AI-Statistical Machine Learning Approaches to Liver Disease Prediction

Team ID: PNT2022TMID48272

Faculty Mentor:

D.**Pradhiba**

Team Leader: G.lydia
Team Member: R.priya
Team Member: U.lavanya sri
Team Member: G.nagalakshmi

Problem Solution Fit

Abstract:

The improvement of patient care, research, and policy is significantly impacted by medical diagnoses. Medical practitioners employ a variety of pathological techniques to make diagnoses based on medical records and the conditions of the patients. Disease identification has been significantly enhanced by the application of artificial intelligence and machine learning in conjunction with clinical data. Data-driven, machine learning (ML) techniques can be used to test current approaches and support researchers in potentially innovative judgments. The goal of this work was to use ML algorithms to derive meaningful predictors of liver disease from the medical data of 615 persons.

INTRODUCTION:

Using certain characteristics such as total bilirubin, direct bilirubin, alkaline phosphatase, totalprotein, albumin, andglobulin, this software can determine whether a patient has liver disease or not. In the human body, liver is considered as the main organ, which plays a central role in several bodily functions. In the human body the production of glucose, processing waste products, producing protein, removing worn-out tissue or cell, blood clotting to cholesterol, and iron metabolism are the core functions of the liver. Somehow, Liver disease cased due to the failure of any of these functions. According to the World Gastroenterology Organization (WGO)and World Health Organization (WHO), 35 million death cases causes occur due to liver failure. Liver infection frequently cause due to hepatotropic viruses executes a broad channel on health care resources. Liver diseases are basically classified into two classes that are acute and chronic. The acute liver disorder is an uncommon failure where fast debilitating of liver capacity results in coagulopathy, habitually with an

International Normalized Ratio (INR) of greater than 1.5, and variation in the intellectual status (encephalopathy) of an earlier healthy person. For the most part, the youngsters are influenced because of acute liver disorder which conveys a high proportion of death cases. The chronic liver disorder is a disease process of the liver which includes a procedure of dynamic devastation and recovery of the liver parenchyma prompting fibrosis and cirrhosis. We can endure just a couple of days on the off-chance that the liver closes down. At the point when the liver ends up unhealthy, it can do genuine harm our health. There can be several effect and wellbeing conditions that can unconsciously cause liver harm.

*Alcoho*l: Substantial Alcohol Consumption is the most widely recognized reason for liver disease. While drinking alcohol, the liver continues its role from the normal to focusing mostly on renovating alcohol into fewer toxic forms.*Obesity*: People with the substantial fat on their muscles mostly accrue around the liver, cause fatty liver disease. *Diabetes*: Diabetes patients have the risk of 50 percent liver disease, due to the high level of insulin that results in fatty liver disease. Researchers countenance moving tasks in the Healthcare associations to foresee any sort of disease for the early forecast and treatment from the enormous number of medical data. Nowadays data mining and machine learning become basic in healthcare due to its strategies e.g. for example classification, clustering, association rule mining for discovering frequent patterns pragmatic for disease prediction on medical data. The purpose of this study is to proposed a new solution for liver disease diagnoses based on Composite Hypercube on Iterated Random Projection (CHIRP). This study also includes the comparison of previously used models that are based on MLP, KNN, SVM, J48, RF, DS, RT and LR. The performance of each technique on the dataset is taken from UCI Machine Learning Repository is evaluated using MAE, RAE and Accuracy metrics. The rest of the paper is organized as follow: Section 2 and 3 describe the datasets and evaluation metrics employed in this exploration. Section 4 and 5 contain the overview of the existing models along with the proposed solution, while Section 6 includes the results obtained from experiments and discussion on the results.

Dataset Description:

S No	Attribute	Value Type	Description
1	mcv	Integer	Mean Corpuscular Volume
2	alkphos	Integer	Alkaline Phosphatase
3	sqpt	Integer	Alamine Aminotransferase
4	sqot	Integer	Aspartate Aminotransferase
5	gammagt	Integer	Gamma-Glutamyl Transpeptidase
6	drinks	Real	Number of Half-Pint Equivalents of Alcoholic Beverages % Drunk Per Day
7	selector	Selector {1, 2}	Field Used to Split Data into Two Sets

Evaluating your model is an important part of any research study. Your model may give you satisfactory results when you evaluate it with some standard evaluation metrics. In this study, the following measurements are used for the evaluation of the proposed model as compare to other mode.

Mean Absolute Error (MAE):

MAE is used for model valuation based on regression models. The MAE of a technique or model regarding an assessment set is the mean of the absolute estimation of the discrete desire error on inclusive events in the assessment set, that is the distinction between the predicted error and the true error for overall events. MAE is calculated as follow:

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} \mathbf{abs}\,(y_i - \lambda(x_i)}{n}$$

Relative Absolute Error (RAE):

RAE is essentially the same as the relative squared error as likewise, it is comparative with a simple predictor, which is only the average of the

literal values. For this situation, however, the error is only the total absolute error rather than the absolute squared error.
The RAE of a single instance i can be calculated using the following equation:

$$\mathbf{RAE} = \frac{\sum_{j=1}^{n} |P_{(ij)} - T_j|}{\sum_{j=1}^{n} |T_j - \bar{T}|}$$

Accuracy:

This evaluation metric is used for the measurement of classification models, that predict it got right

$$\mathbf{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

For binary classification, accuracy can likewise be determined as far as positives and negatives as pursues:

$$\mathbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

here, TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

Support Vector Machine:

Support Vector Machine (SVM), is created for gainfully plan straight learning machines in kernel induced component chairs by smearing the concept speculation of Vapnik and associates. It makes a small twofold depiction of the created hypothesis which prompts capable learning techniques, that can be taken care of by an enhancement system because of the Karush–Kuhn–Tucker conditions. Likewise, on account of Mercer's conditions on the pieces, the optimization issues are bent and the course of action combines to an overall perfect point. These highlights make SVM stand isolated among other models of recognition strategies, for instance, neural structures. The objective of the support vector learning machine is to discover f(x, α) with α comparable to the weights and prejudices to delineate an essential relationship in the input data and their outcomes.

The SVM system trains machines utilizing the instrument of reducing an upper bound on the disentanglement error while different procedures, for example, neural systems diminish training error on training data.

Decision Tree (J48):

This is a basic C4.5 decision tree utilized for classification problems which make a doubletree. This methodology is recorded significant in classification problems. Utilizing this strategy, a tree is worked to consummate the classification procedure, that is additionally pragmatic to an individual record in the dataset and item in classification for that record. During this procedure, J48 algorithm mocks the lost values e.g. the value for that element can be forecasted grounded on what is perceived close by the classification value for different records. The basic idea is to parcel the data into run reliant on the quality regards for that thing that is found in the working out test. It permits classification through in addition decision trees or rubrics created from scratch.

Random Forest:

Random Forests (RF)s are a collective learning method for regression and classification that work by building lots of decision trees at learning time and outputting the class, which is the technique for the class outcome by every tree. RF does, as an outfit strategy for numerous trees, better to deal with categorical data in the wake of getting the last arrangement in the larger part casting a voting system for the outcomes of each tree is mediated.

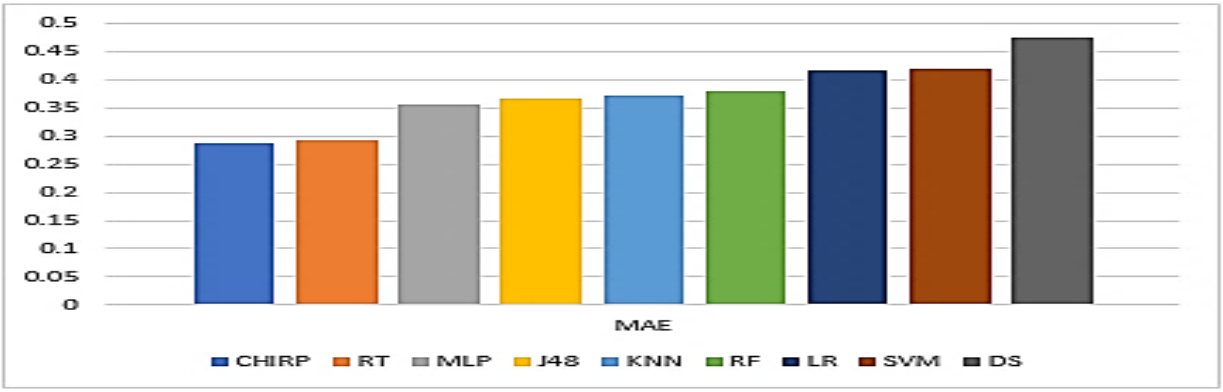
Logistic Regression:

Logistic Regression (LR), measures the connection between a categorical reliant on variable and at least one autonomous variables, which are generally continuous, by utilizing likelihood scores as the anticipated estimations of the reliant on a variable. Chances are the proportion

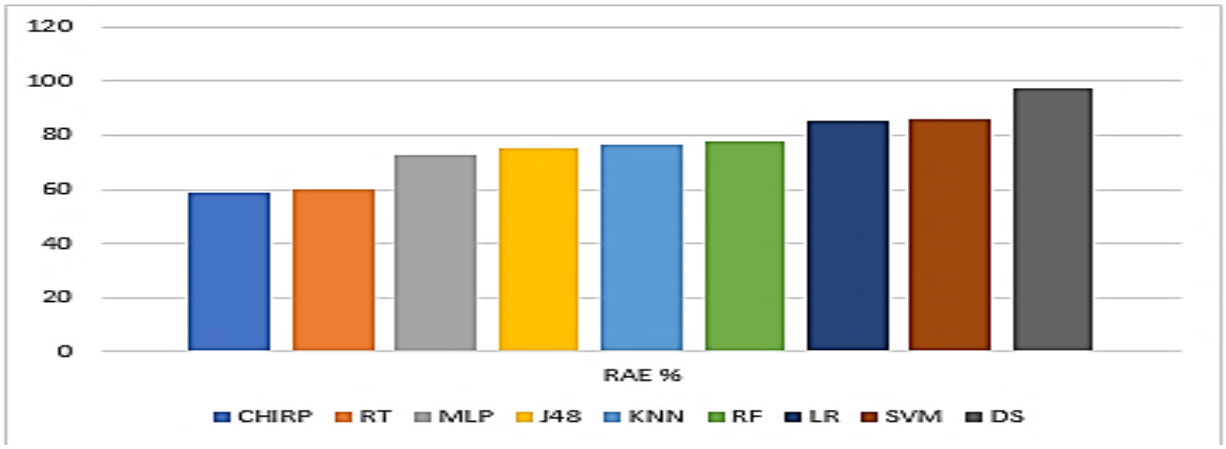
between the likelihood of accomplishment over the likelihood of disappointment, that is, pi / (1-pi), where p is the probability of a record belongs to Class 0. When p > 0.5, a record would be classified as Class 0. Else it would be critic as Class 1.

Random Tree:

Random Tree (RT) , is a collective learning technique that makes numerous individual learners. It is a technique for assembling a tree that treats K arbitrary highlights at every node. It includes a snaring thought to make an arbitrary arrangement of data for structuring a decision tree. To structure a standard tree, every node is part of utilizing the best part among all variables.



1	CHIRP	0.2870	58.8765
2	RT	0.2928	60.0659
3	MLP	0.3543	72.6840
4	J48	0.3673	75.3511
5	KNN	0.3718	76.2906
6	RF	0.3803	78.0322
7	LR	0.4151	85.1648
8	SVM	0.4174	85.6386
9	DS	0.4751	97.4695



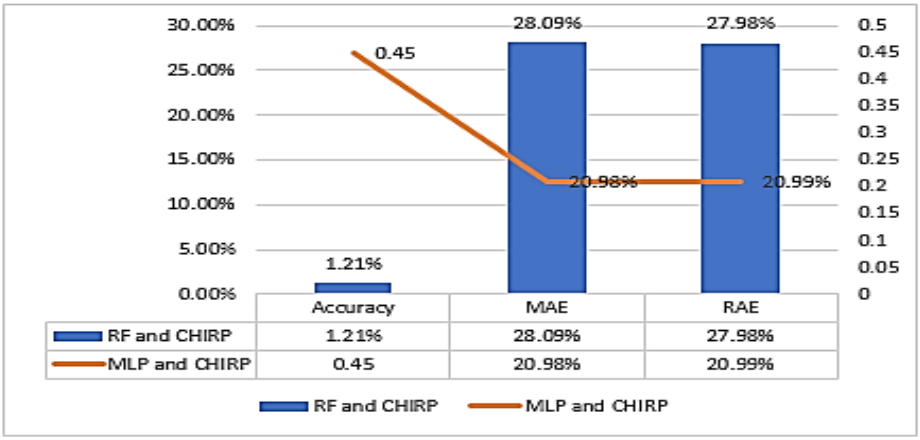
S No	Model	Accuracy
1	RF	72.17%
2	MLP	71.60%
3	CHIRP	71.30%
4	RT	70.72%
5	J48	68.70%
6	LR	68.11%
7	KNN	62.90%
8	SVM	58.30%
9	DS	57.70%

Composite Hypercube on Iterated Random Projection:

Composite Hypercube on Iterated Random Projection (CHIRP), is an iterative mechanism of three phases; anticipating, covering, and binning, which expected to a pact with the scourge of dimensionality, computational unconventionality, and nonlinear notice ability. This technique is not the hybridization of other models, also not the enhancement or alteration of prevailing models; it utilizes new casing techniques. The exactness of CHIRP on generally utilized target datasets surpasses the precision of contenders. The CHIRP algorithm was created to process information gathered by the long-pattern Event Horizon Telescope, the global cooperation that in 2019 caught the black-hole picture of M87* for the first time. This algorithm was not used to produce images but was a mathematical solution for the extraction of data from radio signals fabricating data by a variety of radio telescopes spread far and wide. This technique uses the computationally effective approaches to build 2D forecasts and sets of

quadrangular sections on those forecasts, that comprises opinions from an individual group of data. CHIRP classifies these groups of predictions and sections them into a conclusion incline for counting new data opinions.
Supervised classifiers facet around authentic difficulties. At first, here is the blight of dimensionality: adjacent neighbors in large-dimensional areas veer aggressively with estimation. The resulting issue is computational multifaceted nature: multinomial-time procedures are impracticable in large-dimensional spaces. A third issue is a uniqueness: for some theme sets in a direction space, for example, a 2D hover incorporated by a sphere, there is no equivalency that can let the division of areas by means of hyperplanes in the target space. There are various methods that can overcome these issues, such as k-mean clustering or random projection, reduction of dimensionality through principal component or hybridization of classifier.\

S No	Metrics	RF and CHIRP	MLP and CHIRP
1	Accuracy	1.21%	0.45
2	MAE	28.09%	20.98%
3	RAE	27.98%	20.99%



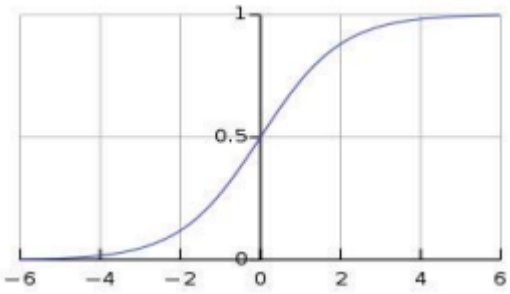
Conclusion:

Liver diseases are expanding on a regular schedule, and it's hard to foresee these ailments in the early premise. Researchers have utilized a huge

number of data mining models and machine learning strategies to foresee such sicknesses in the beginning period. Notwithstanding, in this research, CHIRP based model is presented for early analysis of liver disease. Based on experimental outcomes, it is seen that CHIRP performs well in lessening the error rate in assessment measurements rather than another utilized model. While looking at Accuracy RF and MLP perform well as opposed to CHIRP.

Prediction of Liver Disease using Classification Algorithms:

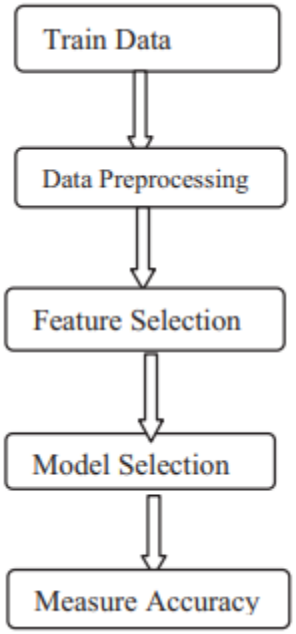
Machine Learning techniques nowadays have become very much important in the healthcare sector for the prediction of disease from the medical database. Many researchers and companies are leveraging machine learning to improve medical diagnostics. Among different machine learning techniques, classification algorithms are widely used in predicting diseases. In this paper, Logistic Regression, KNearest neighbour and Support Vector Machines are been used for prediction of liver disease. We all know that liver is the body largest internal organ which performs very important body function including making blood clotting factors and proteins, manufacturing triglyceride and cholesterol, glycogen synthesis and bile production. Usually, more than 75% of liver tissue needs to be affected by a decrease in function to occur.1 So it's important to detect at an early stage such that the disease can be treated before it becomes severe.
Logistic Regression Logistic Regression is a classification algorithm which is used to predict the binary outcome on a given set of independent variable. In logistic regression, we only look at the probability of outcome dependent variable.2 The odds of success is by odd = P/1-P. In logistic regression, the dependent variable is a logit that is natural log of odd given by log(odds) = logit(P) = ln(P/1-P). In logistic regression, we find logit(P) = a + bX = y. ln(P/1-P) = y. P/1-P = ey . P = ey /1+ey . Logistic regression equation: P = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)); where y= b0+b1*x A typical logistic function is given by…



K-nearest neighbour can be said as a classification, nonparametric algorithm which stores all available cases and its works is to classify new cases based on a similarity measure.

METHODOLOGY USED:

The proposed methods are used to compare classification accuracy of Logistic Regression, K-nearest neighbour and Support Vector Machine. The first step is to clean the data. Filling the missing values followed by transforming nominal attribute to binary attribute. The next step is feature selection to select the best attribute for a subset of features. In this work, the relationship between the attributes and the predictor variable been seen by using a pivot table. Based on the findings, attributes been selected. The third step is data transformation. In this technique, data is been standardized such that the data follows Gaussian Distribution with a mean of 0 and standard deviation of 1. In the fourth step, the classification model is trained to predict the results in unseen data. In this section, results are analysed which are given by three different classification algorithms consist of Logistic Regression(LR), K-nearest neighbour(KNN) and Support Vector Machine(SVM). In the experiment, the dataset is divided into training set and testing set. The ratio of the training set is 70% and 30% respectively. In this work, 10- fold cross-validation is used to train and test the machine learning model. The experiment is conducted in Python programming language and the library used are pandas and sci-kit learn.



Business Problem:

Problem Context:

Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors.

Content:

This data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.Any patient whose age exceeded 89 is listed as being of age "90".

Features:

- Age of the patient
- Gender of the patient

- Total Bilirubin
- Direct Bilirubin
- Alkaline Phosphotase
- Alamine Aminotransferase
- Aspartate Aminotransferase
- Total Protiens
- Albumin
- Albumin and Globulin Ratio
- Dataset: field used to split the data into two sets (patient with liver disease, or no disease)

As stated earlier, classification is when the feature to be predicted contains categories of values. Each of these categories is considered as a class into which the predicted value falls. Classification algorithms include:

- Naive Bayes
- Logistic regression
- K-nearest neighbors
- (Kernel) SVM
- Decision tree
- Ensemble learning

Naive Bayes:

Naive Bayes applies the Bayes' theorem to calculate the probability of a data point belonging to a particular class. Given the probability of certain related values, the formula to calculate the probability of an event *B*, given event *A* to occur is calculated as follows.

$P(B|A) = (P(A|B) * P(B) / P(A))$

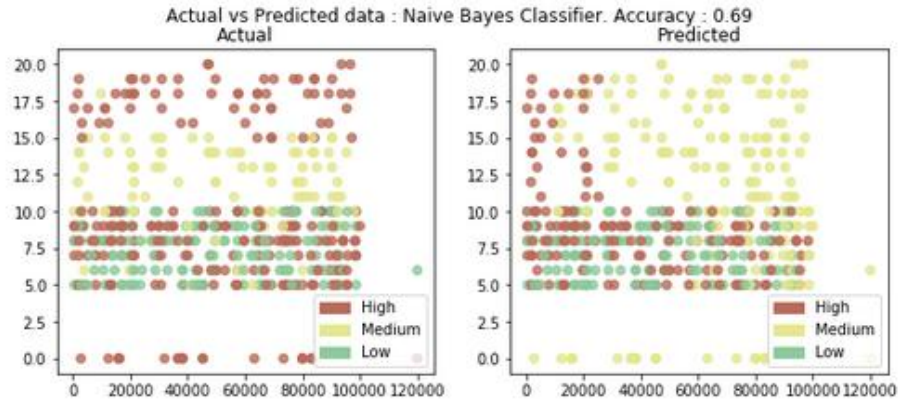
This theory is considered naive, because it assumes that there is no dependency between any of the input features. Even with this not true or naive assumption, the Naive Bayes algorithm has been proven to perform really well in certain use cases like spam filters.

The following code snippet shows an example of how to create and predict a Naive Bayes model using the libraries from scikit-learn.

```
from sklearn.naive_bayes import MultinomialNB

model_name = 'Naive Bayes Classifier'

nbClassifier = MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

nb_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures),('classifier', nbClassifier)])

nb_model.fit(X_train,y_train)

y_pred_nb= nb_model.predict(X_test)
```

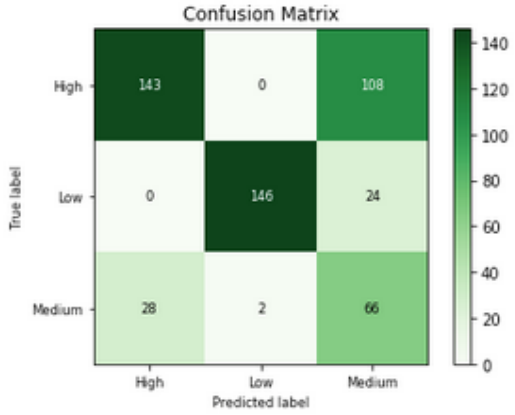
While analyzing the predicted output list, we see that the accuracy of the model is at 69%. A comparative chart between the actual and predicted values is also shown.

```
y_test = label_encoder.transform(y_test)
#y_pred_nb = label_encoder.transform(y_pred_nb)
two_d_compare(y_test,y_pred_nb,model_name)
```



```
y_test = label_encoder.inverse_transform(y_test)
y_pred_nb = label_encoder.inverse_transform(y_pred_nb)
model_metrics(y_test,y_pred_nb)
```

```
Decoded values of Churnrisk after applying inverse of label encoder : ['High' 'Low' 'Medium']
```



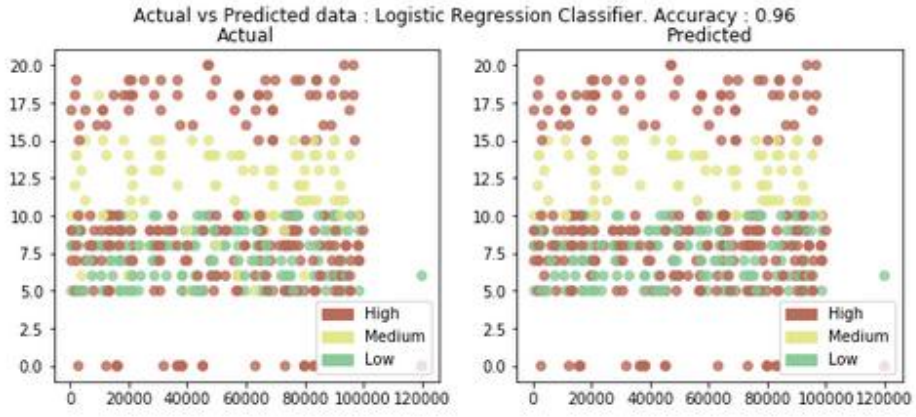
Logistic regression:

Logistic regression is an extension to the linear regression algorithm. The details of the linear regression algorithm are discussed in Learn regression algorithms using Python and scikit-learn. In a logistic regression algorithm, instead of predicting the actual continuous value, we predict the probability of an outcome. To achieve this, a *logistic function* is applied to the outcome of the linear regression. The logistic function is also referred to as a *sigmoid function*. This outputs a value between 0 and 1. Then, we select a line that depends on the use case. Any data point with a probability value above the line is classified into the class represented by 1. The data point below the line is classified into the class represented by 0.The following code snippet shows an example of how to create and predict a logistic regression model using the libraries from scikit-learn.

```
from sklearn.linear_model import LogisticRegression

model_name = "Logistic Regression Classifier"

logisticRegressionClassifier = LogisticRegression(random_state=0,multi_class='auto',solver='lbfgs',max_iter=1000)

lrc_model = Pipeline(steps=[('preprocessor', preprocessorForCategoricalColumns),
                            ('classifier', logisticRegressionClassifier)])

lrc_model.fit(X_train,y_train)

y_pred_lrc = lrc_model.predict(X_test)
```

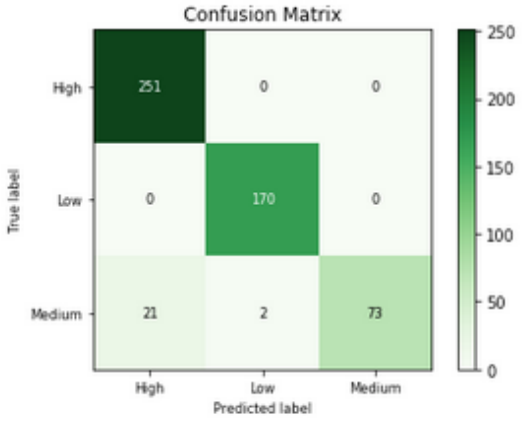
While analyzing the predicted output list, we see that the accuracy of the model is at 92%. A comparative chart between the actual and predicted values is also shown.

```
In [19]: y_test = label_encoder.transform(y_test)
         #y_pred_lrc = label_encoder.transform(y_pred_lrc)
         two_d_compare(y_test,y_pred_lrc,model_name)
```




```
y_test = label_encoder.inverse_transform(y_test)
y_pred_lrc = label_encoder.inverse_transform(y_pred_lrc)
model_metrics(y_test,y_pred_lrc)
```

```
Decoded values of Churnrisk after applying inverse of label encoder : ['High' 'Low' 'Medium']
```



K-nearest neighbors

The general idea behind K-nearest neighbors (KNN) is that data points are considered to belong to the class with which it shares the most number of common points in terms of its distance. K number of nearest points around the data point to be predicted are taken into consideration. These K points

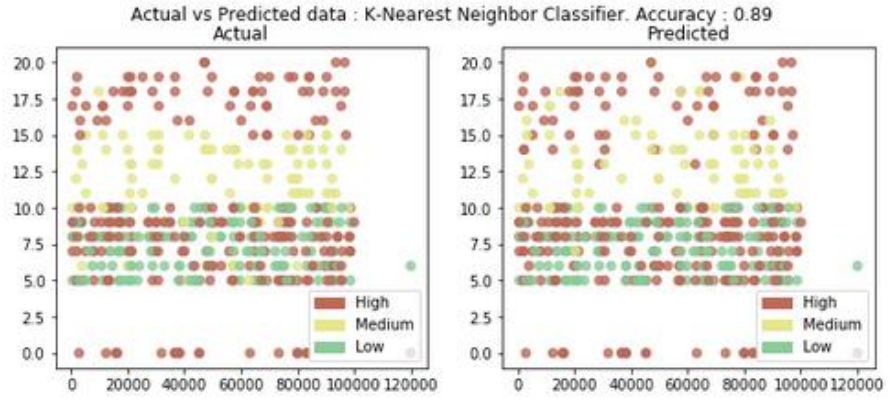
at this time already belong to a class. The data point under consideration is said to belong to the class with which the most number of points from these K points belong. There are several methods to calculate the distance between points. The most popular formula to calculate this is the Euclidean distance.

The following code snippet shows an example of how to create and predict a KNN model using the libraries from scikit-learn.

```
In [15]: from sklearn.neighbors import KNeighborsClassifier

         model_name = "K-Nearest Neighbor Classifier"

         knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric='minkowski', p=2)

         knn_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures),('classifier', knnClassifier)])

         knn_model.fit(X_train,y_train)

         y_pred_knn = knn_model.predict(X_test)
```

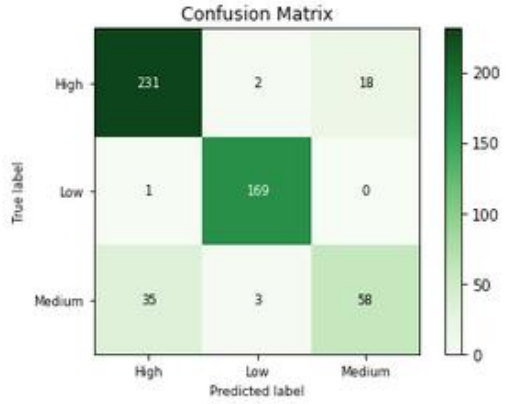
While analyzing the predicted output list, we see that the accuracy of the model is at 89%. A comparative chart between the actual and predicted values is also shown. The general idea behind K-nearest neighbors (KNN) is that data points are considered to belong to the class with which it shares the most number of common points in terms of its distance. K number of nearest points around the data point to be predicted are taken into consideration. These K points at this time already belong to a class. The data point under consideration is said to belong to the class with which the most number of points from these K points belong. There are several methods to calculate the distance between points. The most popular formula to calculate this is the Euclidean distance.

```
#y_test = label_encoder.transform(y_test)
#y_pred_knn = label_encoder.transform(y_pred_knn)
two_d_compare(y_test,y_pred_knn,model_name)
```




```
y_test = label_encoder.inverse_transform(y_test)
y_pred_knn = label_encoder.inverse_transform(y_pred_knn)
model_metrics(y_test,y_pred_knn)
```

```
Decoded values of Churnrisk after applying inverse of label encoder : ['High' 'Low' 'Medium']
```



Support Vector Machines:

Support Vector Machines (SVM) output an optimal line of separation between the classes, based on the training data entered as input. This line of separation is called a hyperplane in a multi-dimensional environment. SVM takes into consideration outliers that lie pretty close to another class to derive this separating hyperplane. After the model is constructed with this hyperplane, any new point to be predicted checks to see which side of the hyperplane this values lies in.

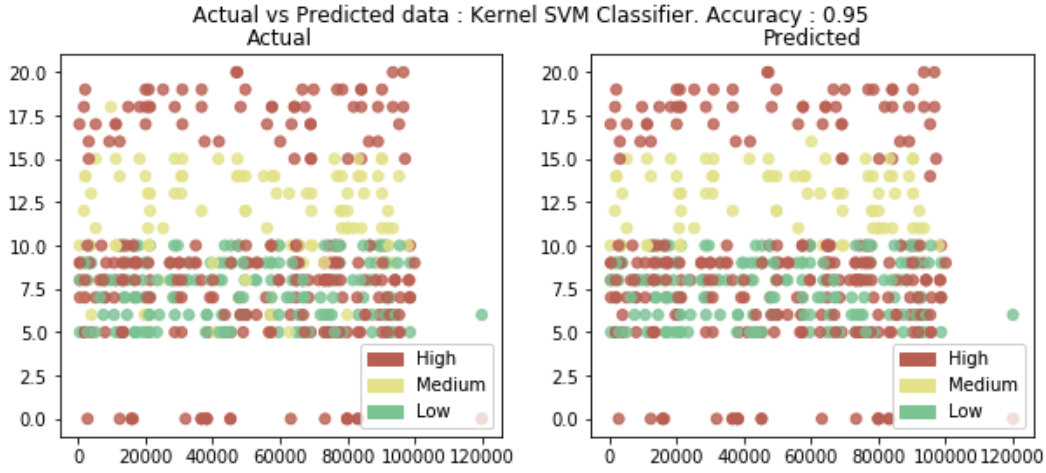
Even in 2-dimensional space, constructing this line of separation between classes can sometimes be tricky if the points are distributed without a clear distinction. Also, doing this when multiple features contribute to describe a data point is a complicated process. For these multi-dimensional spaces, where data is not linearly separable, we map it to a higher dimensional space to create this separation. This mapping to a higher dimension is achieved by applying a *kernel function*. There are several types of kernel functions, and the most common ones are the polynomial and the Guassian radial basis function (RBF). After this plane of separation is derived, the data is mapped back to its original dimension. Prediction at this point is merely finding if this point lies within or outside the plane. The following code snippet shows an example of how to create and predict an SVM model using the libraries from scikit-learn. The kernel value is set to 'rbf' to generate the hyperplane.

```
In [24]: from sklearn.svm import SVC

         model_name = 'Kernel SVM Classifier'

         svmClassifier = SVC(kernel='rbf', gamma= 'auto')

         svm_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures),('classifier', svmClassifier)])

         svm_model.fit(X_train,y_train)

         y_pred_svm = svm_model.predict(X_test)
```

While analyzing the predicted output list, we see that the accuracy of the model is at 95%. A comparative chart between the actual and predicted

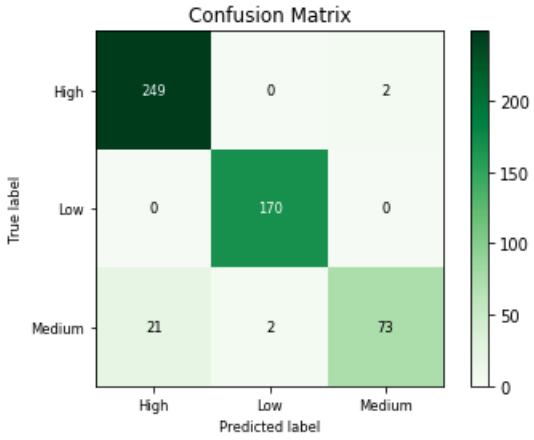
values is also shown.

```
In [25]: y_test = label_encoder.transform(y_test)
         #y_pred_svm = label_encoder.transform(y_pred_svm)
         two_d_compare(y_test,y_pred_svm,model_name)
```




```
In [26]: y_test = label_encoder.inverse_transform(y_test)
         y_pred_svm = label_encoder.inverse_transform(y_pred_svm)
         model_metrics(y_test,y_pred_svm)
```

```
Decoded values of Churnrisk after applying inverse of label encoder : ['High' 'Low' 'Medium']
```



Decision trees:

Decision tree-based models use training data to derive rules that are used to predict an output. For example, assume that the problem statement was to identify if a person can play tennis today. Depending on the values from

the training data, the model forms a decision tree. The model derived could have constructed a decision tree with the following rules.

1. First check the outlook column. If it's overcast, you definitely never go.
2. But if it's sunny and humid, then you don't go.
3. If it's sunny and normal, you go.
4. If it's rainy and windy, you don't go.
5. And if it's rainy and not windy, you go.

Ensemble learning:

Ensemble learning refers to the type of machine learning algorithms where more than one algorithm is combined to produce a better model. When two or more same algorithms are repeated to achieve this, it is called a *homogenous ensemble* algorithm. If different algorithms are assembled together, it is called a *heterogenous ensemble*. In this section, we'll look at how we can combine a decision tree-based model into a random forest and gradient boosted tree to get a higher accuracy level.

Random forest:

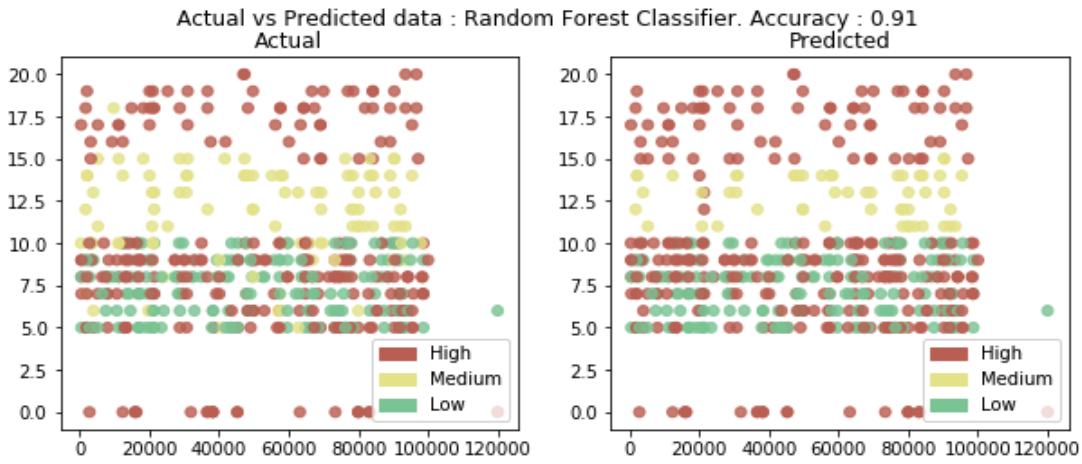
Decision tree algorithms are efficient in eliminating columns that don't add value in predicting the output. In some cases, we are even able to see how a prediction was derived by backtracking the tree. However, this algorithm doesn't perform individually when the trees are huge and hard to interpret. Such models are often referred to as weak models. The model performance is improvised by taking an average of several such decision trees derived from the subsets of the training data. This approach is called the *random forest* classification.

The following code snippet shows an example of how to create and predict a random forest model using the libraries from scikit-learn.

In [18]:
```
from sklearn.ensemble import RandomForestClassifier

model_name = "Random Forest Classifier"

randomForestClassifier = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0)

rfc_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures),('classifier', randomForestClassifier)])

rfc_model.fit(X_train,y_train)

y_pred_rfc = rfc_model.predict(X_test)
```

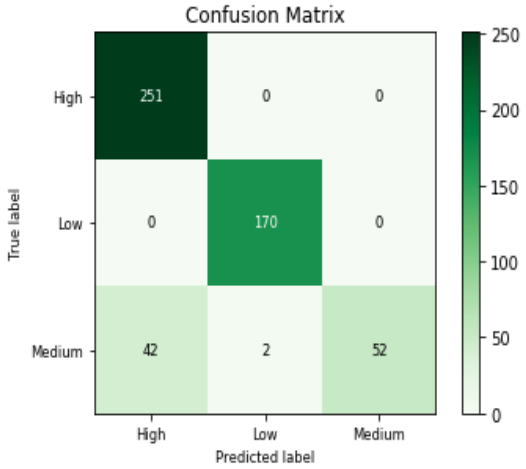
While analyzing the predicted output list, we see that the accuracy of the model is at 91%. A comparative chart between the actual and predicted values is also shown.

In [19]:
```
y_test = label_encoder.transform(y_test)
#y_pred_rfc = label_encoder.transform(y_pred_rfc)
two_d_compare(y_test,y_pred_rfc,model_name)

#three_d_compare(y_test,y_pred_rfc,model_name)
```



```
In [29]: y_test = label_encoder.inverse_transform(y_test)
         y_pred_rfc = label_encoder.inverse_transform(y_pred_rfc)
         model_metrics(y_test,y_pred_rfc)
```

```
Decoded values of Churnrisk after applying inverse of label encoder : ['High' 'Low' 'Medium']
```



Gradient boosted trees:

Gradient boosted trees are also a type of ensemble learning. They are based on the method called boosting, which involves training a model one after another based up on the outputs from the previous models.

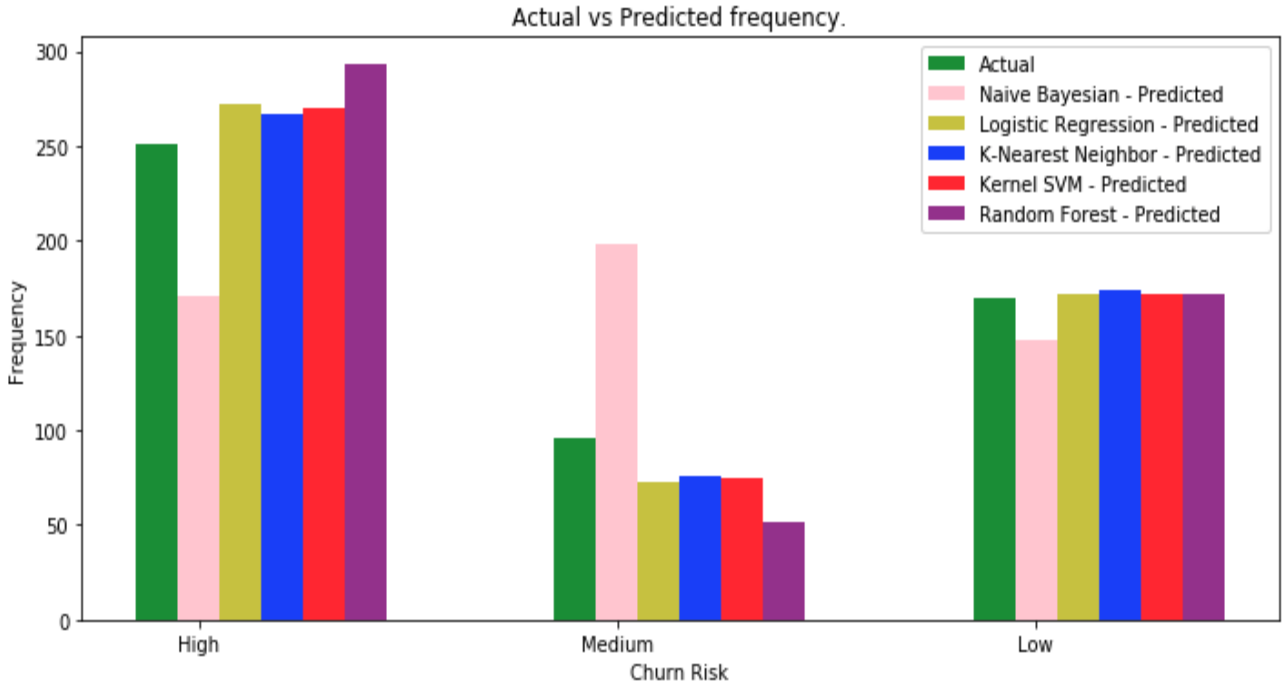
In gradient boosted trees, we calculate the error from the previous model, also known as *residuals*. Now we define another model that is trained on this residual. The resulting model is the sum of previous model and the model trained on residuals. This process is repeated until convergence. Even though gradient boosted trees out perform random forest models, they are computationally expensive because they are built sequentially. A specific implementation called XGBoost is used to overcome this issue. The details of XGBoost are out of scope of this tutorial.

Summary:

In this tutorial, we used the same data set to make predictions using several classification algorithms. The algorithims discussed in this tutorial are:

- Naive Bayes
- Logistic regression
- K-nearest neighbors
- SVM (Kernel)
- Decision tree
- Ensemble learning

We see that different algorithms behaves different accuracy scores. This does not mean that one algorithm is consistently better than the other ones. Even though certain classification algorithms consistently perform better than others, model performances are typically affected by the use case. Several hyperparameters can also be tuned in different ways within each of these algorithms to yield better accuracy. In the following bar chart, we compare the different classification algorithms against the actual values.



In general, we see that the predictions around the Medium values is low on accuracy. One reason for this could be that the number of entries for the Medium values were much less represented than the High and Low values. Further testing can be done by either increasing the number of Medium entries or involving several data fabrication techniques.

CONCLUSION:

In this research work, different classification algorithms namely Logistic Regression, Support Vector Machine and KNearest Neighbour have been used for liver disease prediction. The comparison of all these algorithms been done based on classification accuracy which is found through confusion matrix. From the experiment, Logistic Regression and K-Nearest Neighbour have the highest accuracy but logistic regression have the highest sensitivity. Therefore it can be concluded that Logistic Regression is appropriate for predicting liver disease.