Comparative analysis of Supervised Machine Learning Algorithms

Taslima Yeasmin Emu, Sayma Obaida, Nushrat Jahan, Sabrina Alam Khushbu
2016-1-60-038, 2016-1-60-033,2016-1-60-050, 2016-1-60-082

7, January 2021

1 Introduction

Introduction: Machine learning is one of the fastest growing areas of computer science that refers to the automated detection of meaningful patterns in data. Data mining is one of the application in Machine Learning. Classification is one of data mining task which classify an instance mutually exclusive into one of the target class.[6] The goal of classification is to predict the target class for each instance in the data set.[5]. This paper describes various supervised machine learning classification techniques and then compare these algorithms as well as determines the most efficient classification algorithm based on the data set, the number of instances and variables (features). Eight different machine learning algorithms were considered:Naïve Bayes (NB),Support Vector Machine (SVM),Neural Networks (Perceptron),Decision Tree,liniar regression, logistic regression,Random Forests ,k-nearest neighbors using Anaconda python.

1.1 Objectives

:

1.This work focuses on the classification of different machine learning algorithms and the objectives of our work -

2.To determine the most efficient algorithm with accuracy, precision ,recall, f1 score.

3.To establish the performance of different algorithms on our data set with visualization.

1.2 Motivation

In this generation of millions of available data motivated us to work with data.And for data analysis, gaining knowledge about machine learning is one of the important aspect. Which brought us to work in this project.

1.3 Existing works

:

According to [7], machine learning algorithms deals more with classification includes Logistic Regression, Naïve Bayes Classifier, Perceptron, Support Vector Machine; Quadratic Classifiers, K-Means Clustering, Boosting, Decision Tree, Random Forest (RF); Neural networks, Bayesian Networks and so on.

Logistic regression is a classification function that uses a single multinomial logistic regression model with single estimator. It is used for applied statistics and discrete data analysis. It states the boundary between the existing classes and the class probabilities which depend on distance from the boundary. The paper shows the performance of logistic regression with machine learning algorithms for clinical prediction model.[3]

A comparative analysis of Nonlinear Machine Learning Algorithms is presented in this paper for Detecting Breast Cancer. The objective of this paper is to evaluate the performance in classifying data with respect to efficiency and effectiveness of each algorithm in terms of classification test accuracy, precision, and recall. The performance comparison between machine learning algorithms is conducted on the Wisconsin Breast Cancer diagnostic dataset for the classification. The breast image database is loaded. Then the features are extracted and the classification model is trained and used for prediction of benign and malignant.[1]

A classification model for Chronic Kidney Disease based on a number of supervised learning algorithms are presented. The methodology that is defined as a process of revealing meaningful patterns from large Database . Different classification algorithms like -ZeroR, Rule Induction, Naïve Bayes, Regression, Support Vector Machine, Decision Tree, Decision Stump, k Nearest Neighbour are compared among each other. The classification algorithms as well as the evaluation metrics are used to evaluate the performance of the classifiers.[8]

The paper analyze the algorithms in terms of classification test accuracy to find out which algorithm perform best for the detection of Dementia. To find out the classification accuracy, the percentage of number of correctly classified samples divided by the total number of samples obtained is calculated. The algorithms are applied to both the datasets using WEKA tool. J48 algorithm is performing best among all the algorithms for the detection of Dementia.[4]

The paper detect network intrusion use of machine learning through data mining. The motivation of the paper is to monitor network for any harmful activity. And the data is large so data mining was used. Weka is used for 9 different classifier's performance measurement and the performance was compared.From this work we can get the assessment for which classifier can possibly give the best accuracy.This work gives the assessment of classifiers accuracy. [10]

The paper analyze different machine learning algorithms for real-time IP traffic classification.The motivation was to see how which machine learning algorithm gives the better result with real-time internet traffic classification.Dataset was created by capturing internet traffic and 5 different algorithms were used to find the accuracy.This work gives a comparative analysis how ML algorithm

performs with dynamic IP classification. [2]

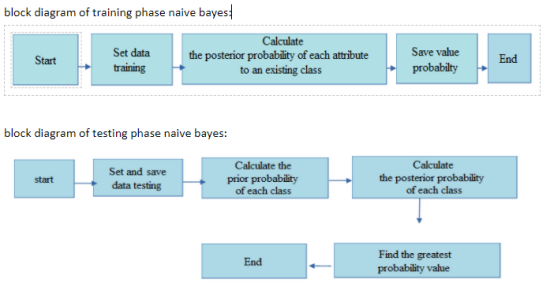
1.4 Necessity

This project is about machine learning algorithms, implement algorithms and analyze those algorithms. So, this research may be helpful for further analyzing related data.

2 Methodology

: In this project 8 supervised machine learning algorithm is used to compare the algorithms. The classification algorithm methodology is described bellow.
a) Naive Bayes Algorithm:

Supervised learning algorithms analyze the input data and learns a function to map the relationship between the input and output variables. Naive Bayes algorithm represents a supervised learning method as well as a statistical technique for classification based on Bayes Theorem. This probabilistic model permit to capture uncertainty about the model by determining probabilities.[9] In the case of Bayesian classification, it have a hypothesis that the given data belongs to a particular class. And then the probability is being calculated for the hypothesis to be true. Bayes theorem calculates the posterior probability P(c/x) from class prior probability P(c), predictor prior probability P(x) and liklihood P(x|c).[11]$Equation$:$P(c|x) = P(x|c)P(c)/P(x)(1)$This algorithm assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption simplifies computation, and this assumption is called class conditional independence.



Naive Bayes Algorithm Steps:

1.To determine the data that will be analyzed by Naive Bayes method, the first step is to read the train data.

2.Calculate number and probability. Find the mean and standard deviation values of each of the parameters that are numerical data.

3.Split the dataset by class and calculate statistics for each row

4.Calculate Gaussian Probability Density Function

5.Find the probabilities of predicting each class for a given row

6.Make a prediction with Naive Bayes on dataset and get Accuracy

These steps are involved in the Naive Bayes Classifier but Python comes with a library sklearn which makes all the mentioned steps easy to implement. That's why we use this library function on our project and the steps are :

1.Importing Libraries and Loading Datasets

2.Creating our Naive Bayes Model using sklearn

3.Making Predictions

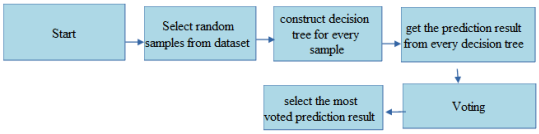
4.Getting Accuracy and Statistics

5.Confusion Matrix

6.Visualizing the result(ROC Curve)

b)Random Forest Algorithm:

Random forest is a supervised learning algorithm which creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.



Random Forest Algorithm Steps:

Step 1: Ensemble data by bootstrap method.

Step 2: Multiple decision trees by random classes.

Step 3: Select result by majority.

Bias are the main causes of difference in actual and predicted values. Ensemble can reduce it. Random Forest is a ensemble machine learning algorithm which is called bagging. Bagging is a bootstrap procedure which is applied to a high-variance machine learning algorithm.

The random forest is a model which is made up of many decision trees

Select random K data points from the training set.

Building the decision trees associated with the selected data points.

Choose the number N for decision trees that will be build.

For new data points, find the predictions of each decision tree and assign the new data points to the category that gains the majority votes.

c)Logistic regression: Logistic regression is one of the most popular machine learning algorithm which predicts the output of a categorical dependent variable and the outcome of this algorithm must be a categorical or discrete value. In this algorithm, we fit an "S" shaped logistic function and the S-shaped curve is called the Sigmoid function which predicts two maximum values (0 or 1). [12]

Sigmoid function: f(x)= 1/(1+e-x)

Initially, values of all coefficient is 0 and by using gradient descent we get new values for coefficients. Using these values we will create a model and pass this model to the sigmoid function to get predicted sigmoid values (between 0 and 1).

By comparing these values to threshold we perform the classification. We take derivative with respect to weights and bias to change the values of the coefficients. Then we will update the weights and bias by multiplying the learning rate to the derivatives we got with respect to weights and bias.

Predict function takes input data and give the predicted value. This function create a linear model using new weights and bias and pass this model to sigmoid function which will return the predicted values in the range of 0 to 1 and the accuracy function give the accuracy of the model.[13]

These steps are involved in the logistic regression but Python comes with a library sklearn which makes all the mentioned steps easy to implement. That's why we use this library function on our project and the steps are:

Logistic regression steps:

1.Loading Data

2.Selecting Feature

3.Splitting Data

4.Fitting logistic regression to the training set

5.Predicting the test result

6.Test accuracy of the result by creating confusion matrix

7.Visualizing the test set result(ROC Curve)

d)Neural Network:

To create a neural network, it begins with the perceptron. Perceptron is used for learning from training instances by running the algorithm through the training set until it finds a prediction which is correct on all of the training set. This prediction rule is used for predicting the labels on the test set.

The multilayer perceptron(MLP) is a feedforward neural network model which consists of multiple layers and each layer is connected to the following one. It is an interconnection of perceptrons in which data and calculations flow in a single direction from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons.

MLP is composed of three layers - input layer, output layer and the hidden layer. The nodes of the layers are using nonlinear activation functions, except the nodes of the input layer. And that input layer contains the inputs features of the network. The first hidden layer receives the weighted inputs and then send data from the previous layer to the next one.

Each neuron of the hidden layers receives the output from every neuron of the previous layers and transforms these values with a weighted linear summation, wixi=w0x0+w1x1+...+wn1xn1 , n is the number of neurons of the layer and wi corresponds to the ith component of the weight vector.

e) K-nearest neighbor :

The algorithm stores all the obtainable cases and classifies new cases based on a distance function. It is famous for its vastly use of estimating statistical data and pattern recognition. The procedure maintained by this algorithm is discussed below.

1. This algorithm will be used to find the K number of samples which are nearest to the data.

2. It initializes K to the chosen number of neighbors. For every chosen number it will calculate the distance between the query and the current sample from the dataset.

3. After initialization, distance and index will be added. 4. The collected distance will be sorted to the smaller one in ascending order.

5. K entries will be picked and labeled.

6. Since we are using classification, the mode of the K labels will be calculated.

7. In case of huge dataset some weights and parameters might need to be used.

The accuracy can be kept stable if we choose right K value. But we have to be careful in case of increased independent variables for which the algorithm might work slowly.

f) Linear Regression :

We used this supervised learning algorithm to perform a regression task. This model predicts value based on independent variables. We will try to interpret the output coefficients. This will help to find out the accuracy rate among the independent and dependent variable by calculating the mean of the variables. It might get difficult since mean is not a complete description of a single variable. We are planning to use Simple Linear Regression thereby we are considering relationship in two dimensional space between two variables to get a straight line. If we consider to assume graphically, then plotting independent variable on the x-axis and dependent variable on the y-axis will give the straight line to fit the best data points. The equation will be y = mx + b.

Now we will follow the given steps for our dataset. 1. Loading of the data

2. Since we will use one independent variable so simple regression will be followed. We might also check normality.

3. To prepare the data, we will take two variables to store the columns.

4. Splitting o f the data comes next. We will split them into training and testing.

5. After splitting, we will train the algorithm.

6. Accuracy prediction of our algorithm has to be followed by predicting the test data.

g)Decision Tree Induction :

Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Geometrically, the split describes a partition orthogonal to one of the coordinates of the decision space.

Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules. The working procedure is given below: The algorithm is called with three parameters: D, attribute list,

and Attribute selection method. We refer to D as a data partition. Initially, it is the complete set of training tuples and their associated class labels. The parameter attribute list is a list of attributes describing the tuples. Attribute selection method specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class. This procedure employs an attribute selection measure such as information gain or the Gini index. Whether the tree is strictly binary is generally driven by the attribute selection measure. Some attribute selection measures, such as the Gini index, enforce the resulting tree to be binary. Others, like information gain, do not, therein allowing multiway splits (i.e., two or more branches to be grown from a node).

The technical details of a decision tree are in how the questions about the data are formed. In the CART algorithm, a decision tree is built by determining the questions (called splits of nodes) that, when answered, lead to the greatest reduction in Gini Impurity. The Gini Impurity of a node is the probability that a randomly chosen sample in a node would be incorrectly labeled if it was labeled by the distribution of samples in the node. What this means is the decision tree tries to form nodes containing a high proportion of samples (data points) from a single class by finding values in the features that cleanly divide the data into classes. Each split is a single line that divides data points into nodes based on feature values. For this simple problem and with no limit on the maximum depth, the divisions place each point in a node with only points of the same class.

h)Support Vector Machine :

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. To understand SVM's a bit better, Lets first take a look at why they are called support vector machines. So say we got some sample data over here of features that classify whether a observed picture is a dog or a cat, so we can for example look at snout length or and ear geometry if we assume that dogs generally have longer snouts and cat have much more pointy ear shapes. So how do we decide where to draw our decision boundary? Well we can draw it over here or here or like this. Any of these would be fine, but what would be the best? If we do not have the optimal decision boundary we could incorrectly misclassify a dog with a cat. So if we draw an arbitrary separation line and we use intuition to draw it somewhere between this data point for the dog class and this data point of the cat class. These points are known as support Vectors – Which are defined as data points that the margin pushes up against or points that are closest to the opposing class. So the algorithm basically implies that only support vector are important whereas other training examples are 'ignorable'.

3 Implementation

3.1 Data Collection

The data are MC generated (see below) to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. Cherenkov gamma telescope observes high energy gamma rays, taking advantage of the radiation emitted by charged particles produced inside the electromagnetic showers initiated by the gammas, and developing in the atmosphere. This Cherenkov radiation (of visible to UV wavelengths) leaks through the atmosphere and gets recorded in the detector, allowing reconstruction of the shower parameters. The available information consists of pulses left by the incoming Cherenkov photons on the photomultiplier tubes, arranged in a plane, the camera. Depending on the energy of the primary gamma, a total of few hundreds to some 10000 Cherenkov photons get collected, in patterns (called the shower image), allowing to discriminate statistically those caused by primary gammas (signal) from the images of hadronic showers initiated by cosmic rays in the upper atmosphere (background). Typically, the image of a shower after some pre-processing is an elongated cluster. Its long axis is oriented towards the camera center if the shower axis is parallel to the telescope's optical axis, i.e. if the telescope axis is directed towards a point source. A principal component analysis is performed in the camera plane, which results in a correlation axis and defines an ellipse. If the depositions were distributed as a bivariate Gaussian, this would be an equi-density ellipse. The characteristic parameters of this ellipse (often called Hillas parameters) are among the image parameters that can be used for discrimination. The energy depositions are typically asymmetric along the major axis, and this asymmetry can also be used in discrimination. There are, in addition, further discriminating characteristics, like the extent of the cluster in the image plane, or the total sum of depositions. Data Set Characteristics: Multivariate Number of Instances: 19020 Number of Attributes: 11 Source of the dataset: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. Shortened link: https://bit.ly/2OZRIYd

3.2 Data Processing

The data has been preprocessed using standard scaler for SVM implementation. This method scales data in such a way that the mean value becomes zero and the standard deviation becomes 1. Another preprocessing used for implementing Naïve Bayes and that is Normalization. This process scales data in between 0 and 1. In addition, the Label Encoder has been used to encode the class labels from categorical to numerical values. The preprocessing called Test Train Splitting method has been used to implement all the algorithms.

3.3 Model Development

Naïve Bayes

Dataset Description: The data set used in this work is gamma telescope data set collected from UC Irvine Machine Learning Repository which contains 19020 Instances, 11 attributes(including the class).The data set was generated by a Monte Carlo program to simulate registration of high energy gamma particles. There are no missing values in this dataset. So, data preprocessing is not needed on this dataset.

Data Mining Tool: Anaconda python is used on our project as because anaconda covered python fundamentals, data analysis, and machine learning.

1.Importing Libraries and Loading Datasets:

import pandas as pd

rdata = pd.read$_csv('Downloads.csv')$

2.Splitting Data:

Separate the columns into dependent and independent variables (features and label). Then split those variables into train and test set.

3.Model Generation:

Now, Naive Bayes Model is created by using Sklearn. GaussianNB() function is used in that case.

from sklearn.naive$_bayesimportGaussianNB$

model = GaussianNB()

Train the model using the training sets

model.fit(X$_train, y_train$)

4.Evaluating Model:

After model generation, find out the accuracy.

model.score(X$_train, y_train$)

A classification report is created that contains various statistics required to judge a model. After that, a confusion matrix is created to get clear idea of the accuracy and the fitting of the model.

Performance evaluation

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.91	0.81	2472
1	0.69	0.36	0.48	1332
accuracy			0.72	3804
macro avg	0.71	0.64	0.64	3804
weighted avg	0.71	0.72	0.69	3804

In this section, the model is evaluated using model evaluation metrics like accuracy, precision,recall,f1 score.

Accuracy: Accuracy is the proportion from the set of tuples correctly being classified by the classifier and i got classification rate of 72

Precision: Precision is about how accurate our model is. In the prediction case, logistic Regression model has achieved 0.71 (macro avg) 0.71 (weighted avg) in terms of precision.

Recall: Recall is also referred as true positive (TP) rate. logistic Regression model has achieved 0.64 (macro avg) 0.72 (weighted avg) in terms of recall.

f1 score: In the prediction case, logistic Regression model has achieved 0.64 (macro avg) 0.69 (weighted avg) in terms of F1-score.

6.Model Evaluation using Confusion Matrix : Confusion matrix is used to evaluate the performance of classification model.

co = confusion$_m atrix(y_test, y_pred)$

Confusion matrix: [[2294 192]

[801 517]]

The confusion matrix is in the form of array object and the dimension of this matrix is 2*2 as because this model is binary classification. It have two classes g and h. The diagonal values are accurate predictions and the non-diagonal values are inaccurate predictions. So, the output 2294 and 517 are actual predictions. 801 and 192 are incorrect predictions.

Implementation(logistic regression):

Logistic regression is implemented as the baseline for any binary classification problem. It describes and estimates the relationship between one dependent binary variable and independent variables.

1.load the telescope dataset using the pandas read CSV function:

dataset = pd.read$_csv('Downloads.csv')$

2.To Select Feature,it is needed to divide the given columns into two types of variables -target variable and feature variables.

label = dataset['class']

div = dataset.drop('class', axis = 'columns')

3.Dividing the dataset into training and test set. So, split the dataset using train$_test_split()$

Function.

$X_train, X_test, y_train, y_test = train_test_split(div, label, test_size = 0.2, random_state =$ 0)

4.The Dataset is broken into two parts in a ratio of 80:20.It means 80

Model Generation:

5.Apply scaling on training data using StandardScaler function:

from sklearn.pipeline import make$_pipeline$

from sklearn.preprocessing import StandardScaler

model = make$_pipeline(StandardScaler(), LogisticRegression())$

And then create logistic regression classifier object using logistic regression() function.

6.Then fit the model on the train set using fit()

model.fit(X_train, y_train)

7.Evaluating Model:

After model generation, find out the accuracy.

model.score($X_train, y_train)Performanceevaluation$:

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.90	0.85	2457
1	0.77	0.58	0.66	1347
accuracy			0.79	3804
macro avg	0.78	0.74	0.75	3804
weighted avg	0.79	0.79	0.78	3804

In this section, the model is evaluated using model evaluation metrics like accuracy, precision,recall,f1 score

Accuracy: Here,we got classification rate of 79

Precision: In the prediction case, logistic Regression model has achieved 78

Recall: logistic Regression model has achieved 74

f1 score: In the prediction case, logistic Regression model has achieved 75

9.Model Evaluation using Confusion Matrix : The confusion matrix is in the form of array object and the dimension of this matrix is 2*2 as because this model is binary classification. It have two classes g and h. Confusion matrix is used to evaluate the performance of classification model.

co = confusion$_m atrix(y_test, y_pred)$

Confusion matrix: [[2219 238]

[565 782]]

The diagonal values are accurate predictions and the non-diagonal values are inaccurate predictions. So, the output 2219 and 782 are actual predictions. 238 and 565 are incorrect predictions.

Implementation(Random Forest):

Random Forests is a highly accurate method because of the number of decision trees participating in the process. Random Forest algorithm has multiple decision trees and we randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model.

The first task is to divide data into attributes and label sets. Then divide the data into training and testing sets.

Then we scale our dataset, we can train random forest algorithm to solve this classification problem

The Random Forest Regressor class of the sklearn.ensemble library is used to solve regression problems via random forest.

The metrics used to evaluate an algorithm are accuracy, confusion matrix, precision recall, and F1 values. Performance evaluation Classification Report:

	precision	recall	f1-score	support
0	0.88	0.95	0.91	1237
1	0.89	0.77	0.82	665
accuracy			0.88	1902
macro avg	0.89	0.86	0.87	1902
weighted avg	0.88	0.88	0.88	1902

The output shows the performance of the model on training data.

Accuracy: The accuracy of our model is 0.89 which is considered as good accuracy.

Precision: The model get 0.89 macro avg precision and 0.89 weighted avg precision.

Recall: In case of recall, 87

F1 score: In that case, 87

Confusion matrix: To summarize the performance of a classification algorithm, confusion matrix is used

[[1166 71]

[134 531]]

1166 and 531 are the correct predictions(diagonals). 71 and 134 are incorrect predictions.(off diagonals).

Implementation(neural network):

Multilayer perceptrons are applied to supervised learning problems and the MultilayerPerceptron class encapsulates all data. This methods required to intialize and propagate the network. The two steps that are implemented in this algorithm are:

Calculate the dot product between inputs and weights. The nodes in the input layer are connected with the output layer through weight parameters. Calculate the predicted output is known as feedforward

Pass the result through an activation function. Implementaion with python:

	precision	recall	f1-score	support
0	0.89	0.90	0.89	2499
1	0.81	0.78	0.79	1305
accuracy			0.86	3804
macro avg	0.85	0.84	0.84	3804
weighted avg	0.86	0.86	0.86	3804

Performance evaluation Classification Report:

The output shows the performance of the model on training data.

Accuracy: The accuracy of our model is 86

Precision: The model get 85

Recall: In case of recall, 84

F1 score: In that case, 84

Confusion matrix: [[2256 243]

[287 1018]]

2256 and 1018 are the correct predictions(diagonals). 287 and 243 are incorrect predictions.(off diagonals).

K-nearest neighbor : In our work, we used Python's Scikit-Learn library to implement the algorithms.For K-Nearest Neighbor we are using the given procedure and functions for the code.

1.The first procedure we followed was to load the selected dataset using the pandas read function.

2.We divided the columns into labels and attributes.

3.The dataset was splitted into training and testing set. The function used for this step were $\mathrm{X}_t rain, X_t est, y_t rain, and y_t est$.

4. We took .8 training data and .2 test data.

5. The value selected for k was 5. The parameter used for it is $n_neighbors = 5$.

Here we used the class KNeighborsClassifier which is imported from sklearn.neighbors.

6. The final step of implementation was to predict on the test data and evaluate by using confusion matrix, precision, recall metrics.

The confusion matrix and classification report methods are also from sklearn.metrics.

This matrix is used to evaluate the performance. Performance evaluation:

```
Accuracy score:  0.805993690851735
[[1122   94]
 [ 275  411]]
              precision    recall  f1-score   support

           0       0.80      0.92      0.86      1216
           1       0.81      0.60      0.69       686

    accuracy                           0.81      1902
   macro avg       0.81      0.76      0.77      1902
weighted avg       0.81      0.81      0.80      1902
```

Accuracy : Now after the implementation we got the accuracy rate of 80.599

Precision : K-nearest Neighbor got 0.81 of macro avg. and weighted avg.

Recall : In terms of recall, the result for this algorithm is 0.7 that of macro avg and 0.81 weighted avg.

f1 score : The algorithm gave 0.77 of macro avg. and 0.80 weighted avg. with respect to accuracy.

Confusion matrix : The confusion matrix achieved from the implemented algorithm is [1122 275] and [94 411]

Linear Regression : Simple Linear Regression is used for the project work. The implementation of the Python Scikit-Learn library is described thoroughly.

1. For simple linear regression two variables was taken.

2. We divided the data into attributes and labels. Here, attributes are the independent one while labels are the dependent one.

We predicted the dependent variables. The attributes are stored in the X variable. We specified "-1" as the range for columns.

All the columns except the last one are contained by X.

the variable y contains "Score" label which is specified as 1.

3. The dataset was splitted into training and testing set. The function used for this step were $X_train, X_test, y_train, and y_test.test_size = 0.2 which means 20$

4. LinearRegression class is imported. We called for the method "fit()" alongside train data using "regressor" as class.

5. To evaluate the algorithm by using confusion matrix, precision, recall and others we took round value using the function $y_pred.round().Performanceevaluation$

```
Accuracy score:  0.33688621787822615
[[2227  226    0]
 [ 597  748    6]
 [   0    0    0]]
              precision    recall  f1-score   support

         0.0       0.79      0.91      0.84      2453
         1.0       0.77      0.55      0.64      1351
         2.0       0.00      0.00      0.00         0

    accuracy                           0.78      3804
   macro avg       0.52      0.49      0.50      3804
weighted avg       0.78      0.78      0.77      3804
```

Accuracy : Linear Regression got the accuracy rate of only 33.688
Precision : The output we got for precision of Linear Regression is 52
Recall : In case of recall, the result for this algorithm is 49
f1 score : For the f1 score, the algorithm remarked 50
Confusion matrix : The confusion matrix achieved from the implemented algorithm is [2227 597] and [226 748].

Decision Tree Induction:

TRP of Decision tree: 0.8447927199191102 FRP of Decision tree 0.17257142857142857
Classification report:

```
Classification Report of Decision Tree Induction:


Accuracy of DT:  83.8415702769015 %

              precision    recall  f1-score   support

           0       0.84      0.92      0.88      3644
           1       0.83      0.70      0.76      2062

   micro avg       0.84      0.84      0.84      5706
   macro avg       0.83      0.81      0.82      5706
weighted avg       0.84      0.84      0.84      5706
```

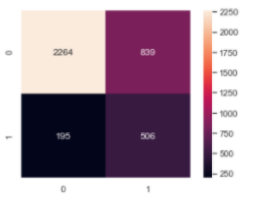
SVM Classifier:

TPR of SVM: 0.8168507577703571 FPR of SVM: 0.25592939878654164
Classification report:

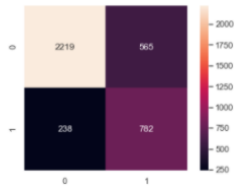
```
=== Summary ===

Correctly Classified Instances        4507               78.987  %
Incorrectly Classified Instances      1199               21.013  %
Kappa statistic                          0.5138
Mean absolute error                      0.2101
Root mean squared error                  0.4584
Relative absolute error                 46.0963 %
Root relative squared error             96.0521 %
Total Number of Instances             5706

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.898    0.411    0.802      0.898    0.847      0.522    0.744     0.786     g
               0.589    0.102    0.758      0.589    0.663      0.522    0.744     0.591     h
Weighted Avg.  0.790    0.302    0.786      0.790    0.783      0.522    0.744     0.718

=== Confusion Matrix ===

    a    b   <-- classified as
 3328  376 |    a = g
  823 1179 |    b = h
```

3.4 Results

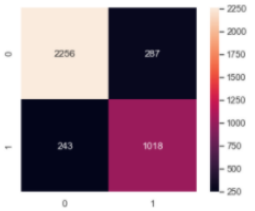
Seaborn package is used in Python to get a more vivid display of the matrix. Visualizing Confusion Matrix using Heatmap:



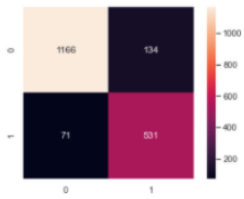
Naïve Bayes



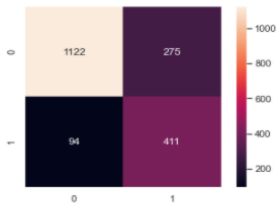
Logistic Regression



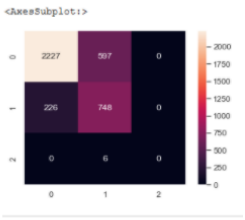
Neural Network



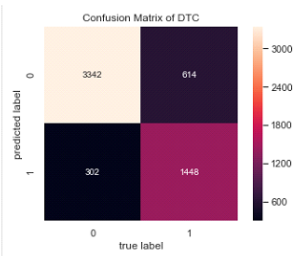
Random Forests

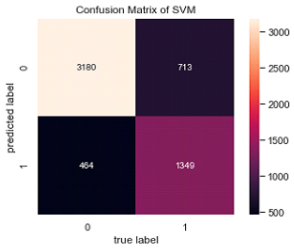


KNN



Linear Regression

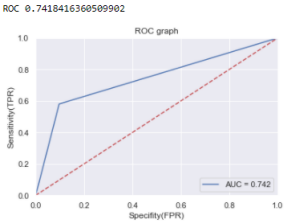




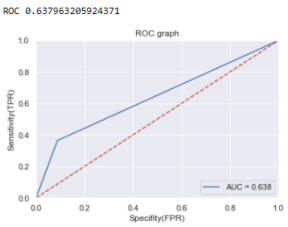
ROC Curve:

ROC curve is needed to visualize and compare the performance of classifier methods .This curve is created by plotting true positive rate against the false positive rate. This curve is a two dimensional graph and it plots two parameters -True Positive Rate which is plotted on the Y axis False Positive Rate which is is plotted on the X axis. It represents the tradeoff between sensitivity (true positive rate) and 1-specificity (false positive rate). If AUC score is 1 then it represents perfect classifier. If AUC score is 0.5 then it represents a worthless classifier.

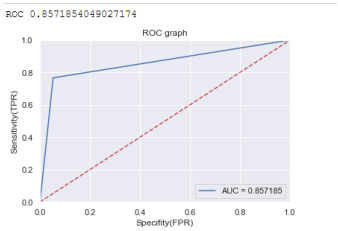
Naive Bayes:



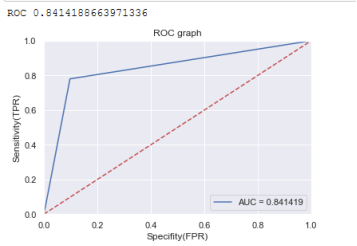
logistic regression:



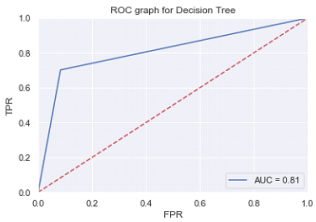
Neural Network:



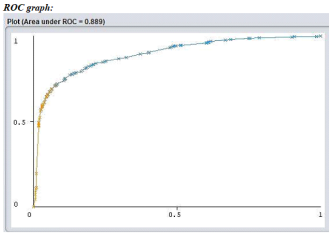
Random Forests:



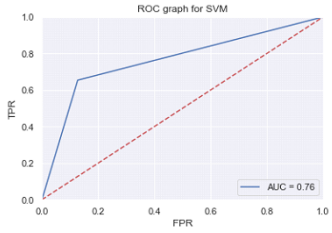
Decision Tree:

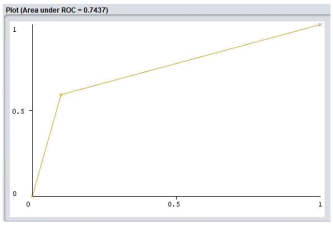


ROC graph:



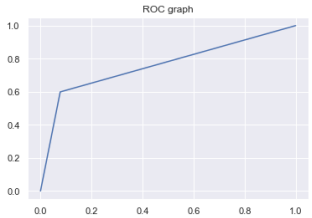
SVM:



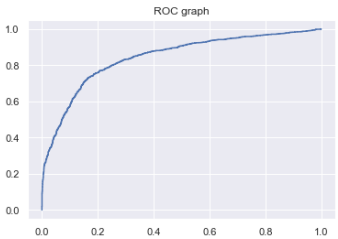


KNN:

```
ROC 0.7609113664262698
3]: [<matplotlib.lines.Line2D at 0x1fdd5106dc0>]
```



Linear Regression:



Classifier	Accuracy	Precision(g)/(h)	Recall(g)/(h)	F1 Score(g)/(h)	Specifity	Sensitivity
Naïve Bayes	0.72	0.73/0.69	0.91/0.36	0.81/0.48	0.638	0.31
Logistic Regression	0.79	0.80/0.77	0.90/0.58	0.85/0.66	0.7418	0.233
Neural Network	0.87	0.89/0.81	0.90/0.78	0.89/0.79	0.841	0.193
Random Forests	0.88	0.88/0.89	0.95/0.77	0.91/0.82	0.857	0.113
SVM	0.79	.854/.834	0.924/.707	0.887/0.765	0.889	0.293
Linear Regression	0.336	0.79/0.77	0.91/0.55	0.84/0.64	0.843	0.232
KNN	0.80	0.80/0.83	0.93/0.59	0.86/0.69	0.758	0.171
Dicision Tree	0.84	0.84/0.83	0.84/0.81	0.83/82 0.81	0.256	0.817

Table 1: Comparison of different machine learning algorithms by evaluation metrics

From the table we can find that Random Forest algorithm gives the best accuracy and Linear regression gives the lowest accuracy. The dataset is most sensitive for SVM but for specifity and sensitivity ratio the dataset is best for random forest. According to the ROC curve SVM should be best for the dataset, but according to accuracy result we can analyze Random forest gives best result. Neural network also gives the near best result .Comparing precision, recall, f1score it gives better result for class 'g' then class 'h'. At the end we can say for this dataset Random forest is most preferable.

4 Conclusion

4.1 Challenges

Comparison of different machine learning algorithms will be helpful in future for data analyst to save time. In our project, we faced many challenges while choosing the dataset. The dataset which provides best result almost every time is the perfect dataset, but in term of comparison it might not be the useful one. Thats why while choosing and comparing datasets to find and understand the usefulness of dataset we faces many difficulties.

4.2 Limitations

After the selection of dataset, we faced many limitations. Remarkable limitations are the appearance of the smaller scope as a result of limited time and knowledge. Beside it, there can be a prime difficulty of multicollinearity between the predicted variables in the dataset, which was not analysed. Our objective was to compare the selected algorithms. But in future, in case of similar patterned dataset, analyst will be able to determine easily which will be the proper algorithm to apply to get the preferable result.

4.3 Future Directions

For this project we only compared the Supervised Machine Learning algorithms. We can assure that there are opportunities of comparison and other source of work between unsupervised Machine Learning algorithms. Even there can be some hybrid algorithms which are the advantageous mixture of both section of Machine Learning algorithms.

References

[1] Abdulsalam Alarabeyyat, Mohannad Alhanahnah, et al. Breast cancer detection using k-nearest neighbor machine learning algorithm. In *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*, pages 35–39. IEEE, 2016.

[2] Omar MK Alhawi, James Baldwin, and Ali Dehghantanha. Leveraging machine learning techniques for windows ransomware network traffic detection. In *Cyber Threat Intelligence*, pages 93–106. Springer, 2018.

[3] Peter C Austin and Juan Merlo. Intermediate and advanced topics in multilevel logistic regression analysis. *Statistics in medicine*, 36(20):3257–3277, 2017.

[4] Deepika Bansal, Rita Chhikara, Kavita Khanna, and Poonam Gupta. Comparative analysis of various machine learning algorithms for detecting dementia. *Procedia computer science*, 132:1497–1502, 2018.

[5] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.

[6] Ayon Dey. Machine learning algorithms: a review. *International Journal of Computer Science and Information Technologies*, 7(3):1174–1179, 2016.

[7] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[8] Hamid Mohamadlou, Anna Lynn-Palevsky, Christopher Barton, Uli Chettipally, Lisa Shieh, Jacob Calvert, Nicholas R Saber, and Ritankar Das. Prediction of acute kidney injury with a machine learning algorithm using electronic health record data. *Canadian journal of kidney health and disease*, 5:2054358118776326, 2018.

[9] Yaohao Peng and Mateus Hiro Nagata. An empirical overview of nonlinearity and overfitting in machine learning using covid-19 data. *Chaos, Solitons & Fractals*, 139:110055, 2020.

[10] K Sravani and P Srinivasu. Comparative study of machine learning algorithm for intrusion detection system. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*, pages 189–196. Springer, 2014.

[11] K Vembandasamy, R Sasipriya, and E Deepa. Heart diseases detection using naive bayes algorithm. *International Journal of Innovative Science, Engineering & Technology*, 2(9):441–444, 2015.