The result was two times more accurate than the clinical measurement. It made this methodology a practical use in the clinical purpose of predicting scoliosis deformity. [7]

2. Machine Learning

Machine learning is a modern-day science that deals with learning as a computational process that combines computer science and statistics tools. Machine learning deals with designing programs that learn rules based on the given data set, improving the performance, and adapting to changes. [8] Machine learning is three types: [9]

- Supervised learning [9]
    1. Regression
    2. Classification
- Unsupervised learning [9]
    1. Clustering
    2. Dimensionality Reduction
- Reinforcement learning [9]

Here in this study, we have specifically chosen supervised learning, and under that, we use classification type.

2.1 Classification

The technique determines the class to which the dependent variable belongs based on the number of independent variables in the given dataset. This technique will train a model for qualitative predictions. [10] There are seven types of classification algorithms, but we will use only four algorithms in our study. The algorithms are: [11]

- **K-Nearest Neighbours**
- **Logistic Regression**
- **Random Forest**
- **Decision Tree**
- Support Vector Machine

- Naive Bayes
- Stochastic Gradient Descent

The classification algorithm in bold is to develop a model to predict scoliosis progression in our study.

2.1.1 K-Nearest Neighbours

K-Nearest Neighbours is a type of supervised machine learning algorithm usable in regression and classification problems. This algorithm will find the similarity between the new data and the available dataset, which will classify the new data into the appropriate category after finding the similarity. This algorithm will efficiently and accurately classify the new data by just finding the similarity. It is a non-parametric algorithm as it does not make any assumptions on the given data. KNN is also a lazy learner algorithm as it does not learn from the training set. It stores the data into the data set, and during the time of classification, it performs specific actions to allocate the data into the particular category. [12]

The reason for selecting this algorithm in our study is that we need to design a model to predict scoliosis progression or non-progression. From the data set given to us, it is understandable that it is a classification type problem. When new data is present, the K-Nearest Neighbours will find the similarity between the given dataset and classify the new data on whether scoliosis will progress. The pseudo-code for K-Nearest Neighbours is:

Algorithm: [13]

1) Initially, we need to load the data
2) Initialize the value of K
3) Iterating from 1 to the total number of training data points to get the predicted class.
    i. Calculate the distance between the test data and each row of training data using the Euclidean distance, Chebyshev, or Cosine, etc.
    ii. Sorting the distances in ascending order
    iii. From the sorted array, get the top K rows

iv. Obtain the most frequent class of these rows

v. Return the predicted class

Advantages: [12]

1. It is a simple algorithm for implementation.
2. It is workable with any number of classes.
3. It is easy to add any number of data.

Disadvantages: [12]

1. It has a high computational cost for calculating the distance between the data points of training samples.
2. It is deplorable for high dimensional data.

2.1.2 Logistic Regression

A supervised machine learning algorithm that deals with classification technique, predicting the probability with only two values. When the data is categorical, we can use logistic regression. Logistic regression is of three types, namely binary, ordinal and multi. Some assumptions in logistic regression are: [14]

- The dependent variable in binary logistic regression should be binary. [14]
- To include a variable, it has to be meaningful. [14]
- When the dependent variable has a one-factor level, it represents the desired outcome in binary regression. [14]
- There exists a linear relationship between the log of the odds and the independent variables. [14]
- A large sample size is preferable for logistic regression. [14]

- In logistic regression, the independent variables should be highly independent of each other. In other terms, the model has little or no multicollinearity. [14]

The reason for selecting this algorithm in our study is that we need to design a model to predict scoliosis progression or non-progression. From the data set given to us, it is understandable that it is a classification type problem. Since we have to predict whether scoliosis will progress or not based on the given dataset, this supervised machine learning algorithm comes into action. We have to choose between progression or non-progression of scoliosis from the given data set, which is achievable using logistic regression.

Advantages: [11]

1. It is beneficial for understanding the influence of several independent variables on a single outcome variable.
2. It is useful for solving both regression and classification problems.

Disadvantages: [11]

1. It works when the outcome or predicted variable is binary.
2. It assumes that the independent variables are independent of each other.
3. It assumes the dataset is not having any missing values.

2.1.3 Random Forest

A supervised machine learning algorithm that creates multiple decision trees are known as a random forest algorithm. When we combine the trees, it is known as an ensemble method. The random forest chooses the outcome's final decision or the predicted variable's final decision based on most trees. The ensemble method will combine all the weak learners to produce a strong learner. [15]

The reason for selecting this algorithm in our study is that we need to design a model to predict scoliosis progression or non-progression. From the data set given to us, it is understandable that it is a classification type problem. Since we have to predict whether scoliosis will progress or not based on the given dataset, this

supervised machine learning algorithm comes into action. Here the random forest classifier will get the highest vote carrying data and will allocate them whether scoliosis will progress or not.

When the data is categorical, we can use the random forest algorithm. The random forest algorithm has two subdivisions: [15]

- Creating random forest classifiers.
- After the creation of a random forest classifier, perform prediction.

Random Forest creation algorithm: [15]

1. Select k features from total m features randomly, where k<<m.
2. Calculate the d node using split point amidst the k features.
3. Using the best split method, split the nodes into offspring nodes.
4. Repeat all the above steps until it reaches n nodes.
5. By repeating steps 1 to 4 for z number of times to create a z number of trees, we can build a forest.

Performing prediction algorithm: [15]

1. Each random decision tree predicts and stores the outcome by taking the test features.
2. For each predicting targets, calculate the votes.
3. The highest vote carrying predicting value or target is the final decision taken by the random forest algorithm.

Advantages: [15]

1. Highly preferable for both supervised and unsupervised learning.
2. Highly capable of handling many input variables without the process of selection.
3. Effectively handles the missing data internally.

Disadvantages: [15]

1. Interpretation of random forest is highly tricky because it is a black-box model.
2. Computational time is longer because it uses a large number of trees.

2.1.4 Decision Tree

A decision tree algorithm is helpful for both regression and classification type problems. It follows a tree-like structure and is a non-parametric type of method. A decision tree has a single node and follows downwards like a branch structure which gives the outcome. Classification type will give the predicting value the outcome to be two values (Eg: Yes or No). [16]

The reason for selecting this algorithm in our study is that we need to design a model to predict scoliosis progression or non-progression. From the data set given to us, it is understandable that it is a classification type problem. Since we have to predict whether scoliosis will progress or not based on the given dataset, this supervised machine learning algorithm comes into action.

Algorithm: [16]

1. The best attribute uses the best split method to split the data.
2. Answer the relevant question.
3. Follow the branches for the tree-like structure.
4. Jump to step 1 until we obtain the answer.

Advantages: [16]

1. The algorithm is easy to understand.
2. Incredibly useful for data exploration.
3. It handles numerical and categorical variables.

Disadvantages: [16]

1. This model has a common problem of over-fitting.
2. Balancing the dataset is a must before fitting it into the decision tree.
3. When many class labels are present, the calculation becomes complex.

3. Results and Discussion

3.1 Correlation

The statistical relationship between two variables or the dependency of one variable on the other variable is present in table 3.1 (how well were they related).

Table 3.1 Correlation mapping in tabular form

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
Output (Progression (P)/Nonprogression (NP)	1.000000	0.397526	-0.063077	-0.064509	-0.107131	-0.024331
US Cobb Change	0.397526	1.000000	-0.134470	-0.082747	-0.059670	-0.028798
Apical Axial Vertebral rotation (AVR)	-0.063077	-0.134470	1.000000	0.898426	0.053471	0.187917
Torsion	-0.064509	-0.082747	0.898426	1.000000	0.085864	0.177991
RC value	-0.107131	-0.059670	0.053471	0.085864	1.000000	0.047044
Curve magnitude	-0.024331	-0.028798	0.187917	0.177991	0.047044	1.000000

From the correlation heatmap shown in figure 3.1, we can see a positive correlation between Torsion and Apical Axial Vertebral rotation (AVR). The correlation between Torsion and Apical Axial Vertebral rotation (AVR) is also clearly visible in the scatterplot shown in figure 3.2. If one variable increases, the other correlated variable changes accordingly.

Figure 3.1 Correlation Heatmap

The skin tone colours in figure 3.1 show how highly the torsion and Apical Axial Vertebral Rotation (AVR) are correlating with each other.

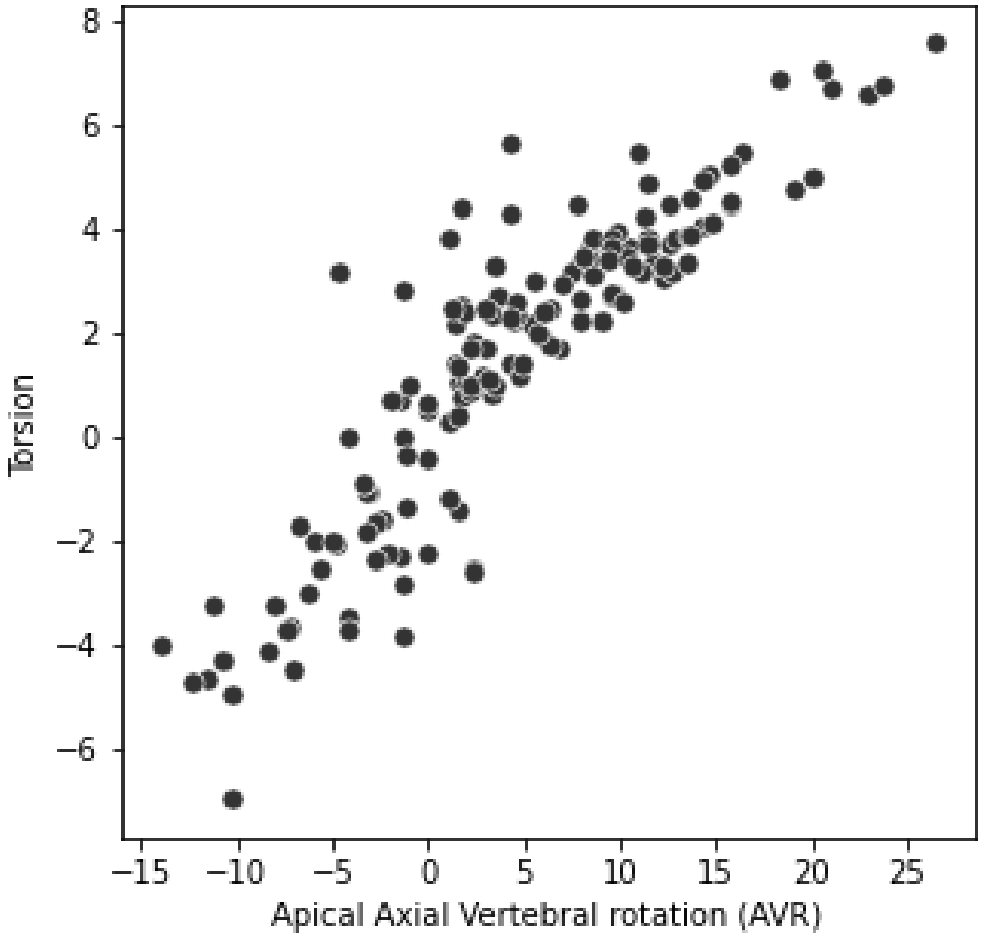


Figure 3.2 Scatterplot of Torsion Vs AVR

3.2 Results and Analysis

The classification models' measuring metric is the confusion matrix. It measures machine learning classification algorithms' performance where the output or the dependent variable is of two classes. The confusion matrix is a 2x2 matrix with four different combinations of predicted and actual values as shown in figure 3.3. So, the confusion matrix helps us in calculating Precision, Recall, Specificity and Accuracy. [17]

True Positive: Prediction is positive, and it is true. [17]

True Negative: Prediction is negative, and it is true. [17]

False Positive (Type 1 Error): Prediction is positive, and it is false. [17]

False Negative (Type 2 Error): Prediction is negative, and it is false. [17]

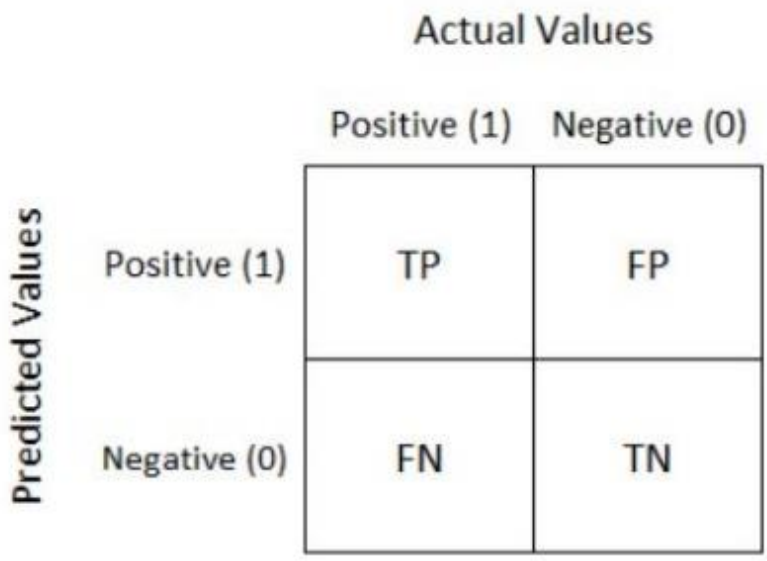


Figure 3.3 Confusion Matrix [17]

Since the dataset we have is extremely limited in size and on top of it, the data we had was imbalanced. The splitting ratio for the training set was 80% of the data size and 20% for test size. There were 122 data points related to Non-Progression class and 32 data points related to the Progression class.

Logistic Regression

Figure 3.4 depicts the confusion matrix for the **Logistic Regression** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the logistic regression's metric values. Calculation of the test dataset on 31 new data points gives us the logistic regression's metric values.

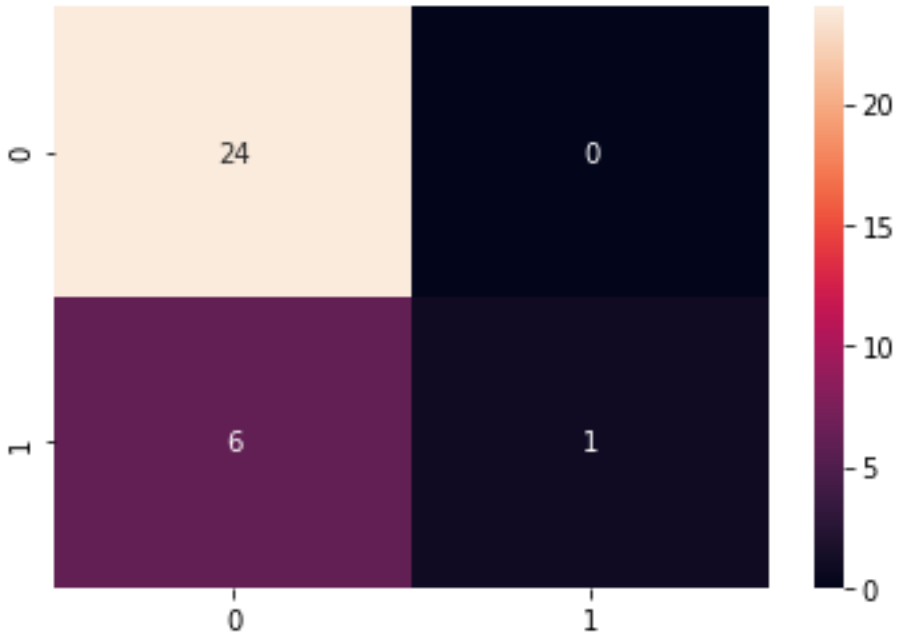


Figure 3.4 Confusion Matrix for Logistic Regression

1. Accuracy score: 80.65
2. Precision score: 1.00
3. Recall score: 0.14
4. f1_score: 0.25

Since the model shows, the value for true positives is 0 (the true predictions were 0 for Progressive class). So, the precision, recall, and f- score depends on the factor true positives parameter, which shows the metric values 0. In this model, there are false negatives (Type 2 error) which shows the model was impacting on the data points that were actually "Progressive" but predicting them as "Non-Progressive". Furthermore, on the other hand, the training accuracy was 80%, and testing accuracy was 81% which shows that the model was neither overfitting nor underfitting. It indicates that the model has low bias and high variance (i.e., model was learning noise present in the data we had).

Decision Tree

Figure 3.5 depicts the confusion matrix for the **Decision Tree** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the decision tree's metric values. Calculation of the test dataset on 31 new data points gives us the decision tree's metric values.

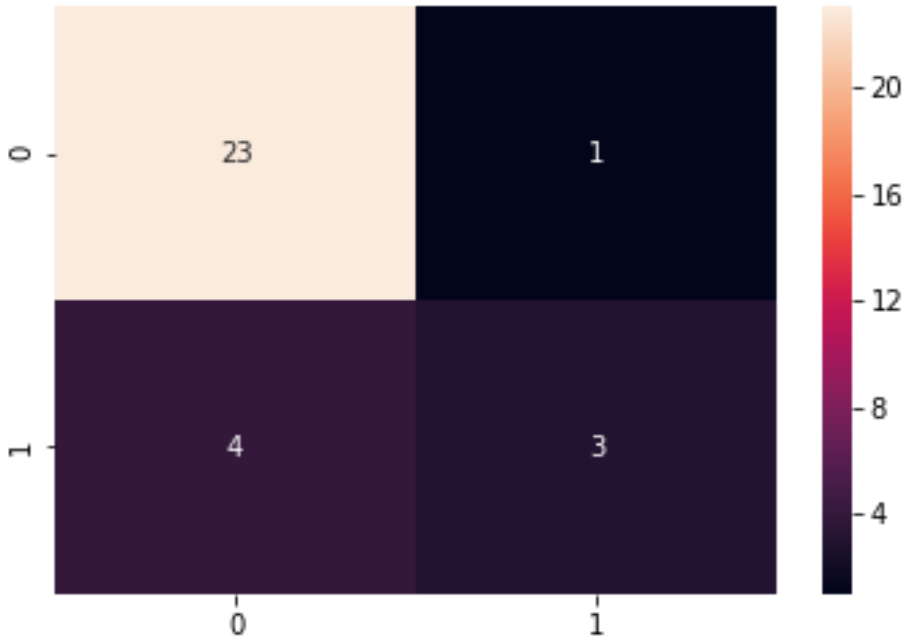


Figure 3.5 Confusion Matrix for Decision Tree

1. Accuracy score: 83.87
2. Precision score: 0.75

3. Recall score: 0.43
4. f1_score: 0.55

In this model there are both Type 1, and Type 2 errors and the total number of true predictions were 26 out of 31 data points, which was better than the previous model Logistic regression. This model's training accuracy and testing accuracies were 100% and 84% respectively, clearly showing that the model is overfitting. It means that the model could perform poorly on any new random data point.

KNN Classifier

Figure 3.6 depicts the confusion matrix for the **KNN Classifier** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the KNN's metric values. Calculation of the test dataset on 31 new data points gives us the KNN's metric values.

1. Accuracy score: 83.87
2. Precision score: 1.00
3. Recall score: 0.29
4. f1_score score: 0.44

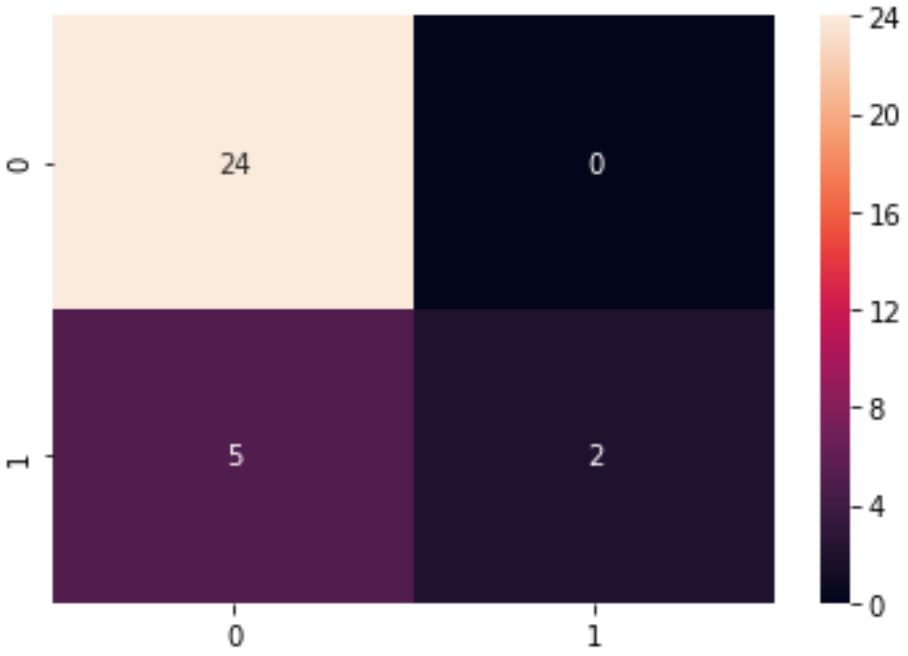


Figure 3.6 Confusion Matrix for KNN Classifier algorithm

In this model there are both Type 1, and Type 2 errors and the total number of true predictions were 26 out of 31 data points, which is remarkably similar to the above decision tree classifier. The training accuracy and testing accuracy for the KNN model is 84% and 84% respectively, showing that the model is neither overfitting nor underfitting.

Random Forest Classifier

Figure 3.7 depicts the confusion matrix for the **Random Forest Classifier** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the random forest's metric values. Calculation of the test dataset on 31 new data points gives us the random forest's metric values.

1. Accuracy score: 83.87
2. Precision score: 1.00
3. Recall score: 0.29
4. f1_score score: 0.44

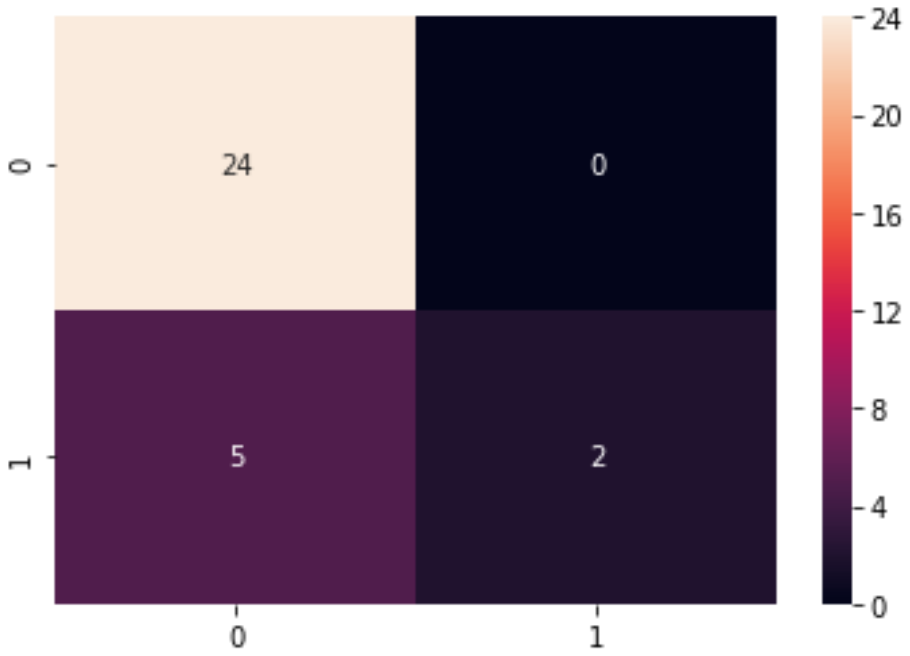


Figure 3.7 Confusion Matrix for Random Forest Classifier algorithm

In this model there are both Type 1, and Type 2 errors and the total number of true predictions were 26 out of 31 data points, which is remarkably similar to the above decision tree classifier. The training accuracy and testing accuracy for the random forest model is 98% and 84% respectively, clearly showing it is overfitting. It means that the model could perform poorly on any new random data point. The best accuracy calculation is chosen after trying out with different estimator values. In the random forest classifier algorithm, various hyperparameters values chosen are overfitting. However, the hyperparameter with the value ten (estimator = 10) might improve the model's performance if the data size is extensive and also making sure to balance the dataset.

The overall accuracy score bar plot for each classifier algorithm without k-fold cross validation is shown in figure 3.8.

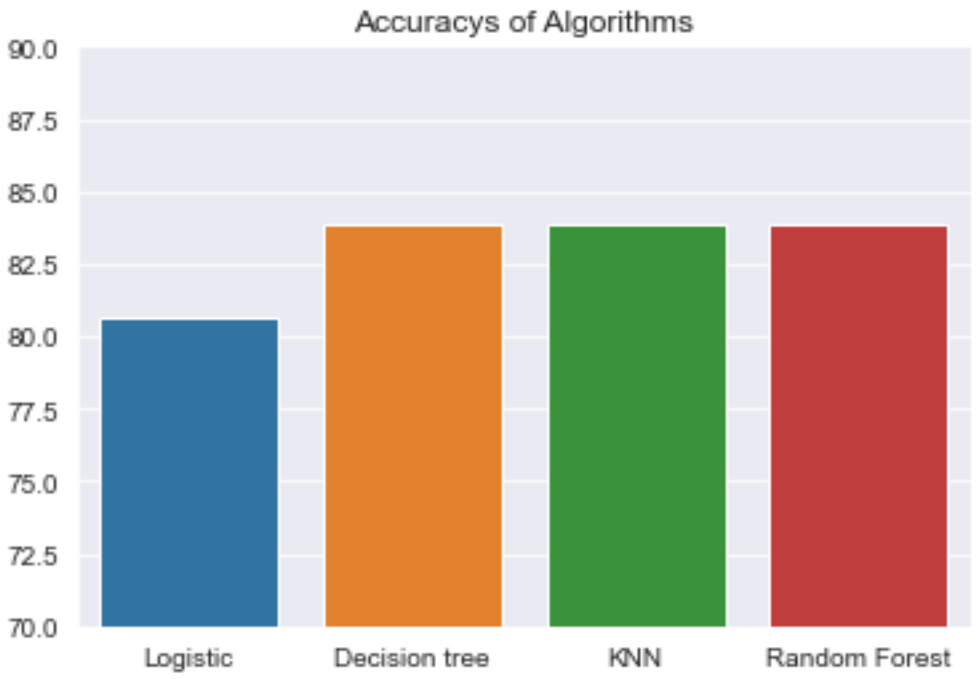


Figure 3.8 Bar plot of accuracy for four classifier algorithms

Accuracy of all the classifiers after K-fold cross validation

When we have our data size extremely limited, splitting the data into training and test sample will leave us with a minimal dataset. With a small dataset, if we do an 80:20 split ratio, we will get 20% of our test data size, and it is not enough. In this case, the model might overfit, or by chance, it would bring us out an exceedingly high accuracy, but in real life, the model might not be that good enough. It could be even worse if there are more classes or categories of a dependent variable. For example, suppose if there are 10 data points related to 10 different classes, we have one specific label class for one data point. Based on this, we cannot make any conclusions given by the model.

If we use k-fold cross-validation, we can build K different models and make predictions on our data. We can divide our data based upon the folds we specify. We can use the whole data for both training and testing to evaluate or validate our models by performing cross-validation.

Suppose let us say that we will be getting five different accuracies when we specify there are five-folds. If the accuracies were in a similar range, we could say that the model is consistent. If the accuracies were not in a similar range, then it is evident that the model is not efficient.

Accuracies of models after k-fold cross validation

So, we had divided the data into 15 folds for the training and testing.

Logistic Regression accuracy:

[0.81818182 0.90909091 0.90909091 0.81818182 0.8 0.8

0.5 0.9 0.9 0.9 0.7 0.7

0.8 0.7 0.7]

So, the average accuracy of all the 15 folds is 79.03%.

Decision Tree accuracy:

[0.72727273 0.72727273 1. 0.81818182 0.8 0.8

0.6 0.6 0.6 0.6 0.7 0.8

0.7 0.7 1.]

So, the average accuracy of all the 15 folds is 74.48%.

KNN accuracy:

[0.81818182 0.90909091 0.72727273 0.63636364 0.5 0.9

0.9 0.8 0.4 0.6 1. 0.9

0.6 0.8 0.7]

So, the average accuracy of all the 15 folds is 74.60%.

Random Forest Accuracy:

[0.90909091 0.63636364 0.72727273 1. 0.6 1.

0.8 1. 0.7 0.7 0.5 0.8

0.7 0.8 0.9]

So, the average accuracy of all the 15 folds is 78.48 %.

The overall accuracy score bar plot for each classifier algorithm after k-fold cross validation is shown in figure 3.9.

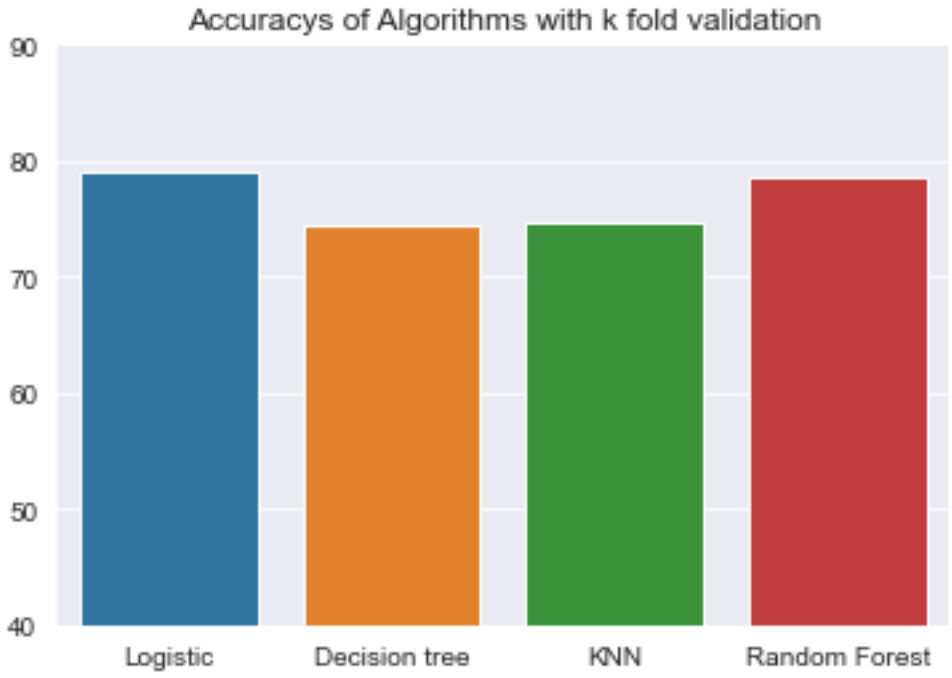


Figure 3.9 Bar plot of accuracy for four classifier algorithms after k-fold cross-validation

After removal of correlated column feature (Removal of Torsion)

It validates the models after removing the correlated features to understand how the independent variable's absence will alter the accuracy score.

Logistic Regression

Figure 3.10 depicts the confusion matrix for the **Logistic Regression** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the logistic regression's metric values. Calculation of the test dataset on 31 new data points gives us the logistic regression's metric values.

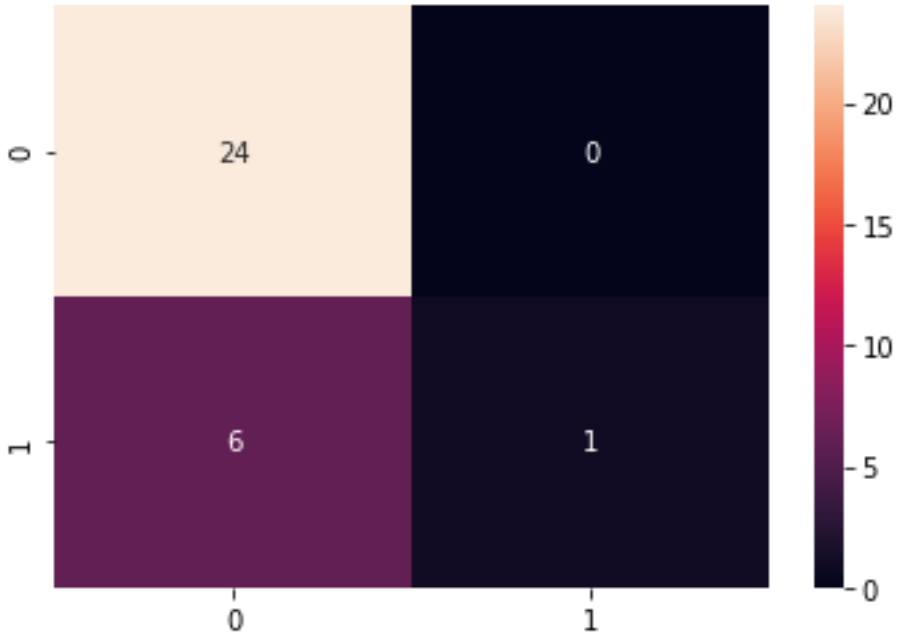


Figure 3.10 Confusion Matrix for Logistic Regression

1. Accuracy score: 80.65
2. Precision score: 1.00
3. Recall score: 0.14

4. f1_score score: 0.25

Since the model shows, the value for true positives is 0 (the true predictions were 0 for the Progressive class). The precision, recall, and f-score depend on the factor true positives parameter, which shows the metric values 0. In this model, there are false negatives (Type 2 error) which shows the model was impacting on the data points that were actually "Progressive" but predicting them as "Non-Progressive". Furthermore, on the other hand, the training accuracy was 80%, and testing accuracy was 81%, showing that the model is neither overfitting nor underfitting.

Decision Tree Classifier

Figure 3.11 depicts the confusion matrix for the **Decision Tree** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the decision tree's metric values. Calculation of the test dataset on 31 new data points gives us the decision tree's metric values.

1. Accuracy score: 80.65
2. Precision score: 0.57
3. Recall score: 0.57
4. f1_score score: 0.57

In this model, there are both Type 1 and Type 2 errors, and the total number of true predictions is 24 out of 31 data points". On the other hand, the training accuracy was 100%, and testing accuracy was 81% which shows that the model was overfitting. The number of true predictions decreased slightly compared to the model without removing the correlated feature.

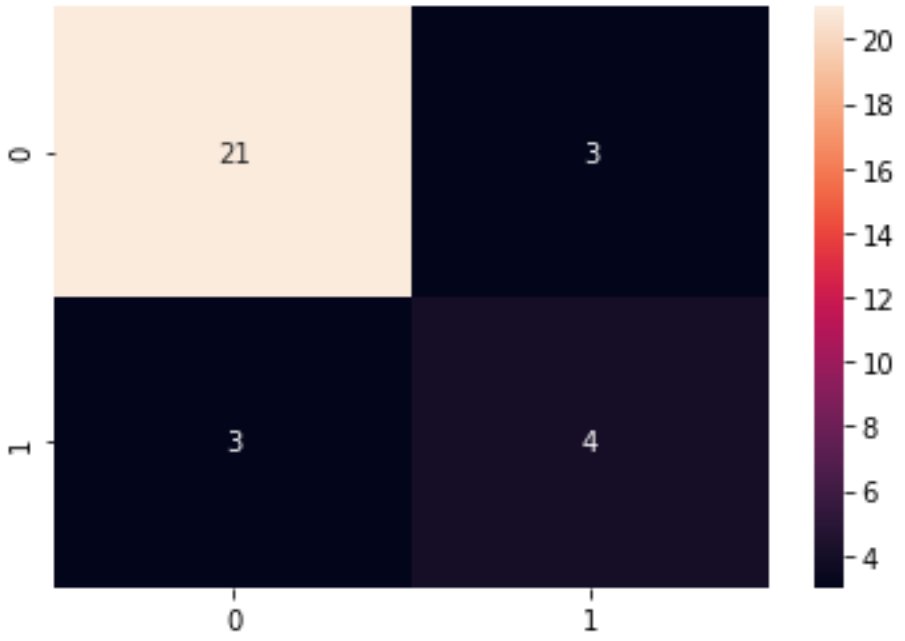


Figure 3.11 Confusion Matrix for Decision Tree

KNN Classifier

Figure 3.12 depicts the confusion matrix for the **KNN Classifier** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the KNN's metric values. Calculation of the test dataset on 31 new data points gives us the KNN's metric values.

1. Accuracy score: 83.87
2. Precision score: 1.00
3. Recall score: 0.29

4. f1_score score: 0.44

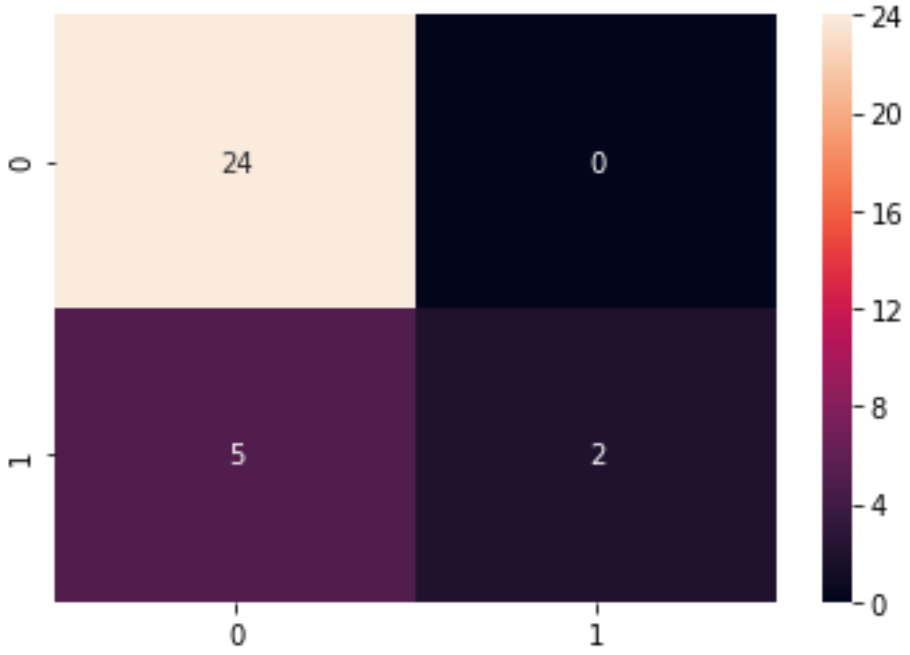


Figure 3.12 Confusion Matrix for KNN Classifier

In this model, both Type 1 and Type 2 errors and the total number of true predictions were 26 out of 31 data points. The training accuracy and the testing accuracy for the KNN model is 85% and 84%, which shows that the model is neither overfitting nor underfitting.

Random Forest Classifier

Figure 3.13 depicts the confusion matrix for the **Random Forest Classifier** algorithm.

- True Negatives (top left): cm (0,0)
- True Positives (bottom right): cm (1,1)
- False Negatives (bottom left): cm (0,1)
- False Positives (top right): cm (1,0)

The values below depict the random forest's metric values. Calculation of the test dataset on 31 new data points gives us the random forest's metric values.

1. Accuracy score: 80.65
2. Precision score: 0.67
3. Recall score: 0.29
4. f1_score score: 0.40

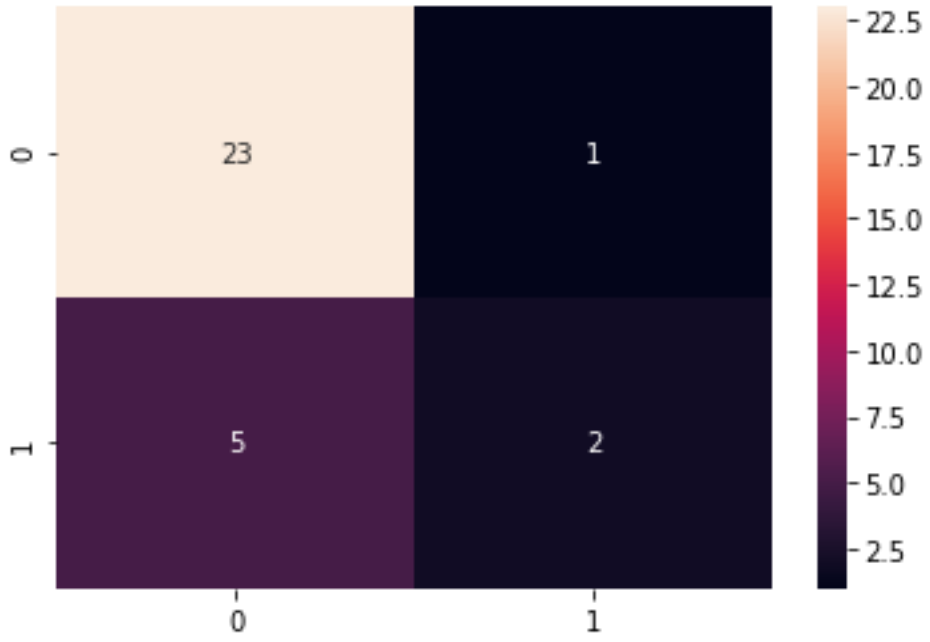


Figure 3.13 Confusion Matrix for Random Forest Classifier

In this model, there are both Type 1 and Type 2 errors. The total number of true predictions was 25 out of 31 data points. The training accuracy and testing accuracy for the random forest model is 98% and 81%, clearly showing that it was overfitting. In the random forest classifier algorithm, various hyperparameters values chosen are overfitting. However, the hyperparameter with the value ten (estimator = 10) might improve the model's performance if the data size is extensive and also making sure to balance the dataset.

The overall accuracy score bar plot for each classifier algorithm without k-fold cross validation is shown in figure 3.14. So, there is nothing much change in all four models' performance with removing the correlated feature. The prime reasons might be due to an imbalanced dataset and lower data size.

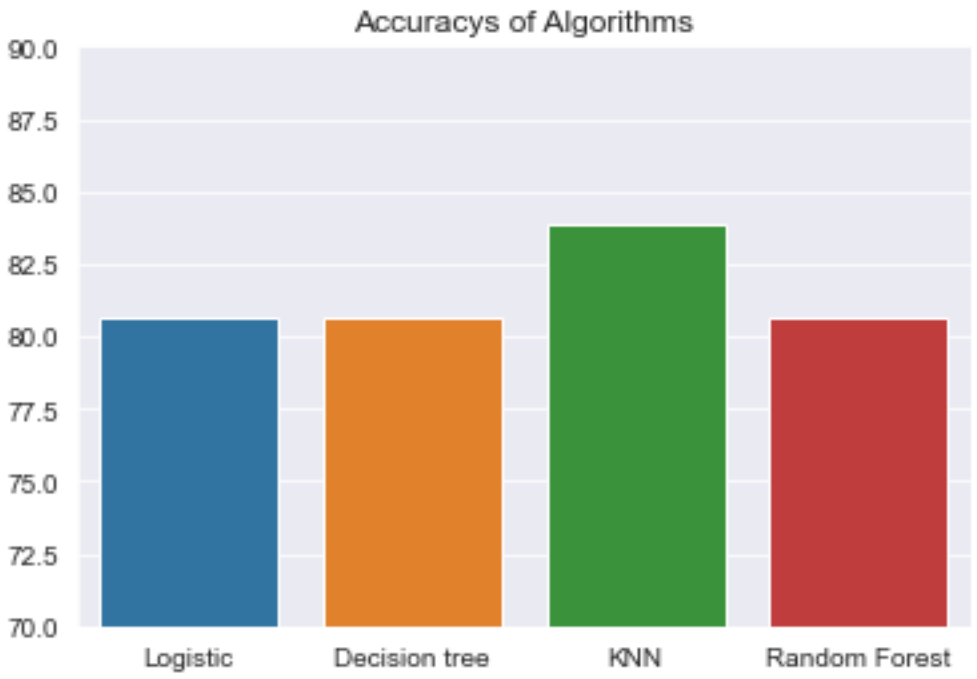


Figure 3.14 Bar plot of accuracy for four classifier algorithms

Accuracy of all the classifiers after K-fold cross validation

Logistic Regression accuracy:

[1. 0.72727273 0.81818182 0.81818182 0.9 0.8

0.7 0.8 0.9 0.7 0.6 0.8

0.7 0.9 0.8]

So, the average accuracy of all the 15 folds is 79.75 %.

Decision Tree accuracy:

[0.72727273 0.63636364 0.72727273 0.63636364 0.8 0.7

0.5 0.7 0.8 0.7 0.7 0.8

0.7 0.9 0.5]

So, the average accuracy of all the 15 folds is 70.18 %.

KNN accuracy:

[0.81818182 0.90909091 0.72727273 0.72727273 0.6 0.9

0.7 0.6 0.8 0.9 0.8 0.6

1. 0.6 0.8]

So, the average accuracy of all the 15 folds is 76.54 %.

Random Forest accuracy:

[1. 0.72727273 0.81818182 0.72727273 0.8 0.9

0.8 0.8 0.6 0.6 1. 0.7

0.8 0.7 0.8]

So, the average accuracy of all the 15 folds is 78.48 %. The overall accuracy score bar plot for each classifier algorithm after k-fold cross validation is shown in figure 3.15.

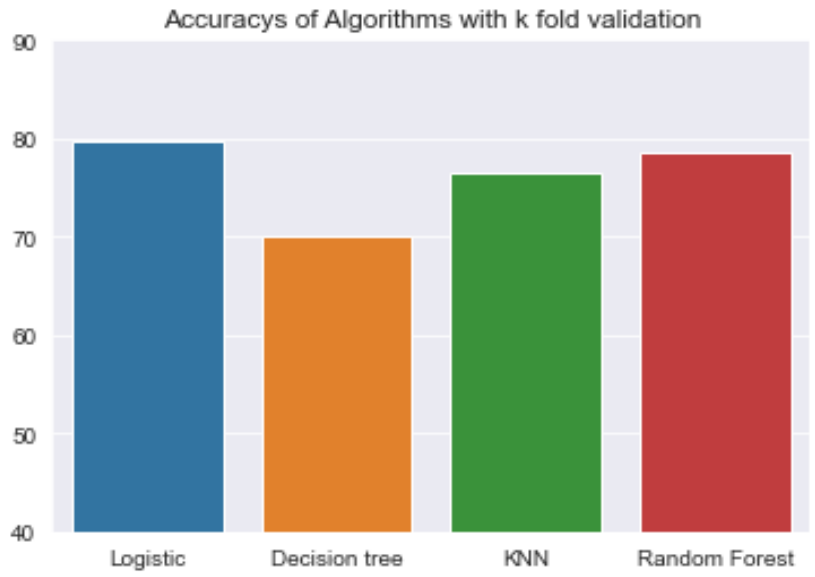


Figure 3.15 Bar plot of accuracy for four classifier algorithms after k-fold cross-validation

4. Conclusion

We have calculated the accuracies for the models with and without correlated feature. We also observe the performance of the models after the k fold cross-validation.

From table 4.1, we can observe the training and testing accuracies of Logistic Regression for optimal regularization wights, KNN for optimal neighbours, Random Forest for optimal estimators and Decision Tree. Out of all models, Logistic Regression & KNN models is neither overfitting nor underfitting. However, out of those two models, KNN is performing well compared to the Logistic regression.

Table 4.1 Training and Testing Accuracies

S.No	Supervised Classification Algorithm	After Removal of Correlated Independent Variable		Before Removal of Correlated Independent Variable	
		Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy
1.	Logistic Regression	80%	81%	80%	81%
2.	Decision Tree	100%	81%	100%	84%
3.	K-Nearest Neighbours	85%	84%	84%	84%
4.	Random Forest	98%	81%	98%	84%

Table 4.2 shows the accuracy score of each model from the confusion matrix with and without k-fold cross-validation. Even with K fold cross-validation, the model performance was not consistent and not

performing well. The prime reasons might be the lack of data given (lower data size) and the imbalanced dataset.

Table 4.2 Accuracy after and before k-fold cross-validation

S.No	Supervised Classification Algorithm	After Removal of Correlated Independent Variable		Before Removal of Correlated Independent Variable	
		Accuracy before k-fold	Accuracy after k-fold	Accuracy before k-fold	Accuracy after k-fold
1.	Logistic Regression	80.65% ≈ 81%	79.75% ≈ 80%	80.65% ≈ 81%	79.03% ≈ 79%
2.	Decision Tree	80.65% ≈ 81%	70.18% ≈ 70%	83.87% ≈ 84%	74.48% ≈ 75%
3.	K-Nearest Neighbours	83.87% ≈ 84%	76.54% ≈ 77%	83.87% ≈ 84%	74.60%≈ 75%
4.	Random Forest	80.65% ≈ 81%	78.48% ≈ 79%	83.87% ≈ 84%	78.48% ≈ 79%

We can even enhance the model performance with certain assumptions like collecting more data and converting the imbalanced dataset. In order to transform an imbalanced dataset into a balanced dataset, there are two methods.

- Random oversampling
- Random undersampling

Random oversampling: It always looks for increasing the instances for minority class data points randomly by re-creating or replicating the data points. [18]

Random Undersampling: It always tries to balance the distribution by eliminating the majority class data points randomly. [18]

However, due to the lack of data given, either of the strategies would not help solve this problem.

5. REFERENCES

[1] H. H. Publishing, "Scoliosis," *Harvard Health*. [Online]. Available: https://www.health.harvard.edu/a_to_z/scoliosis-a-to-z. [Accessed: 09-Apr-2021].

[2] "Scoliosis," *AANS*. [Online]. Available: https://www.aans.org/Patients/Neurosurgical-Conditions-and-Treatments/Scoliosis#:~:text=Scoliosis%20affects%202%2D3%20percent,occurring%20equally%20among%20both%20genders. [Accessed: 09-Apr-2021].

[3] Louise, "5 Fast Facts about Scoliosis," *Life with Scoliosis*, 23-Oct-2020. [Online]. Available: https://lifewithscoliosis.com/2020/06/5-fast-facts-about-scoliosis/. [Accessed: 09-Apr-2021].

[4] H. for S. Surgery and Hospital-For-Special-Surgery, "Scoliosis Infographic," *Issuu*. [Online]. Available: https://issuu.com/hospital-for-special-surgery/docs/spine-1. [Accessed: 09-Apr-2021].

[5] "Cobb's angle," *Physiopedia*. [Online]. Available: https://www.physio-pedia.com/Cobb%27s_angle. [Accessed: 09-Apr-2021].

[6] J. Yang, K. Zhang, H. Fan, Z. Huang, Y. Xiang, J. Yang, L. He, L. Zhang, Y. Yang, R. Li, Y. Zhu, C. Chen, F. Liu, H. Yang, Y. Deng, W. Tan, N. Deng, X. Yu, X. Xuan, X. Xie, X. Liu, and H. Lin, "Development and validation of deep learning algorithms for scoliosis screening using back images," *Communications Biology*, vol. 2, no. 1, 2019.

[7] H. Wu, J. L. Ronsky, P. Poncet, F. Cheriet, X. Deyi, J. A. Harder, and R. F. Zernicke, "Prediction of Scoliosis Progression in Time Series Using a Hybrid Learning Technique," *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005.

[8] A. Blum. Machine learning theory. Essay, 2007.

[9] “Top 10 Algorithms every Machine Learning Engineer should know,” *GeeksforGeeks*, 20-Aug-2019. [Online]. Available: https://www.geeksforgeeks.org/top-10-algorithms-every-machine-learning-engineer-should-know/. [Accessed: 09-Apr-2021].

[10] B. Shetty, “An in-depth guide to supervised machine learning classification,” *Built In*. [Online]. Available: https://builtin.com/data-science/supervised-machine-learning-classification. [Accessed: 09-Apr-2021].

[11] R. Garg, “7 Types of Classification Algorithms,” *Analytics India Magazine*, 08-Dec-2020. [Online]. Available: https://analyticsindiamag.com/7-types-classification-algorithms/. [Accessed: 09-Apr-2021].

[12] Datasciencelovers, “K-Nearest Neighbors (KNN) – Theory,” *datasciencelovers.com*, 28-Mar-2020. [Online]. Available: http://www.datasciencelovers.com/machine-learning/k-nearest-neighbors-knn-theory/. [Accessed: 09-Apr-2021].

[13] T. S. T. Srivastava, “K Nearest Neighbor: KNN Algorithm: KNN in Python & R,” *Analytics Vidhya*, 18-Oct-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/. [Accessed: 09-Apr-2021].

[14] Datasciencelovers, “Logistic Regression-Theory,” *datasciencelovers.com*, 01-Dec-2019. [Online]. Available: http://www.datasciencelovers.com/machine-learning/logistic-regression-theory/. [Accessed: 09-Apr-2021].

[15] Datasciencelovers, “Random Forest-Theory,” *datasciencelovers.com*, 04-Jan-2020. [Online]. Available: http://www.datasciencelovers.com/machine-learning/random-forest-theory/. [Accessed: 09-Apr-2021].

[16] Datasciencelovers, “Decision Tree - Theory,” *datasciencelovers.com*, 23-Dec-2019. [Online]. Available: http://www.datasciencelovers.com/machine-learning/decision-tree-theory/. [Accessed: 09-Apr-2021].

[17] S. Sareen, "Confusion Matrix," *Medium*, 08-Mar-2020. [Online]. Available: https://medium.com/@shivangisareen/confusion-matrix-3ac02a1719ba. [Accessed: 09-Apr-2021].

[18] J. Brownlee, "Random Oversampling and Undersampling for Imbalanced Classification," *Machine Learning Mastery*, 04-Jan-2021. [Online]. Available: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/. [Accessed: 09-Apr-2021].

Appendix A

ECE 710 - Wearable Device Technologies and IoT

Project 3 - Model to predict the Progression or Non-Progression of Scoliosis

Done by

Rithvik Ramesh (1654067)

Srija Piratla (1651539)

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix as cm
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
```

Reading Excel file

In [2]:
```
df = pd.read_excel(r'C:\Users\Rithvik Ramesh\Desktop\University of Alberta - Winter 2021\ECE 710\Project 3\Data fro F
```

Displaying top 100 rows

In [3]:
```
df.head(100) # displays the 100 rows from data set in excel sheet
```

Out[3]:

	1st ID	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
0	SRS85	NP	-8.0	1.7	2.525000	0.099778	20.0
1	SRS87	NP	-10.0	-3.1	-1.033333	0.086335	35.0
2	US17	NP	8.0	-4.2	0.000000	0.077022	24.0
3	US83	NP	1.0	4.4	2.266667	0.091409	14.0
4	US19	NP	7.0	7.9	3.366667	0.059883	38.0
...	...	...	...	...	...	...	...
95	US530	NP	1.0	3.0	1.720000	0.059461	11.0
96	US541	NP	5.0	3.3	2.350000	0.062135	30.0
97	US560	NP	-9.0	11.2	3.200000	0.063483	36.0
98	US561	P	6.0	-4.9	-2.033333	0.031209	30.0
99	US576	NP	-6.0	-11.6	-4.640000	0.058261	20.0

100 rows × 7 columns

Deletion of 1st ID column

In [4]:
```
del df['1st ID'] # removed 1st column
```

In [5]:
```
df.head(10) # displays the 10 rows from data set in excel sheet
```

Out[5]:

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
0	NP	-8.0	1.7	2.525000	0.099778	20.0
1	NP	-10.0	-3.1	-1.033333	0.086335	35.0
2	NP	8.0	-4.2	0.000000	0.077022	24.0
3	NP	1.0	4.4	2.266667	0.091409	14.0

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
4	NP	7.0	7.9	3.366667	0.059883	38.0
5	P	11.0	4.2	1.400000	0.105570	29.0
6	NP	-1.0	11.5	3.833333	0.069357	32.0
7	NP	6.0	11.2	3.200000	0.077335	42.0
8	NP	-12.0	1.1	0.314286	0.083830	22.0
9	NP	-7.0	-4.7	3.160000	0.065243	38.0

Describing the data

In [6]:
```
df.describe() # displays statistical distribution/statistical values of the data set in excel sheet
```

Out[6]:

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
count	154.000000	155.000000	155.000000	155.000000	155.000000
mean	-0.006494	4.403871	1.475030	0.071199	25.000000
std	6.147954	7.994218	2.987428	0.023968	9.346157
min	-16.000000	-14.000000	-6.960000	0.013752	9.000000
25%	-3.750000	-1.300000	-0.850000	0.055198	18.000000
50%	0.500000	3.600000	2.266667	0.067117	24.000000
75%	4.000000	10.050000	3.557143	0.088470	32.000000
max	17.000000	26.500000	7.571429	0.130443	49.000000

Displaying out the count of missing values

In [7]:
```
df.isnull().sum() # checking out the missing values
```

Out[7]:
```
Output (Progression (P)/Nonprogression (NP)    6
```

```
US Cobb Change                              9
Apical Axial Vertebral rotation (AVR)       8
Torsion                                     8
RC value                                    8
Curve magnitude                             8
dtype: int64
```

Dropping down all the missing values

In [8]:
```
df1 = df.dropna() # removing missing values
```

In [9]:
```
df1.head(100) # displays the 100 rows from data set in excel sheet after dropping down all missing values
```

Out[9]:

	Output (Progression (P)/Nonprogression (NP))	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
0	NP	-8.0	1.7	2.525000	0.099778	20.0
1	NP	-10.0	-3.1	-1.033333	0.086335	35.0
2	NP	8.0	-4.2	0.000000	0.077022	24.0
3	NP	1.0	4.4	2.266667	0.091409	14.0
4	NP	7.0	7.9	3.366667	0.059883	38.0
...	...	...	...	...	...	...
96	NP	5.0	3.3	2.350000	0.062135	30.0
97	NP	-9.0	11.2	3.200000	0.063483	36.0
98	P	6.0	-4.9	-2.033333	0.031209	30.0
99	NP	-6.0	-11.6	-4.640000	0.058261	20.0
100	NP	-1.0	15.7	5.233333	0.065022	31.0

100 rows × 6 columns

In [10]:
```
df1.describe() # displays statistical distribution/statistical values of the data set in excel sheet after dropping d
```

Out[10]:

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
count	154.000000	154.000000	154.000000	154.000000	154.000000
mean	-0.006494	4.442208	1.496415	0.071266	25.006494
std	6.147954	8.005993	2.985249	0.024032	9.376299
min	-16.000000	-14.000000	-6.960000	0.013752	9.000000
25%	-3.750000	-1.275000	-0.745833	0.055077	18.000000
50%	0.500000	3.900000	2.283333	0.067180	24.000000
75%	4.000000	10.125000	3.592857	0.088607	32.000000
max	17.000000	26.500000	7.571429	0.130443	49.000000

Replacing the dependent variables Output (Progression (P) as "1" /Nonprogression (NP) as "0"

In [11]:
```
df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_rep
                value = 0) # replaces Nonprogression (NP) to 0

df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_rep
                value = 1) # replaces Progression (P) to 1
```

```
<ipython-input-11-55b38457e085>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_r
eplace ="NP",
<ipython-input-11-55b38457e085>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
```

```
  df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_r
eplace ="P",
```

In [12]:
```
df1.head(10) # displays the 10 rows from data set in excel sheet after replacing Replacing the dependent variables Ou
```

Out[12]:

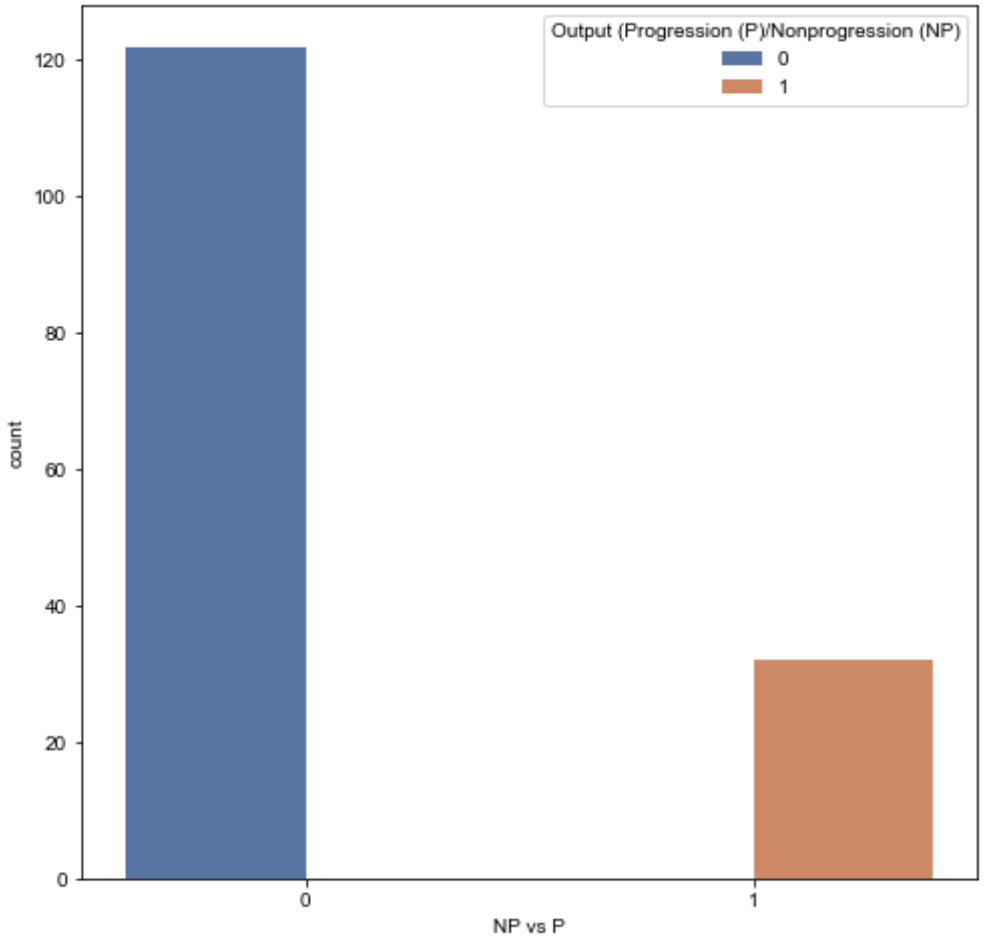
	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
0	0	-8.0	1.7	2.525000	0.099778	20.0
1	0	-10.0	-3.1	-1.033333	0.086335	35.0
2	0	8.0	-4.2	0.000000	0.077022	24.0
3	0	1.0	4.4	2.266667	0.091409	14.0
4	0	7.0	7.9	3.366667	0.059883	38.0
5	1	11.0	4.2	1.400000	0.105570	29.0
6	0	-1.0	11.5	3.833333	0.069357	32.0
7	0	6.0	11.2	3.200000	0.077335	42.0
8	0	-12.0	1.1	0.314286	0.083830	22.0
9	0	-7.0	-4.7	3.160000	0.065243	38.0

Count of dependent variable (Progression (P)/Nonprogression (NP))

In [13]:
```
df1["Output (Progression (P)/Nonprogression (NP)"].value_counts() # counts the dependent variable or output
```

Out[13]:
```
0    122
1     32
Name: Output (Progression (P)/Nonprogression (NP), dtype: int64
```

In [14]:
```
plt.figure(figsize = (8,8)) # Size of the figure
sns.countplot(x=df1["Output (Progression (P)/Nonprogression (NP)"], hue=df1["Output (Progression (P)/Nonprogression (
plt.xlabel("NP vs P") # x-axis label name
sns.set_style('darkgrid') # sets style of grid
plt.show() # displays plot
```



Splitting the dependent and independent variables.

In [15]:
```
X= df1.iloc[:,1:6] # splits the input
y = df1.iloc[:,0:1] # splits the output
X.head(10) # displays the 10 rows after splitting dependent and independent variables
```

Out[15]:

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
0	-8.0	1.7	2.525000	0.099778	20.0

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
1	-10.0	-3.1	-1.033333	0.086335	35.0
2	8.0	-4.2	0.000000	0.077022	24.0
3	1.0	4.4	2.266667	0.091409	14.0
4	7.0	7.9	3.366667	0.059883	38.0
5	11.0	4.2	1.400000	0.105570	29.0
6	-1.0	11.5	3.833333	0.069357	32.0
7	6.0	11.2	3.200000	0.077335	42.0
8	-12.0	1.1	0.314286	0.083830	22.0
9	-7.0	-4.7	3.160000	0.065243	38.0

Finding out the relationship between independent variable

In [16]:
```
df1.corr() # finds the relationship between independent variable
```

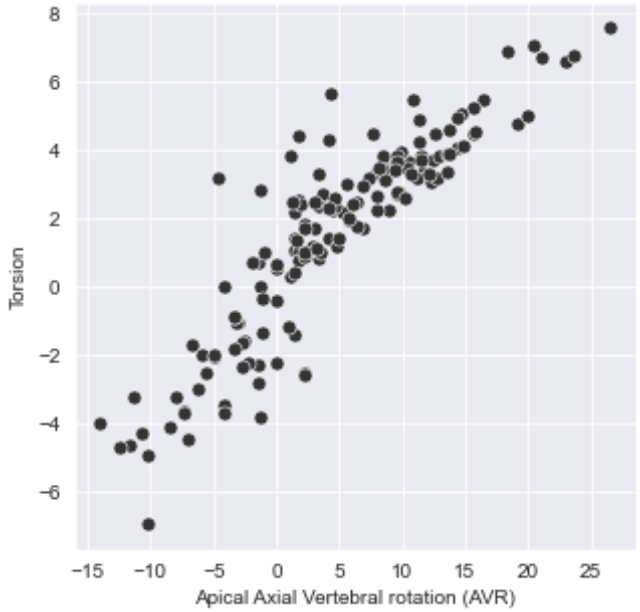
Out[16]:

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
Output (Progression (P)/Nonprogression (NP)	1.000000	0.397526	-0.063077	-0.064509	-0.107131	-0.024331
US Cobb Change	0.397526	1.000000	-0.134470	-0.082747	-0.059670	-0.028798
Apical Axial Vertebral rotation (AVR)	-0.063077	-0.134470	1.000000	0.898426	0.053471	0.187917
Torsion	-0.064509	-0.082747	0.898426	1.000000	0.085864	0.177991
RC value	-0.107131	-0.059670	0.053471	0.085864	1.000000	0.047044
Curve magnitude	-0.024331	-0.028798	0.187917	0.177991	0.047044	1.000000

Scatter plot for the co-related variables (Apical Axial Vertebral rotation

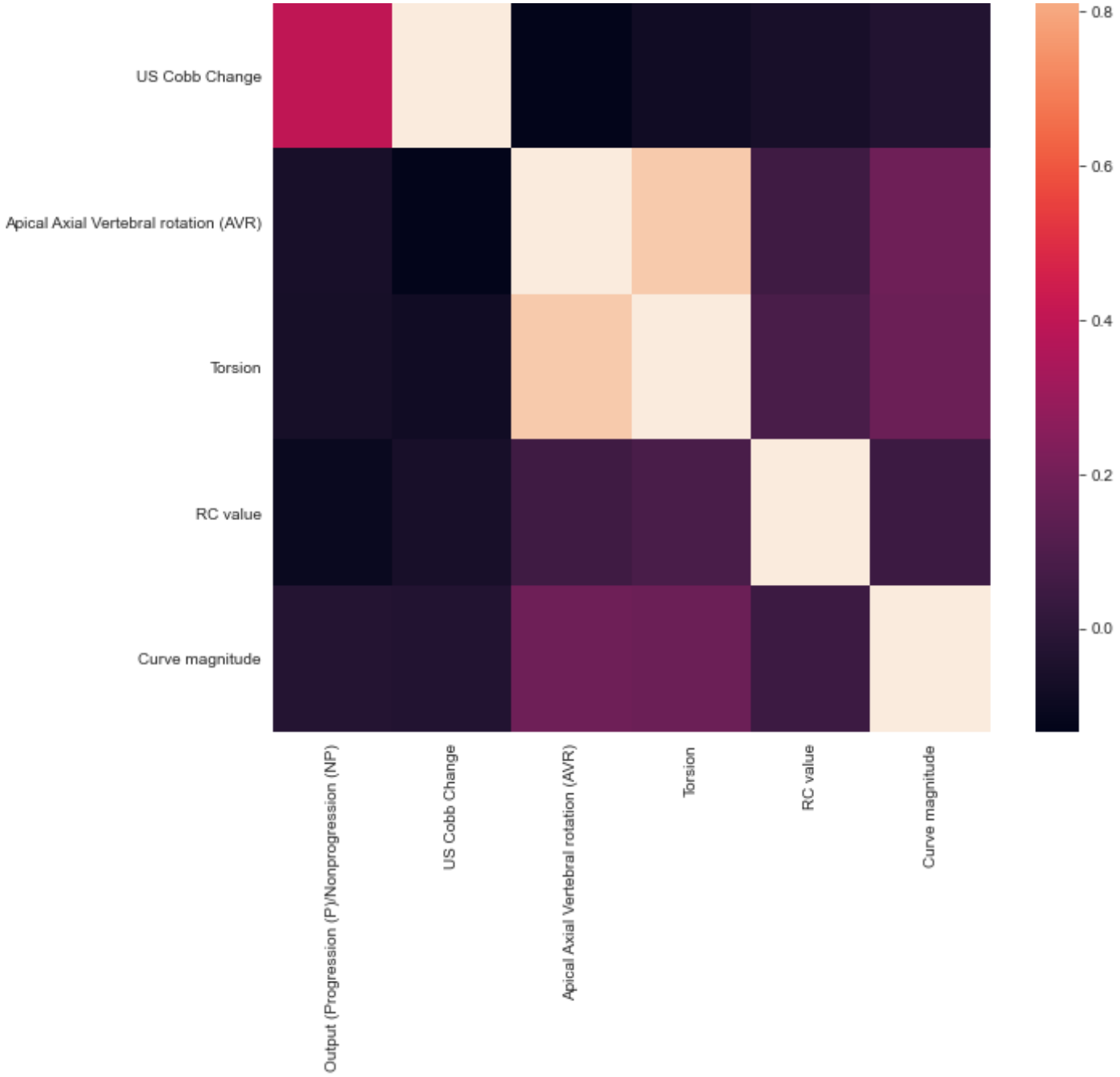
(AVR) and Torsion)

In [17]:
```
plt.figure(figsize = (5,5))
sns.scatterplot(x = df1['Apical Axial Vertebral rotation (AVR)'], y = df1['Torsion'], palette="deep",s=50, color=".2'
plt.xlabel('Apical Axial Vertebral rotation (AVR)')
plt.show()
```



Heatmap for correlation matrix

In [18]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df1.corr()) # plots the heatmap of correlation
plt.show()
```



Splitting the data into training and testing data

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20,random_state=27) # splits the data set into 80
```

Standardizing the values

In [20]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logistic Regression

In [21]:
```
c = [0.5, 1, 1.5, 2,3]
for i in range(len(c)):

    logreg = LogisticRegression(C = c[i])

    logreg.fit(X_train, y_train)
    print('estimators=' +  (str(c[i])) + ", " +  'Accuracy of Logistic Regression training set {:.2f}'
    .format(logreg.score(X_train, y_train)))
    print('estimator=' +  (str(c[i])) + ", " + 'Accuracy of Logistic Regression on test set {:.2f}'
    .format(logreg.score(X_test, y_test)))
```

```
estimators=0.5, Accuracy of Logistic Regression training set 0.80
estimator=0.5, Accuracy of Logistic Regression on test set 0.77
estimators=1, Accuracy of Logistic Regression training set 0.80
estimator=1, Accuracy of Logistic Regression on test set 0.77
estimators=1.5, Accuracy of Logistic Regression training set 0.80
estimator=1.5, Accuracy of Logistic Regression on test set 0.81
estimators=2, Accuracy of Logistic Regression training set 0.80
estimator=2, Accuracy of Logistic Regression on test set 0.84
estimators=3, Accuracy of Logistic Regression training set 0.80
estimator=3, Accuracy of Logistic Regression on test set 0.84
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
```

```
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
```

Optimal value for logistic regression

In [22]:

```
#Logistic Regression
logreg1 = LogisticRegression(C = 1.5)
logreg1.fit(X_train,y_train)
logreg_predict = logreg1.predict(X_test)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg1.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg1.score(X_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set: 0.80
Accuracy of Logistic regression classifier on test set: 0.81
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
```

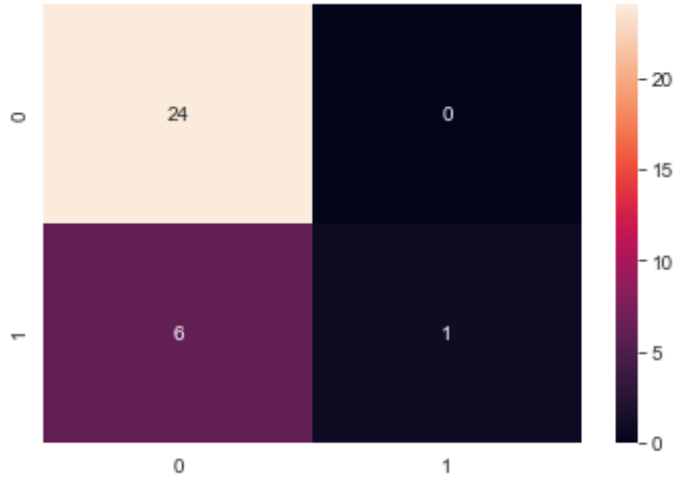
Confusion Matrix for Logistic Regression

In [23]:

```
predicted_labels_1 = logreg1.predict(X_test)# Logistic Rgeression
```

```
cf_matrix = cm(y_test, predicted_labels_1)
sns.heatmap(cf_matrix, annot=True)
```

Out[23]: <AxesSubplot:>



Printing out the Classiifcation Report for Logistic Regression

In [24]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_1)*100))
s1 = (accuracy_score(y_test, predicted_labels_1)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_1)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_1)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_1)))
```

```
Accuracy score : 80.65
Precision score : 1.00
Recall score : 0.14
f1_score score : 0.25
```

Descision Tree Classifier

In [25]:
```
#Descision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
```

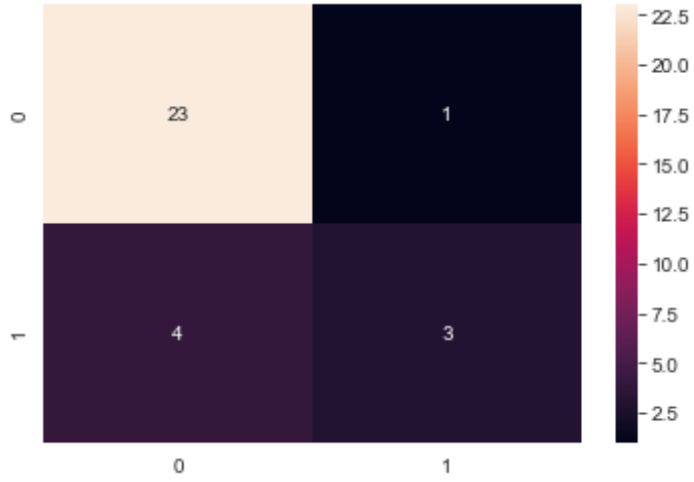
```
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.84
```

Confusion matrix for Decision Tree Classiifcation

In [26]:
```
predicted_labels_2 = clf.predict(X_test)# Descision Tree Classification
cf_matrix = cm(y_test, predicted_labels_2)
sns.heatmap(cf_matrix, annot=True)
```

Out[26]: <AxesSubplot:>



Printing out the classification report for Decision Tree Classiifcation

In [27]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_2)*100))
s2 = (accuracy_score(y_test, predicted_labels_2)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_2)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_2)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_2)))
```

```
Accuracy score : 83.87
Precision score : 0.75
Recall score : 0.43
f1_score score : 0.55
```

K neighbhors

In [28]:

```
# K neighbhors
n_neighbhor = [5, 10, 15, 20, 30]
from sklearn.neighbors import KNeighborsClassifier
for i in range(len(n_neighbhor)):

    knn = KNeighborsClassifier(n_neighbors=n_neighbhor[i])
    knn.fit(X_train, y_train)
    print( 'estimators=' +  str(n_neighbhor[i]) + ", " +  'Accuracy of KNN classifier on training set {:.2f}'
     .format(knn.score(X_train, y_train)))
    print( 'estimator=' +  str(n_neighbhor[i]) + ", " + 'Accuracy of KNN classifier on test set {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
estimators=5, Accuracy of KNN classifier on training set 0.84
estimator=5, Accuracy of KNN classifier on test set 0.84
estimators=10, Accuracy of KNN classifier on training set 0.80
estimator=10, Accuracy of KNN classifier on test set 0.81
estimators=15, Accuracy of KNN classifier on training set 0.80
estimator=15, Accuracy of KNN classifier on test set 0.77
estimators=20, Accuracy of KNN classifier on training set 0.80
estimator=20, Accuracy of KNN classifier on test set 0.77
estimators=30, Accuracy of KNN classifier on training set 0.80
estimator=30, Accuracy of KNN classifier on test set 0.77
```

```
<ipython-input-28-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-28-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-28-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-28-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-28-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
```

Optimal neighbhor value for KNN Classifier

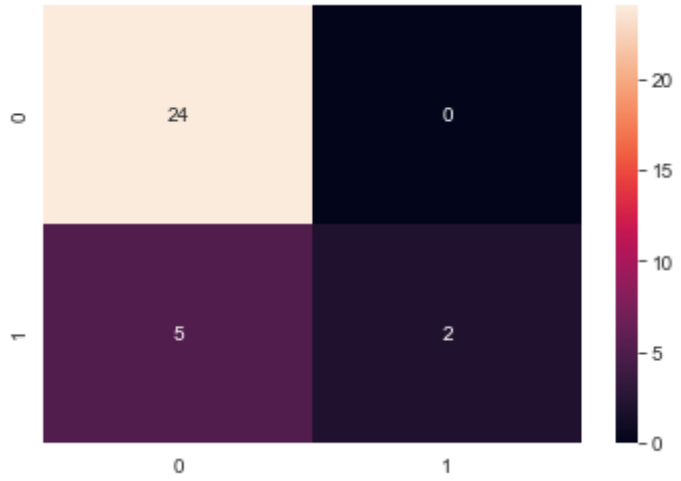
In [29]:
```
knn1 = KNeighborsClassifier(n_neighbors = 5)
knn1.fit(X_train,y_train)
knn1_predict = knn1.predict(X_test)
print(  'Accuracy of KNN classifier on training set {:.2f}'
     .format(knn1.score(X_train, y_train)))
print( 'Accuracy of KNN classifier on test set {:.2f}'
     .format(knn1.score(X_test, y_test)))
```

```
Accuracy of KNN classifier on training set 0.84
Accuracy of KNN classifier on test set 0.84
<ipython-input-29-f13be0dd2392>:2: 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().
  knn1.fit(X_train,y_train)
```

Confusion matrix for KNN

In [30]:
```
predicted_labels_3 = knn1.predict(X_test)# KNN
cf_matrix = cm(y_test, predicted_labels_3)
sns.heatmap(cf_matrix, annot=True)
```

Out[30]:
```
<AxesSubplot:>
```



Printing out the classification report for KNN

In [31]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_3)*100))
s3 = (accuracy_score(y_test, predicted_labels_3)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_3)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_3)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_3)))
```

```
Accuracy score : 83.87
Precision score : 1.00
Recall score : 0.29
f1_score score : 0.44
```

Random Forest Classifier

In [32]:
```
# random forest classifier
n_estimator = [10, 20, 30, 40]
from sklearn.ensemble import RandomForestClassifier
for i in range(len(n_estimator)):
    rfc = RandomForestClassifier(n_estimators=n_estimator[i], random_state= 1)
    rfc.fit(X_train,y_train)
    rfc_predict = rfc.predict(X_test)
    print( 'estimators=' +  str(n_estimator[i]) + ", " +  'Accuracy of Random Forest classifier on training set {:.2f
```

```
     .format(rfc.score(X_train, y_train)))
    print( 'estimator=' +  str(n_estimator[i]) + ", " + 'Accuracy of Random Forest classifier on test set {:.2f}'
     .format(rfc.score(X_test, y_test)))
```

```
<ipython-input-32-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
<ipython-input-32-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
<ipython-input-32-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
<ipython-input-32-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
estimators=10, Accuracy of Random Forest classifier on training set 0.98
estimator=10, Accuracy of Random Forest classifier on test set 0.84
estimators=20, Accuracy of Random Forest classifier on training set 0.99
estimator=20, Accuracy of Random Forest classifier on test set 0.81
estimators=30, Accuracy of Random Forest classifier on training set 0.99
estimator=30, Accuracy of Random Forest classifier on test set 0.81
estimators=40, Accuracy of Random Forest classifier on training set 1.00
estimator=40, Accuracy of Random Forest classifier on test set 0.84
```

Optimal estimator value for Random Forest Classifier

In [33]:

```
rfc1 = RandomForestClassifier(n_estimators=10, random_state= 1)
rfc1.fit(X_train,y_train)
rfc_predict = rfc1.predict(X_test)
print(  'Accuracy of Random Forest classifier on training set {:.2f}'
     .format(rfc1.score(X_train, y_train)))
print( 'Accuracy of Random Forest classifier on test set {:.2f}'
     .format(rfc1.score(X_test, y_test)))
```

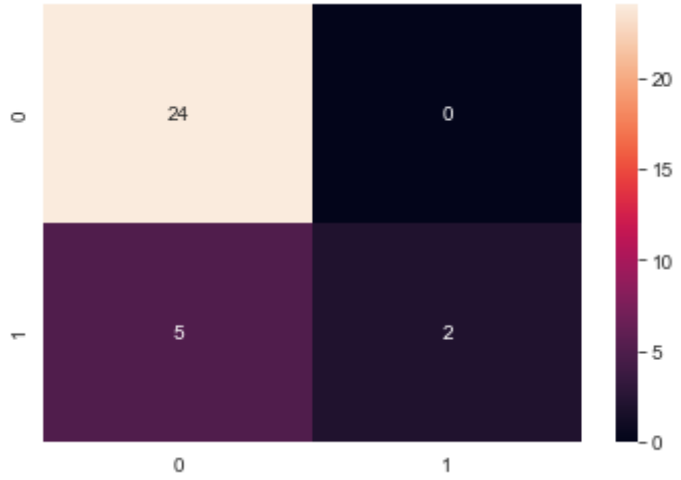
```
Accuracy of Random Forest classifier on training set 0.98
Accuracy of Random Forest classifier on test set 0.84
```

```
<ipython-input-33-d4001a358eef>:2: 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().
  rfc1.fit(X_train,y_train)
```

Confusion matrix for Random Forest Classifer

In [34]:
```
predicted_labels_4 = rfc1.predict(X_test)# RFC
cf_matrix = cm(y_test, predicted_labels_4)
sns.heatmap(cf_matrix, annot=True)
```

Out[34]: <AxesSubplot:>



Printing out the classification report for Random forest Classifier

In [35]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_4)*100))
s4 = (accuracy_score(y_test, predicted_labels_4)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_4)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_4)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_4)))
```

```
Accuracy score : 83.87
Precision score : 1.00
Recall score : 0.29
f1_score score : 0.44
```

K fold validation Logistic regression

In [36]:
```
# K fold validation Logistic regression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
result = cross_val_score(logreg1, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks1 = np.mean(result)
```

```
[0.81818182 0.90909091 0.90909091 0.81818182 0.8        0.8
 0.5        0.9        0.9        0.9        0.7        0.7
 0.8        0.7        0.7       ]
0.7903030303030303
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
```

```
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
```

K fold validation Decision Tree Classification

In [37]:

```
#  K fold validation Decision Tree Classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
result = cross_val_score(clf, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks2 = np.mean(result)
```

```
[0.72727273 0.72727273 1.         0.81818182 0.8        0.8
 0.6        0.6        0.6        0.6        0.7        0.8
```

```
 0.7        0.7        1.        ]
0.7448484848484846
```

K cross fold validation KNN

In [38]:

```
#K croos fold validation KNN
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
result = cross_val_score(knn1, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks3 = np.mean(result)
```

```
[0.81818182 0.90909091 0.72727273 0.63636364 0.5        0.9
 0.9        0.8        0.4        0.6        1.         0.9
 0.6        0.8        0.7       ]
0.7460606060606061
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: 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().
  estimator.fit(X_train, y_train, **fit_params)
```

Random Forest Classifier K fold cross validation

In [39]:

```
#Random Forest Classifier K fold cross validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
```

```
result = cross_val_score(rfc1, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks4 = np.mean(result)
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
```

```
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
[0.90909091 0.63636364 0.72727273 1.         0.6        1.
 0.8        1.         0.7        0.7        0.5        0.8
 0.7        0.8        0.9       ]
0.7848484848484849
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

Accuracies of four different models on scoliosis dataset

In [40]:

```
scores = [] # stores the list of accuracy score before k-fold
Algorithms = ['Logistic', 'Decision tree', 'KNN', 'Random Forest'] # stores the list of algorithm
```

In [41]:

```
scores.extend([s1,s2,s3,s4]) # adds the specified list elements
```

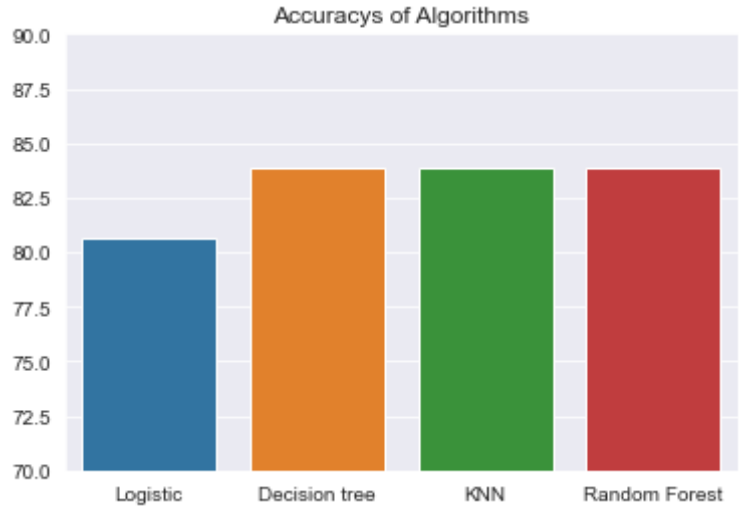
In [42]:

```
print(scores) # prints the scores
```

```
[80.64516129032258, 83.87096774193549, 83.87096774193549, 83.87096774193549]
```

In [43]:

```
plt.title('Accuracys of Algorithms')
sns.barplot(x = Algorithms, y = scores)
plt.ylim(70,90)
plt.show()
```



Accuracies of four different models on scoliosis dataset after k-fold validation

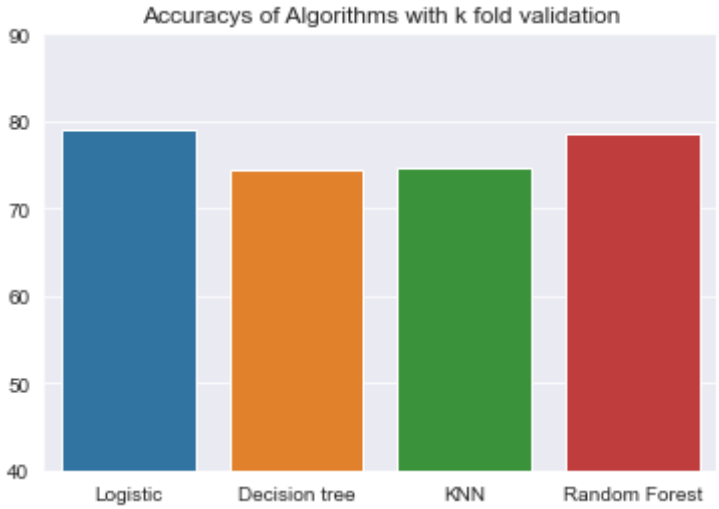
In [44]:
```
scores1 = [] # stores the list of accuracy score after k-fold
Algorithms = ['Logistic', 'Decision tree', 'KNN', 'Random Forest'] # stores the list of algorithm
```

In [45]:
```
scores1.extend([ks1*100,ks2*100,ks3*100,ks4*100]) # adds the specified list elements
```

In [46]:
```
print(scores1) # prints the scores
```

```
[79.03030303030303, 74.48484848484847, 74.60606060606061, 78.48484848484848]
```

In [47]:
```
plt.title('Accuracys of Algorithms with k fold validation')
sns.barplot(x = Algorithms, y = scores1)
plt.ylim(40,90)
plt.show()
```



Appendix B

ECE 710 - Wearable Device Technologies and IoT

Project 3 - Model to predict the Progression or Non-Progression of Scoliosis (After removal of Torsion)

Done by

Rithvik Ramesh (1654067)

Srija Piratla (1651539)

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix as cm
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

Reading Excel file

In [2]:
```
df = pd.read_excel(r'C:\Users\Rithvik Ramesh\Desktop\University of Alberta - Winter 2021\ECE 710\Project 3\Data fro F
```

Displaying top 100 rows

In [3]:
```
df.head(100)
```

Out[3]:

	1st ID	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	Torsion	RC value	Curve magnitude
0	SRS85	NP	-8.0	1.7	2.525000	0.099778	20.0
1	SRS87	NP	-10.0	-3.1	-1.033333	0.086335	35.0
2	US17	NP	8.0	-4.2	0.000000	0.077022	24.0
3	US83	NP	1.0	4.4	2.266667	0.091409	14.0
4	US19	NP	7.0	7.9	3.366667	0.059883	38.0
...	...	...	...	...	...	...	...
95	US530	NP	1.0	3.0	1.720000	0.059461	11.0
96	US541	NP	5.0	3.3	2.350000	0.062135	30.0
97	US560	NP	-9.0	11.2	3.200000	0.063483	36.0
98	US561	P	6.0	-4.9	-2.033333	0.031209	30.0
99	US576	NP	-6.0	-11.6	-4.640000	0.058261	20.0

100 rows × 7 columns

Deletion of 1st ID column

In [4]:
```
del df['1st ID'] # removed 1st column
```

In [5]:
```
del df['Torsion']# removed torsion column (co-orelated variable)
```

In [6]:
```
df.head(10)
```

Out[6]:

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
0	NP	-8.0	1.7	0.099778	20.0
1	NP	-10.0	-3.1	0.086335	35.0

	Output (Progression (P)/Nonprogression (NP))	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
2	NP	8.0	-4.2	0.077022	24.0
3	NP	1.0	4.4	0.091409	14.0
4	NP	7.0	7.9	0.059883	38.0
5	P	11.0	4.2	0.105570	29.0
6	NP	-1.0	11.5	0.069357	32.0
7	NP	6.0	11.2	0.077335	42.0
8	NP	-12.0	1.1	0.083830	22.0
9	NP	-7.0	-4.7	0.065243	38.0

Describing the data

In [7]:
```
df.describe()
```

Out[7]:

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
count	154.000000	155.000000	155.000000	155.000000
mean	-0.006494	4.403871	0.071199	25.000000
std	6.147954	7.994218	0.023968	9.346157
min	-16.000000	-14.000000	0.013752	9.000000
25%	-3.750000	-1.300000	0.055198	18.000000
50%	0.500000	3.600000	0.067117	24.000000
75%	4.000000	10.050000	0.088470	32.000000
max	17.000000	26.500000	0.130443	49.000000

Displaying out the count of missing values

In [8]:
```
df.isnull().sum() # checking out the missing values
```

Out[8]:
```
Output (Progression (P)/Nonprogression (NP)    6
US Cobb Change                                 9
Apical Axial Vertebral rotation (AVR)          8
RC value                                       8
Curve magnitude                                8
dtype: int64
```

Dropping down all the missing values

In [9]:
```
df1 = df.dropna() # removing missing values
```

In [10]:
```
df1.head(100)
```

Out[10]:

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
0	NP	-8.0	1.7	0.099778	20.0
1	NP	-10.0	-3.1	0.086335	35.0
2	NP	8.0	-4.2	0.077022	24.0
3	NP	1.0	4.4	0.091409	14.0
4	NP	7.0	7.9	0.059883	38.0
...	...	...	...	...	...
96	NP	5.0	3.3	0.062135	30.0
97	NP	-9.0	11.2	0.063483	36.0
98	P	6.0	-4.9	0.031209	30.0
99	NP	-6.0	-11.6	0.058261	20.0
100	NP	-1.0	15.7	0.065022	31.0

100 rows × 5 columns

In [11]:
```
df1.describe()
```

Out[11]:

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
count	154.000000	154.000000	154.000000	154.000000
mean	-0.006494	4.442208	0.071266	25.006494
std	6.147954	8.005993	0.024032	9.376299
min	-16.000000	-14.000000	0.013752	9.000000
25%	-3.750000	-1.275000	0.055077	18.000000
50%	0.500000	3.900000	0.067180	24.000000
75%	4.000000	10.125000	0.088607	32.000000
max	17.000000	26.500000	0.130443	49.000000

Replacing the dependent variables Output (Progression (P) as "1" /Nonprogression (NP) as "0"

In [12]:
```
df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_rep
                  value = 0)

df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_rep
                  value = 1)
```

```
<ipython-input-12-55b38457e085>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_r
eplace ="NP",
<ipython-input-12-55b38457e085>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
```

```
  df1['Output (Progression (P)/Nonprogression (NP)']= df1['Output (Progression (P)/Nonprogression (NP)'].replace(to_r
eplace ="P",
```

In [13]:
```
df1.head(10)
```

Out[13]:

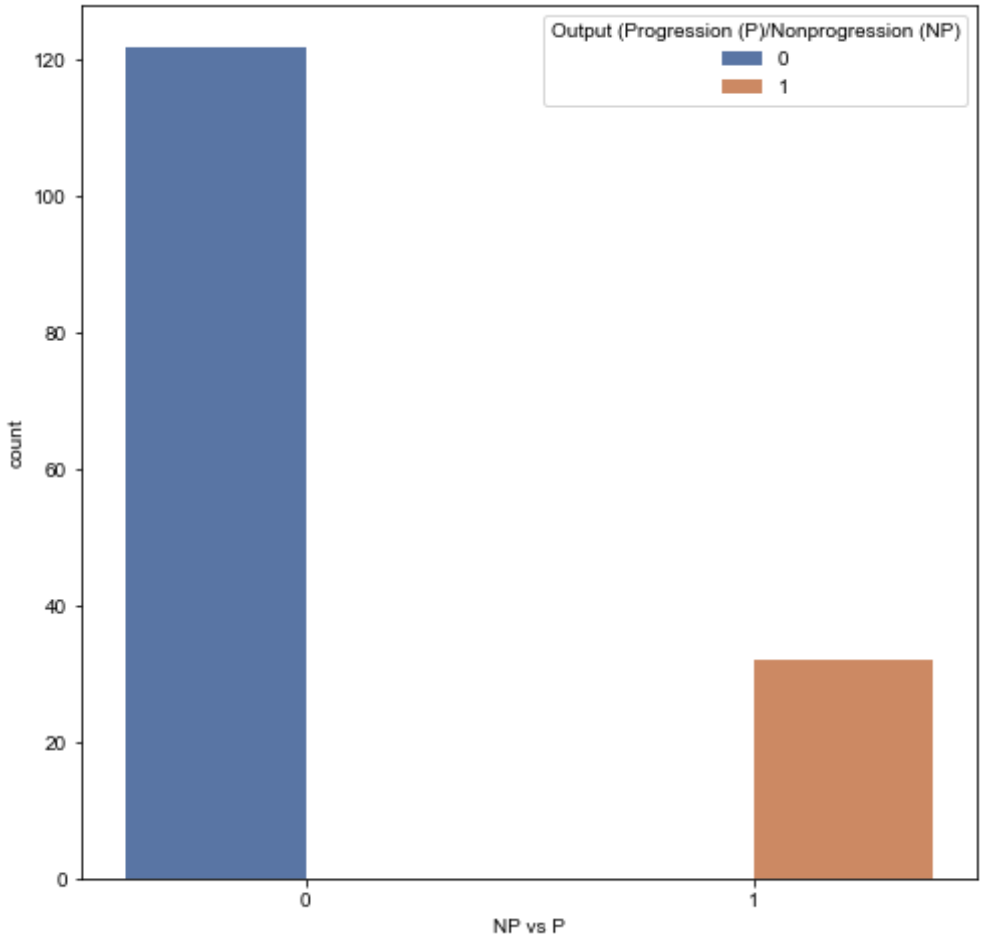
	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
0	0	-8.0	1.7	0.099778	20.0
1	0	-10.0	-3.1	0.086335	35.0
2	0	8.0	-4.2	0.077022	24.0
3	0	1.0	4.4	0.091409	14.0
4	0	7.0	7.9	0.059883	38.0
5	1	11.0	4.2	0.105570	29.0
6	0	-1.0	11.5	0.069357	32.0
7	0	6.0	11.2	0.077335	42.0
8	0	-12.0	1.1	0.083830	22.0
9	0	-7.0	-4.7	0.065243	38.0

Count of dependent variable (Progression (P)/Nonprogression (NP))

In [14]:
```
df1["Output (Progression (P)/Nonprogression (NP)"].value_counts()
```

Out[14]:
```
0    122
1     32
Name: Output (Progression (P)/Nonprogression (NP), dtype: int64
```

In [15]:
```
plt.figure(figsize = (8,8))
sns.countplot(x=df1["Output (Progression (P)/Nonprogression (NP)"], hue=df1["Output (Progression (P)/Nonprogression (
plt.xlabel("NP vs P")
sns.set_style('darkgrid')
plt.show()
```



Splitting the dependent and independent variables.

In [16]:
```
X= df1.iloc[:,1:6]
y = df1.iloc[:,0:1]
X.head(10)
```

Out[16]:

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
0	-8.0	1.7	0.099778	20.0

	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
1	-10.0	-3.1	0.086335	35.0
2	8.0	-4.2	0.077022	24.0
3	1.0	4.4	0.091409	14.0
4	7.0	7.9	0.059883	38.0
5	11.0	4.2	0.105570	29.0
6	-1.0	11.5	0.069357	32.0
7	6.0	11.2	0.077335	42.0
8	-12.0	1.1	0.083830	22.0
9	-7.0	-4.7	0.065243	38.0

In [17]:
```
from sklearn.model_selection import train_test_split
```

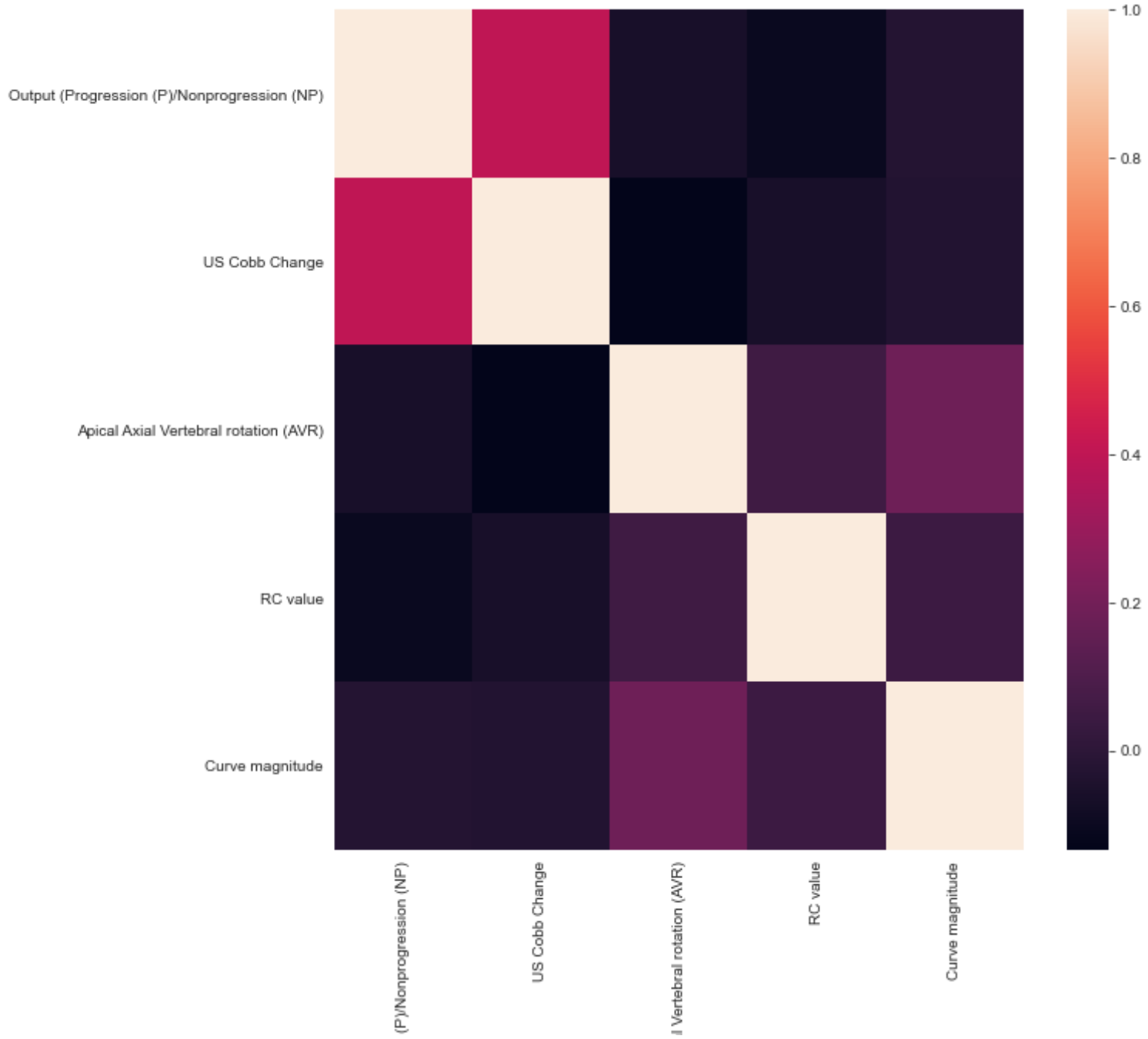
In [18]:
```
df1.corr()
```

Out[18]:

	Output (Progression (P)/Nonprogression (NP)	US Cobb Change	Apical Axial Vertebral rotation (AVR)	RC value	Curve magnitude
Output (Progression (P)/Nonprogression (NP)	1.000000	0.397526	-0.063077	-0.107131	-0.024331
US Cobb Change	0.397526	1.000000	-0.134470	-0.059670	-0.028798
Apical Axial Vertebral rotation (AVR)	-0.063077	-0.134470	1.000000	0.053471	0.187917
RC value	-0.107131	-0.059670	0.053471	1.000000	0.047044
Curve magnitude	-0.024331	-0.028798	0.187917	0.047044	1.000000

Heatmap for correlation matrix

In [19]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df1.corr())
plt.show()
```



Splitting the data into training and testing data

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20,random_state=27)
```

Standardizing the values

In [21]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logistic Regression

In [22]:
```
c = [0.5, 1, 1.5, 2,3]
for i in range(len(c)):

    logreg = LogisticRegression(C = c[i])

    logreg.fit(X_train, y_train)
    print('estimators=' +  (str(c[i])) + ", " +  'Accuracy of Logistic Regression training set {:.2f}'
    .format(logreg.score(X_train, y_train)))
    print('estimator=' +  (str(c[i])) + ", " + 'Accuracy of Logistic Regression on test set {:.2f}'
    .format(logreg.score(X_test, y_test)))
```

```
estimators=0.5, Accuracy of Logistic Regression training set 0.80
estimator=0.5, Accuracy of Logistic Regression on test set 0.77
estimators=1, Accuracy of Logistic Regression training set 0.80
estimator=1, Accuracy of Logistic Regression on test set 0.77
estimators=1.5, Accuracy of Logistic Regression training set 0.80
estimator=1.5, Accuracy of Logistic Regression on test set 0.81
```

```
estimators=2, Accuracy of Logistic Regression training set 0.80
estimator=2, Accuracy of Logistic Regression on test set 0.84
estimators=3, Accuracy of Logistic Regression training set 0.80
estimator=3, Accuracy of Logistic Regression on test set 0.84
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
```

Optimal value for logistic regression

In [23]:

```
#Logistic Regression
logreg1 = LogisticRegression(C = 1.5)
logreg1.fit(X_train,y_train)
logreg_predict = logreg1.predict(X_test)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg1.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg1.score(X_test, y_test)))
```

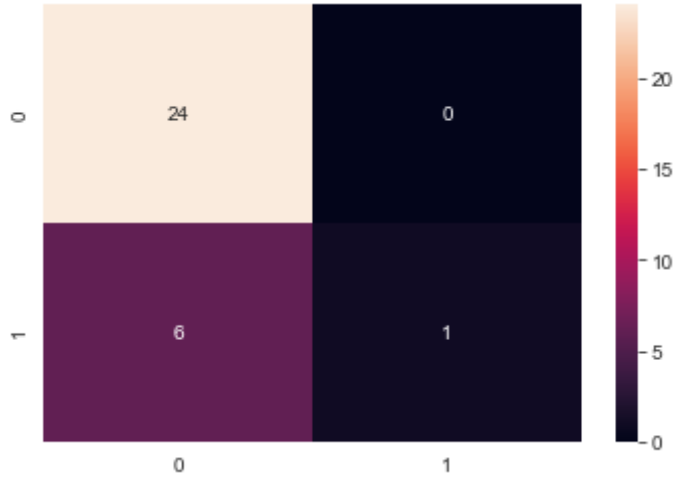
```
Accuracy of Logistic regression classifier on training set: 0.80
Accuracy of Logistic regression classifier on test set: 0.81
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
```

Confusion Matrix for Logistic Regression

In [24]:
```
predicted_labels_1 = logreg1.predict(X_test)# Logistic Rgeression
cf_matrix = cm(y_test, predicted_labels_1)
sns.heatmap(cf_matrix, annot=True)
```

Out[24]: <AxesSubplot:>



Printing out the Classiifcation Report for Logistic Regression

In [25]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_1)*100))
s1 = (accuracy_score(y_test, predicted_labels_1)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_1)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_1)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_1)))
```

```
Accuracy score : 80.65
Precision score : 1.00
Recall score : 0.14
f1_score score : 0.25
```

Descision Tree Classifier

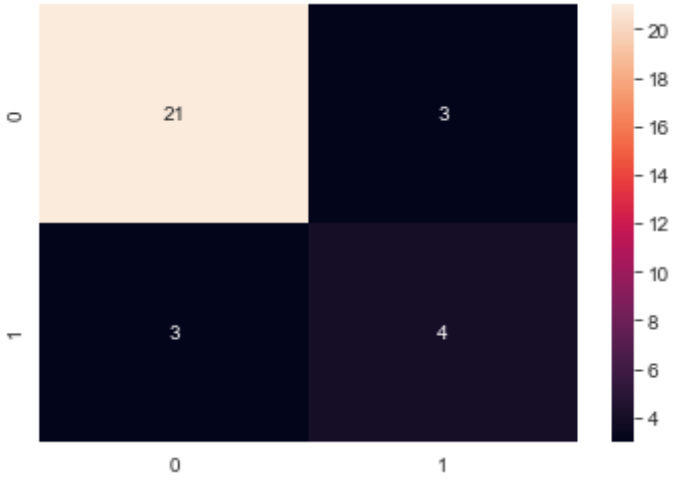
In [26]:
```
#Descision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.81
```

Confusion matrix for Decision Tree Classiifcation

In [27]:
```
predicted_labels_2 = clf.predict(X_test)# Descision Tree Classification
cf_matrix = cm(y_test, predicted_labels_2)
sns.heatmap(cf_matrix, annot=True)
```

Out[27]: <AxesSubplot:>



Printing out the classification report for Decision Tree Classiifcation

In [28]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_2)*100))
```

```
s2 = (accuracy_score(y_test, predicted_labels_2)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_2)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_2)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_2)))
```

```
Accuracy score : 80.65
Precision score : 0.57
Recall score : 0.57
f1_score score : 0.57
```

K neighbhors

In [29]:

```
# K neighbhors
n_neighbhor = [5, 10, 15, 20, 30]
from sklearn.neighbors import KNeighborsClassifier
for i in range(len(n_neighbhor)):

    knn = KNeighborsClassifier(n_neighbors=n_neighbhor[i])
    knn.fit(X_train, y_train)
    print( 'estimators=' +  str(n_neighbhor[i]) + ", " +  'Accuracy of KNN classifier on training set {:.2f}'
     .format(knn.score(X_train, y_train)))
    print( 'estimator=' +  str(n_neighbhor[i]) + ", " + 'Accuracy of KNN classifier on test set {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
estimators=5, Accuracy of KNN classifier on training set 0.85
estimator=5, Accuracy of KNN classifier on test set 0.84
estimators=10, Accuracy of KNN classifier on training set 0.80
estimator=10, Accuracy of KNN classifier on test set 0.81
estimators=15, Accuracy of KNN classifier on training set 0.80
estimator=15, Accuracy of KNN classifier on test set 0.77
estimators=20, Accuracy of KNN classifier on training set 0.80
estimator=20, Accuracy of KNN classifier on test set 0.77
estimators=30, Accuracy of KNN classifier on training set 0.80
estimator=30, Accuracy of KNN classifier on test set 0.77
<ipython-input-29-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-29-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-29-074b59d63d2d>:7: 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().
```

```
  knn.fit(X_train, y_train)
<ipython-input-29-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
<ipython-input-29-074b59d63d2d>:7: 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().
  knn.fit(X_train, y_train)
```

Optimal neighbhor value for KNN Classifier

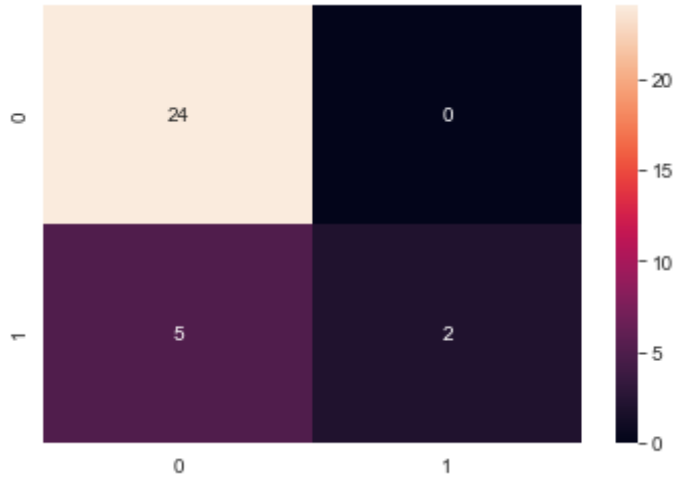
In [30]:
```
knn1 = KNeighborsClassifier(n_neighbors = 5)
knn1.fit(X_train,y_train)
knn1_predict = knn1.predict(X_test)
print(  'Accuracy of KNN classifier on training set {:.2f}'
     .format(knn1.score(X_train, y_train)))
print( 'Accuracy of KNN classifier on test set {:.2f}'
     .format(knn1.score(X_test, y_test)))
```

```
Accuracy of KNN classifier on training set 0.85
Accuracy of KNN classifier on test set 0.84
<ipython-input-30-f13be0dd2392>:2: 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().
  knn1.fit(X_train,y_train)
```

Confusion matrix for KNN

In [31]:
```
predicted_labels_3 = knn1.predict(X_test)# KNN
cf_matrix = cm(y_test, predicted_labels_3)
sns.heatmap(cf_matrix, annot=True)
```

Out[31]: <AxesSubplot:>



Printing out the classification report for KNN

In [32]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_3)*100))
s3 = (accuracy_score(y_test, predicted_labels_3)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_3)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_3)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_3)))
```

```
Accuracy score : 83.87
Precision score : 1.00
Recall score : 0.29
f1_score score : 0.44
```

Random Forest Classifier

In [33]:
```
# random forest classifier
n_estimator = [10, 20, 30, 40]
from sklearn.ensemble import RandomForestClassifier
for i in range(len(n_estimator)):
    rfc = RandomForestClassifier(n_estimators=n_estimator[i], random_state= 1)
    rfc.fit(X_train,y_train)
    rfc_predict = rfc.predict(X_test)
    print( 'estimators=' +  str(n_estimator[i]) + ", " +  'Accuracy of Random Forest classifier on training set {:.2f
```

```
     .format(rfc.score(X_train, y_train)))
    print( 'estimator=' +  str(n_estimator[i]) + ", " + 'Accuracy of Random Forest classifier on test set {:.2f}'
     .format(rfc.score(X_test, y_test)))
```

```
<ipython-input-33-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
<ipython-input-33-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
<ipython-input-33-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
<ipython-input-33-fc1189026350>:6: 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().
  rfc.fit(X_train,y_train)
estimators=10, Accuracy of Random Forest classifier on training set 0.98
estimator=10, Accuracy of Random Forest classifier on test set 0.81
estimators=20, Accuracy of Random Forest classifier on training set 1.00
estimator=20, Accuracy of Random Forest classifier on test set 0.81
estimators=30, Accuracy of Random Forest classifier on training set 1.00
estimator=30, Accuracy of Random Forest classifier on test set 0.81
estimators=40, Accuracy of Random Forest classifier on training set 1.00
estimator=40, Accuracy of Random Forest classifier on test set 0.81
```

Optimal estimator value for Random Forest Classifier

In [34]:

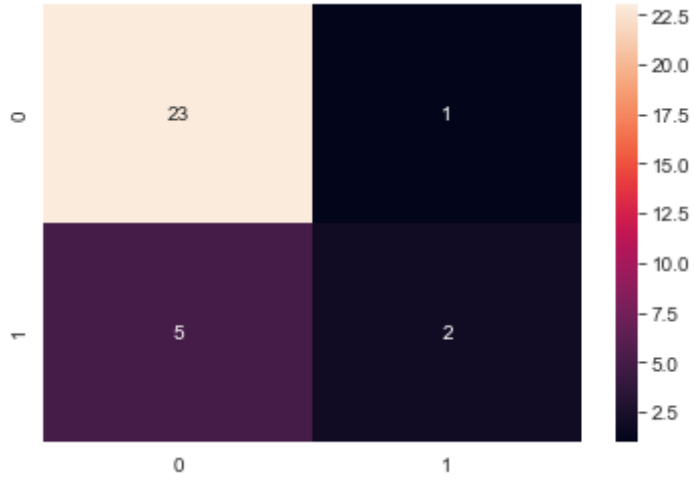
```
rfc1 = RandomForestClassifier(n_estimators=10, random_state= 1)
rfc1.fit(X_train,y_train)
rfc_predict = rfc1.predict(X_test)
print(  'Accuracy of Random Forest classifier on training set {:.2f}'
     .format(rfc1.score(X_train, y_train)))
print( 'Accuracy of Random Forest classifier on test set {:.2f}'
     .format(rfc1.score(X_test, y_test)))
```

```
Accuracy of Random Forest classifier on training set 0.98
Accuracy of Random Forest classifier on test set 0.81
<ipython-input-34-d4001a358eef>:2: 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().
  rfc1.fit(X_train,y_train)
```

Confusion matrix for Random Forest Classifer

In [35]:
```
predicted_labels_4 = rfc1.predict(X_test)# RFC
cf_matrix = cm(y_test, predicted_labels_4)
sns.heatmap(cf_matrix, annot=True)
```

Out[35]: <AxesSubplot:>



Printing out the classification report for Random forest Classifier

In [36]:
```
print("Accuracy score : {:.2f}" .format(accuracy_score(y_test, predicted_labels_4)*100))
s4 = (accuracy_score(y_test, predicted_labels_4)*100)
print("Precision score : {:.2f}" .format(precision_score(y_test, predicted_labels_4)))
print("Recall score : {:.2f}" .format(recall_score(y_test, predicted_labels_4)))
print("f1_score score : {:.2f}" .format(f1_score(y_test, predicted_labels_4)))
```

```
Accuracy score : 80.65
Precision score : 0.67
Recall score : 0.29
f1_score score : 0.40
```

K fold validation Logistic regression

In [37]:
```
# K fold validation Logistic regression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
result = cross_val_score(logreg1, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks1 = np.mean(result)
```

```
[1.         0.72727273 0.81818182 0.81818182 0.9        0.8
 0.7        0.8        0.9        0.7        0.6        0.8
 0.7        0.9        0.8       ]
0.7975757575757576
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
```

```
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  return f(**kwargs)
```

K fold validation Decision Tree Classification

In [38]:

```
#  K fold validation Decision Tree Classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
result = cross_val_score(clf, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks2 = np.mean(result)
```

```
[0.72727273 0.63636364 0.72727273 0.63636364 0.8        0.7
 0.5        0.7        0.8        0.7        0.7        0.8
```

 0.7   0.9   0.5   ]
0.7018181818181818

K cross fold validation KNN

In [39]:

```
#K croos fold validation KNN
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
result = cross_val_score(knn1, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks3 = np.mean(result)
```

```
[0.81818182 0.90909091 0.72727273 0.72727273 0.6        0.9
 0.7        0.6        0.8        0.9        0.8        0.6
 1.         0.6        0.8       ]
0.7654545454545455
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

Random Forest Classifier K fold cross validation

In [40]:

```
#Random Forest Classifier K fold cross validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold  = KFold(n_splits=15, shuffle=True) # shuffle=True
```

```
result = cross_val_score(rfc1, X, y, cv = kfold)
print(result)
print(np.mean(result))
ks4 = np.mean(result)
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
```

```
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
[1.         0.72727273 0.81818182 0.72727273 0.8        0.9
 0.8        0.8        0.6        0.6        1.         0.7
 0.8        0.7        0.8       ]
0.7848484848484848
```

```
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Rithvik Ramesh\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:531: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

Accuracies of four different models on scoliosis dataset

In [42]:

```
scores = []
Algorithms = ['Logistic', 'Decision tree', 'KNN', 'Random Forest']
```

In [43]:

```
scores.extend([s1,s2,s3,s4])
```

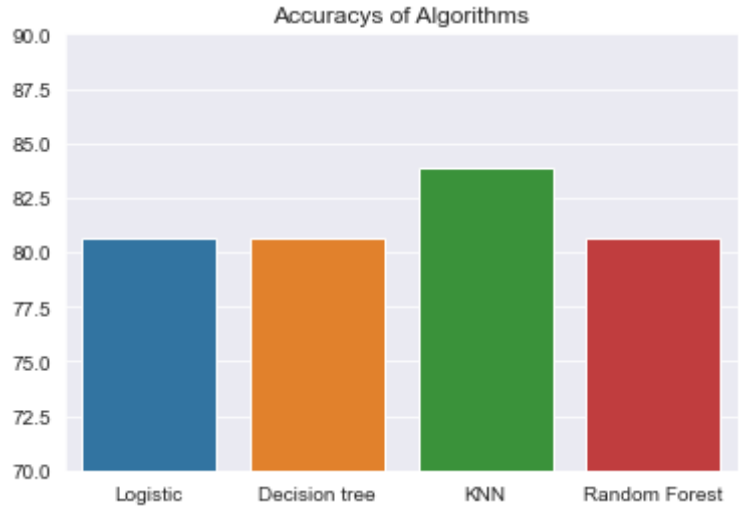
In [44]:

```
print(scores)
```

```
[80.64516129032258, 80.64516129032258, 83.87096774193549, 80.64516129032258]
```

In [45]:

```
plt.title('Accuracys of Algorithms')
sns.barplot(x = Algorithms, y = scores)
plt.ylim(70,90)
plt.show()
```



Accuracies of four different models on scoliosis dataset after k-fold validation

In [46]:
```
scores1 = []
Algorithms = ['Logistic', 'Decision tree', 'KNN', 'Random Forest']
```

In [47]:
```
scores1.extend([ks1*100,ks2*100,ks3*100,ks4*100])
```

In [48]:
```
print(scores1)
```

```
[79.75757575757576, 70.18181818181817, 76.54545454545455, 78.48484848484848]
```

In [49]:
```
plt.title('Accuracys of Algorithms with k fold validation')
sns.barplot(x = Algorithms, y = scores1)
plt.ylim(40,90)
plt.show()
```