CHRONIC KIDNEY DISEASE PREDICTION

RISHAB CHAKRABARTI (102003688)
ANIRUDH BRIAN CHADHA (102003658)

INTRODUCTION

The main function of the kidney is to filter the blood in the body. Chronic kidney disease, also called chronic kidney failure, involves a gradual loss of kidney function. Your kidneys filter wastes and excess fluids from your blood, which are then removed in your urine. Advanced chronic kidney disease can cause dangerous levels of fluid, electrolytes and wastes to build up in your body. Kidney disease is often caused by diabetes and high blood pressure. In the early stages of chronic kidney disease, you might have few signs or symptoms. You might not realize that you have kidney disease until the condition is advanced. Treatment for chronic kidney disease focuses on slowing the progression of kidney damage, usually by controlling the cause. But, even controlling the cause might not keep kidney damage from progressing. Chronic kidney disease can progress to end-stage kidney failure, which is fatal without artificial filtering (dialysis) or a kidney transplant.

Engineers and medical researchers are trying to develop machine learning algorithms and models that can identify chronic kidney disease at an early stage. The problem is that the data generated in the health industry is large and complex, making data analysis difficult. However, we can process this data into a data format using data mining technology, and then this data can be translated into machine learning algorithms. These models also teach the patient how to live a healthy life and help the doctor see the risk and severity of the disease, as well as how to proceed with the treatment in the future. It may be possible to identify patterns of data collection using ANN, mining methods, and the future occurrence of certain diseases that may cause harm can be predicted in advance. Data mining methods are specifically used in models proposed to predict kidney disease. The database can be extended with more information than the existing chronic kidney disease model. That is, more information obtained from patients with chronic kidney disease can be added (although the information must be reliable), which will increase the accuracy of the prediction.

This Document is divided into further sections. Section 1 contains the problem statement and the need of machine learning in this problem. Section 2 contains the Background about this problem. Section 3 describes the dataset and its attributes. Section 4 describes the Architecture of the entire Machine Learning Process . Section 5 contains all the algorithms used and a description about them. Section 6 is the result section from the Dataset

BACKGROUND

There have been articles that have solved this problem in the Past they are:

1. Bemando et al. investigated the relationship between blood-related diseases and their features utilising classifier methods such as Gaussian NB, Bernoulli NB, and Random Forest. These three algorithms anticipate and offer statistical findings in a variety of ways

2. Kumar and Polepaka devised a technique for illness prediction in the medical field. They employed Random Forest and CNN as well as other machine learning methods. For illness dataset classification, precision, recall, and F1-score, these algorithms deliver better results.

3. Sing et al. developed a technique for predicting medical-related illness datasets. For improved prediction, they utilised a support vector machine classifier. The accuracy ranged from 73 to 91 percent, and the author eventually improved accuracy to 91 percent

4. Desai et al. devised a technique for illness prediction in the medical field. The author employed back-propagation NN and LR classification algorithms in this study. These two strategies provide distinct outcomes, with statistical analysis and logistic regression yielding a more accurate model than other algorithms

5. Patil et al. created a database for ECG arrhythmia-related medical conditions. On a disease dataset, the authors employed machine learning approaches such as Support Vector Machine and Cuckoo search optimised Neural Network, and support vector machine estimated 94.44 percent improved accuracy

6. Observed illness dataset for statistical analysis by Liu et al. They estimated superior findings for specificity, sensitivity, positive predictive value, and negative predictive value using machine learning approaches such as support vector machines

7. For better statistical analysis outcomes, Acharya et al. reviewed medical linked illness dataset. They employed several machine learning techniques, such as CNN, and applied machine learning algorithms to the ECG dataset, achieving a classification accuracy of 94 percent

8. Wasle et al. devised a statistical analysis technique. For the examination of the Chronic Kidney Disease dataset, the authors employed a variety of machine learning approaches. They used Nave Bayes, Decision Trees, and Random Forest to improve prediction, and they discovered that Random Forest computed greater classification accuracy than the other algorithms

9. On the Kidney disease dataset, Nithya et al. developed a method for categorization and cluster-based analysis. On diverse sets of photos, the authors utilized the K-Means clustering technique to collect the closest familiar images. They calculated 99.61 percent classification accuracy using Artificial Neural Networks for Kidney Disease Image Prediction

10. Al Imran et al. examined the use of machine learning techniques to analyze datasets for chronic renal disease. For statistical analysis such as F1-score, Precision, Recall, and AUC, the authors employed Logistic Regression and Feed forward Neural Network and generated better results than previous algorithms

11. Navaneeth and Suchetha devised a method for predicting chronic renal disease using a dataset. They employed machine learning methods such as CNN and SVM. Authors estimated greater accuracy, sensitivity, and specificity findings after the prediction

DATASET DESCRIPTION

DATA COLLECTION

This Dataset was created by L.Jerlin Rubini, a Research Scholar at Alagappa University(email id :jel.jerlin@gmail.com, Contact No:+91-9597231281). He has collected this from Dr. P. Soundarapandian. M.D., D.M (Senior Consultant Nephrologist), Apollo Hospitals, Manager, Madurai Main Road, Karaikudi, Tamil Nandu, India. During this entire process he was guided by Dr. P . Eswaran Assistant Professor, Department of Computer Science and Engineering, Alagappa University, Karaikudi, Tamil Nadu, India. email id:eswaranperumal@gmail.com

The Dataset consists of 400 Rows and 25 Columns.

We use the following representation to collect the dataset:

age - age
bp - blood pressure
sg - specific gravity
al - albumin
su - sugar
rbc - red blood cells
pc - pus cell
pcc - pus cell clumps
ba - bacteria
bgr - blood glucose random
bu - blood urea
sc - serum creatinine
sod - sodium
pot - potassium
hemo - hemoglobin
pcv - packed cell volume
wc - white blood cell count
rc - red blood cell count
htn - hypertension
dm - diabetes mellitus
cad - coronary artery disease
appet - appetite
pe - pedal edema
ane - anaemia
class – class

Attribute	id	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc
Min	0	2	50	1.005	0	0	22	1.5	0.4	4.5	2.5	3.1	9	2200	2.1
Max	399	90	180	1.025	5	5	490	391	76	163	47	17.8	54	26400	8
Range	0-399	2-90	50-180	1.005-1.025	0-5	0-5	22-490	1.5-391	0.4-76	4.5-163	2.5-47	3.1-17.8	9-54	2200-26400	2.1-8
Null values	0	9	12	47	46	49	44	19	17	87	88	52	71	106	131
Mean Values	199.5	51.483376	76.4690722	1.01740793	1.01694915	0.45014245	148.036517	57.4257218	3.07245431	137.528754	4.62724359	12.5264368	38.8844985	8406.12245	4.70743494

QUALITATIVE ATTRIBUTES

rbc	Count
abnormal	47
normal	201
Null	152

dm	Count
no	261
yes	137
null	2

pc	Count
abnormal	76
normal	259
Null	65

cad	Count
no	364
yes	34
null	2

pcc	Count
not present	354
present	42
null	4

appet	Count
good	317
poor	82
null	1

htn	Count
no	251
yes	147
null	2

pe	Count
no	323
yes	76
null	1

ane	Count
no	339
yes	60
null	1

classification	Count
ckd	250
notckd	150
null	0

QUANTITATIVE ATTRIBUTES

Attribute	Min	Max	Range	Null values	Mean Values
id	0	399	0-399	0	199.5
age	2	90	2-90	9	51.483376
bp	50	180	50-180	12	76.4690722
sg	1.005	1.025	1.005-1.025	47	1.01740793
al	0	5	0-5	46	1.01694915
su	0	5	0-5	49	0.45014245
bgr	22	490	22-490	44	148.036517
bu	1.5	391	1.5-391	19	57.4257218
sc	0.4	76	0.4-76	17	3.07245431
sod	4.5	163	4.5-163	87	137.528754
pot	2.5	47	2.5-47	88	4.62724359
hemo	3.1	17.8	3.1-17.8	52	12.5264368
pcv	9	54	9-54	71	38.8844985
wc	2200	26400	2200-26400	106	8406.12245
rc	2.1	8	2.1-8	131	4.70743494

DATA PREPROCESSING

DROPPING UNWANTED COLUMNS

This Dataset has an attribute called ID so we are removing that from the dataset. This is because the ID column is not required in ML Training.

REPLACING QUALITATIVE ATTRIBUTES WHICH ARE NULL

This Dataset has Qualitative/Categorical attributes that have Null Values in them. So these Null values can be replaced by the Mode in that Column.

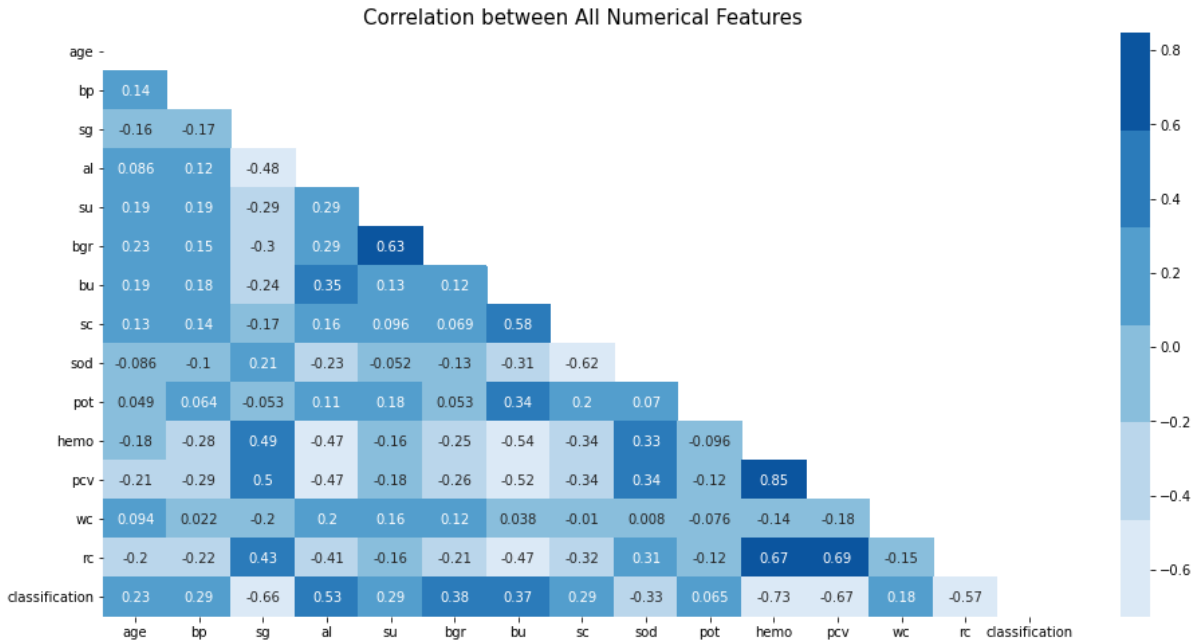
REPLACING QUANTITATIVE ATTRIBUTES WHICH ARE NULL

This Dataset has Quantitative/Numeric attributes that have Null Values in them. So these Null values can be replaced by the Median in that Column.

MAPPING CLASSIFICATION VALUES

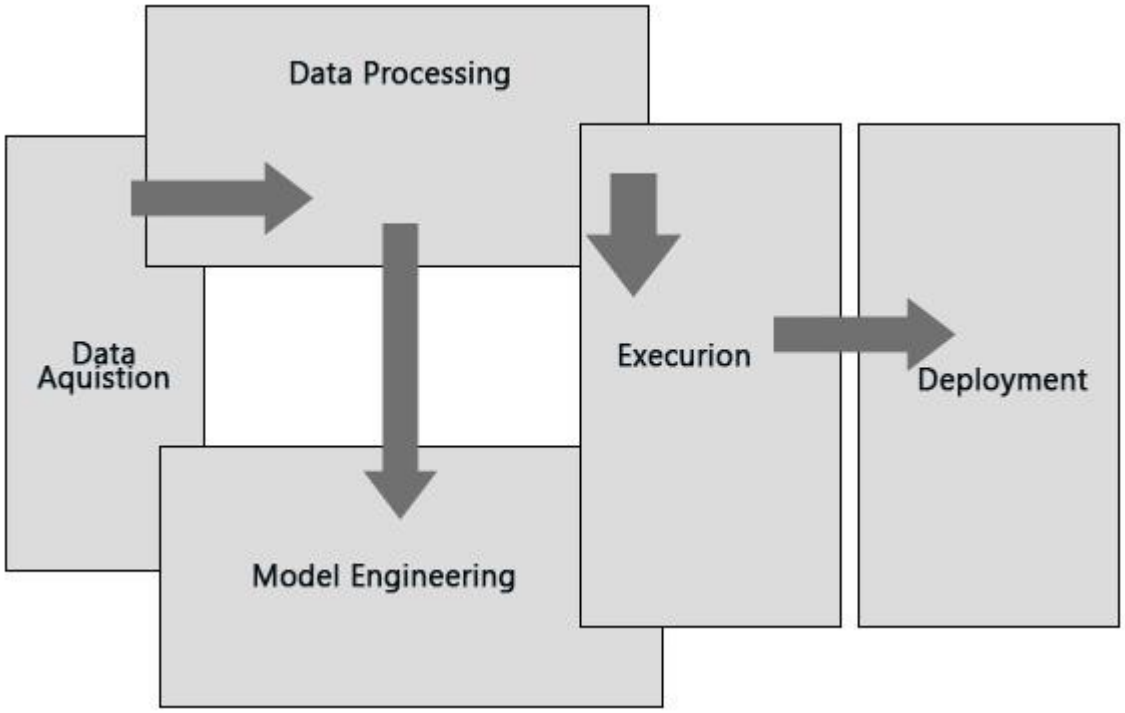
This Dataset has a Column Classification so we have mapped ckd as 1 and notckd as 0.

FEATURE SELECTION



We can see the Correlation between the different features in the Dataset. We can drop the most correlated value which is PCV from the Dataset.

ARCHITECTURE



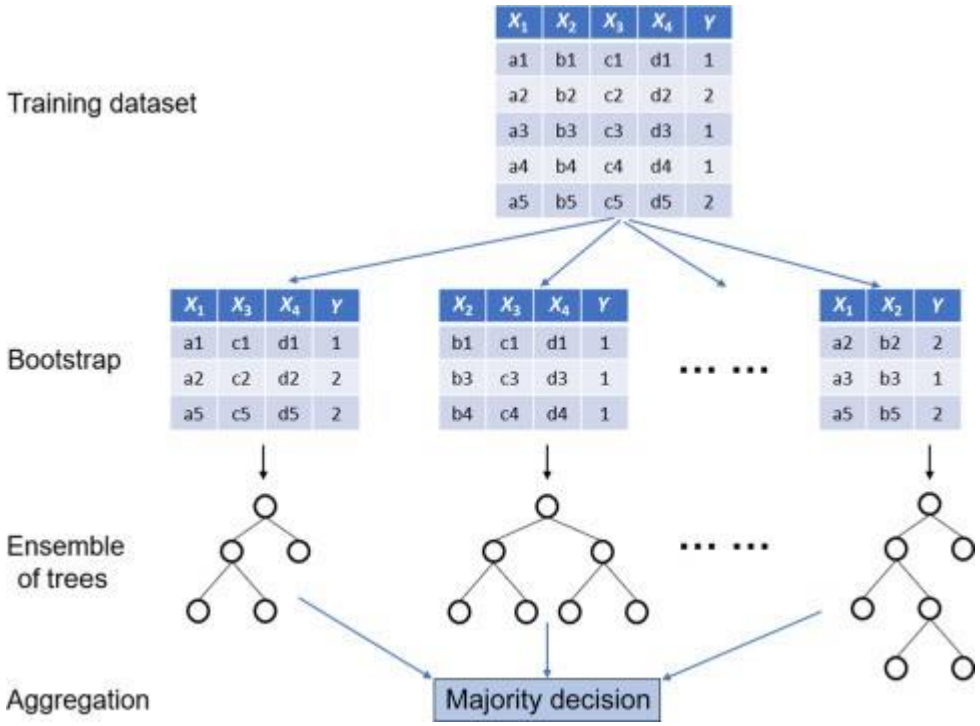
This Chronic Kidney Disease Dataset has labeled Data in it. Also, this data was accessed from the Apollo Hospital Patients Records. So, the creator of this dataset has accessed the Database and filtered out the necessary parameters required to predict Kidney Disease. So initially we have Raw and Uncleaned Data, to begin with. This completes our Data Acquisition procedure. This Raw and Uncleaned Data is passed through a series of Data Cleaning Methods. This Raw Data contains some Null Values in the Dataset that must be removed before ML training can be performed on it. The Dataset consists of Null values in Various fields that are Qualitative as well as Quantitative. We have different techniques to deal with them. To remove the Null Values in the qualitative column we replace the null values with the Mode of the respective column. Next in line are the quantitative features in the Dataset for which we replace them with the median of the column. Also, the column Classification needed labelling so it was encoded as ckd as 1 and notckd as 0. After a bit of analysis of the Correlation Matrix between all the Features, the Column PCV is dropped from the Dataset because of its highly correlated value. The Dataset is cleaned and we are done with Data Pre-processing. Because we are provided with Labelled Data, Supervised Machine Learning Algorithms are used for prediction. The types of Machine Learning algorithms to be used here is Classification because its classifying whether the Patient has Kidney disease or not . 1 for yes and 0 for no. Out of all the possible Classification Algorithms. We need to select which performs the best on this dataset and returns us with high accuracy on the Test data. The different types of Machine Learning Algorithms used are stated below in the next section.

ALGORITHMS

1. EXTRA TREES CLASSIFIER

Extra trees (short for extremely randomized trees) is an ensemble supervised machine learning method that uses decision trees. This method is similar to random forests but can be faster. The extra trees algorithm, like the random forests algorithm, creates many decision trees, but the sampling for each tree is random, without replacement. This creates a dataset for each tree with unique samples. A specific number of features, from the total set of features, are also selected randomly for each tree. The most important and unique characteristic of extra trees is the random selection of a splitting value for a feature. Instead of calculating a locally optimal value using Gini or entropy to split the data, the algorithm randomly selects a split value. This makes the trees diversified and uncorrelated.

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2(p_i)$$



2.K NEAREST NEIGHBOURS

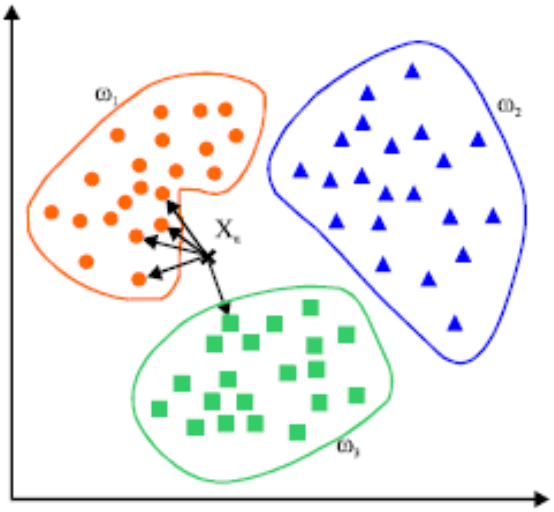
The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another. Regression problems use a similar concept as classification problem, but in this case, the average the k nearest neighbors is taken to make a prediction about a classification. The main distinction here is that classification is used for discrete values, whereas regression is used with continuous ones. However, before a classification can be made, the distance must be defined. Euclidean distance is most commonly used.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

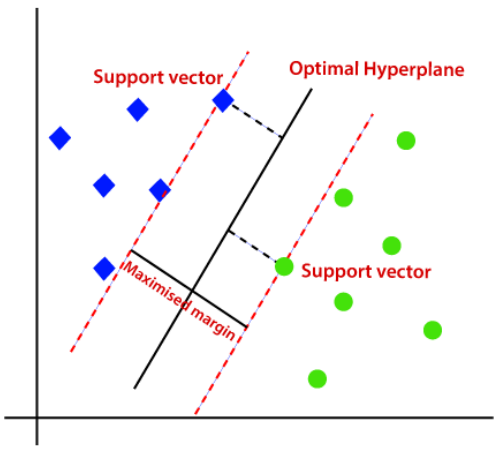
Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$



3.LINEAR SVM

Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

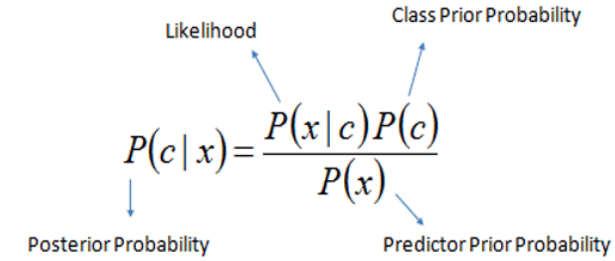


$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \, \phi(x_i) \cdot \phi(x_j)$$

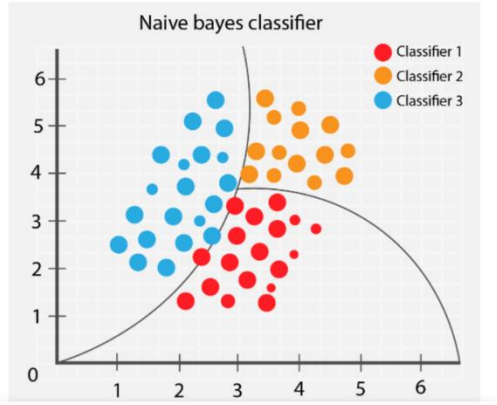
$$where\ 0 \le \alpha_i \le C\ \forall\ i$$

4.NAIVE BYES

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in *text classification* that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identifying that it is an apple without depending on each other. **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem

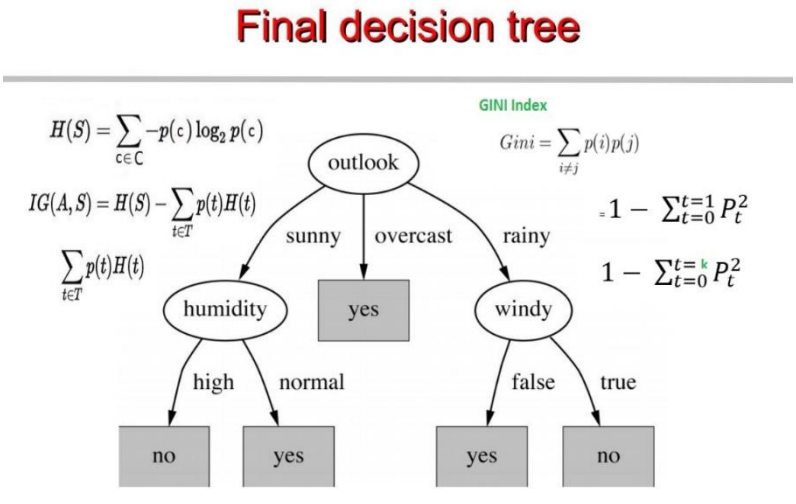


$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$



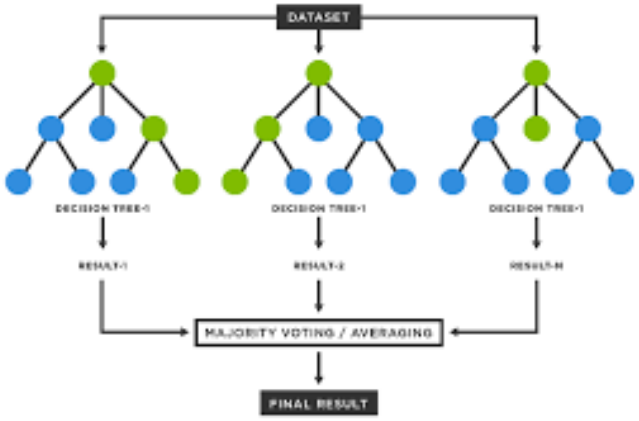
5.DECISION TREE

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.



6.RANDOM FOREST

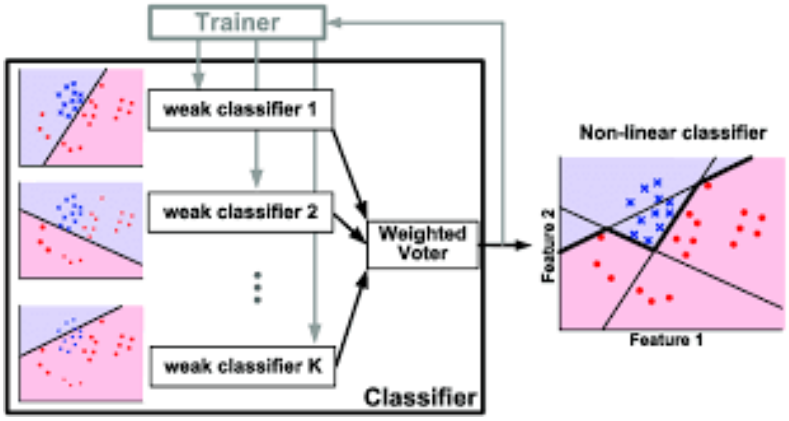
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



$$RFfi_i = \frac{\sum_j normfi_{ij}}{\sum_{j \in all\ features, k \in all\ trees} normfi_{jk}}$$

7. ADABOOST

Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple “weak classifiers” into a single “strong classifier”. It was formulated by Yoav Freund and Robert Schapire.



$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

8. QUADRATIC DISCRIMINANT ANALYSIS

QDA is a variant of LDA in which an individual covariance matrix is estimated for every class of observations. QDA is particularly useful if there is prior knowledge that individual classes exhibit distinct covariances. A disadvantage of QDA is that it cannot be used as a dimensionality reduction technique.In QDA, we need to estimate ΣkΣk for each class k∈{1,…,K}k∈{1,…,K} rather than assuming Σk=ΣΣk=Σ as in LDA. The discriminant function of LDA is quadratic in xx:

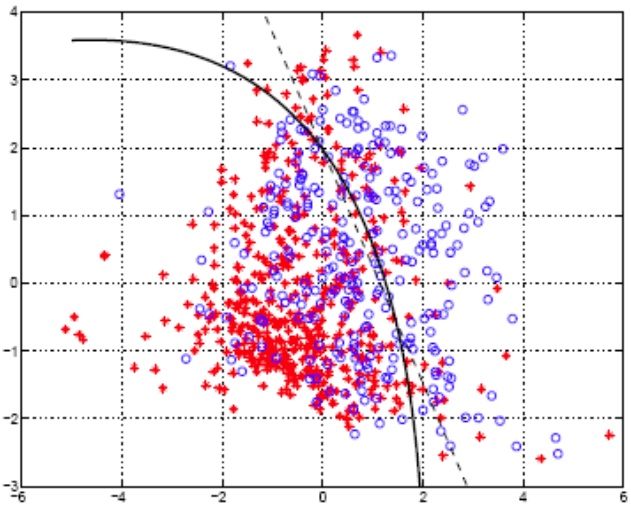
$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k\,.$$

Since QDA estimates a covariance matrix for each class, it has a greater number of effective parameters than LDA. We can derive the number of parameters in the following way.

- We need KK class priors πkπk. Since ∑Ki=1πk=1∑i=1Kπk=1, we do not need a parameter for one of the priors. Thus, there are K−1K−1 free parameters for the priors.
- Since there are KK centroids, μkμk, with pp entries each, there are KpKp parameters relating to the means.
- From the covariance matrix, ΣkΣk, we only need to consider the diagonal and the upper right triangle. This region of the covariance matrix has p(p+1)2p(p+1)2 elements. Since KK such matrices need to be estimated, there are Kp(p+1)2Kp(p+1)2 parameters relating to the covariance matrices.

Thus, the effective number of QDA parameters is K−1+Kp+Kp(p+1)2K−1+Kp+Kp(p+1)2.

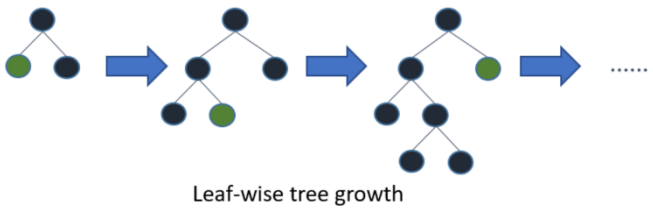
Since the number of QDA parameters is quadratic in pp, QDA should be used with care when the feature space is large.



9. LIGHT GRADIENT BOOSTING MACHINE

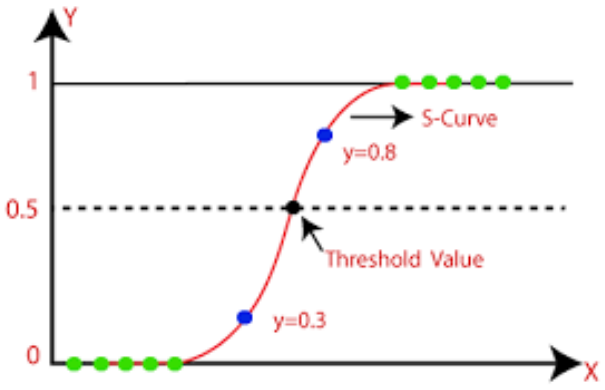
LightGBM is a gradient-boosting framework based on decision trees to increase the efficiency of the model and reduces memory usage. It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfils the limitations of the histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of the LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks. Gradient-based One Side Sampling Technique for LightGBM. Different data instances have varied roles in the computation of information gain. The instances with larger gradients (i.e., under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g., larger than a predefined threshold, or among the top percentiles), and only randomly drops those instances with small gradients to retain the accuracy of information gain estimation. This treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range.

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right)$$



10. LOGISTIC REGRESSION

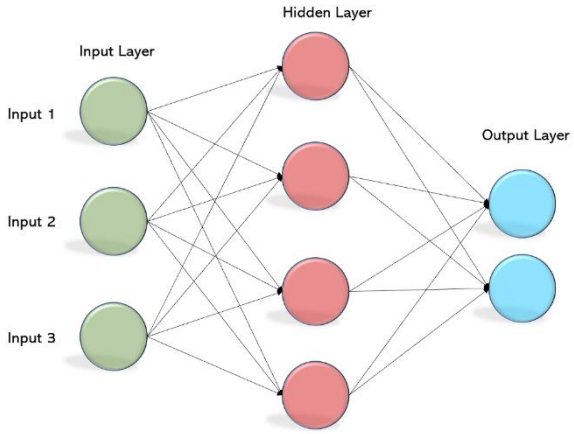
Logistic regression is a statistical method that is used to model a binary response variable based on predictor variables. Although initially devised for two-class or binary response problems, this method can be generalized to multiclass problems. However, our example tumor sample data is a binary response or two-class problem, therefore we will not go into the multiclass case in this chapter. Logistic regression is very similar to linear regression as a concept and it can be thought of as a “maximum likelihood estimation” problem where we are trying to find statistical parameters that maximize the likelihood of the observed data being sampled from the statistical distribution of interest. This is also very related to the general cost/loss function approach we see in supervised machine learning algorithms. In the case of binary response variables, the simple linear regression model, such as yi∼β0+β1xiyi∼β0+β1xi, would be a poor choice because it can easily generate values outside of the 00 to 11 boundary. What we need is a model that restricts the lower bound of the prediction to zero and an upper bound to 11. The first thing to this requirement is to formulate the problem differently. If y can only be 00 or 11, we can formulate y as a realization of a random variable that can take the values one and zero with probabilities pipi and 1−pi1−pi, respectively



$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

11. MLP CLASSIFIER

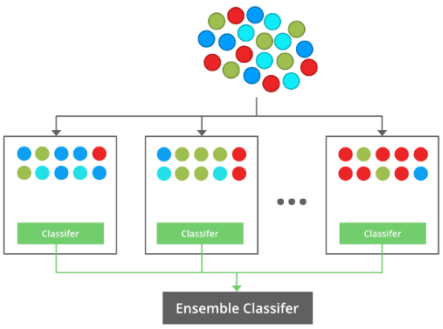
MLP Classifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLP Classifier relies on an underlying Neural Network to perform the task of classification. One similarity though, with Scikit-Learn's other classification algorithms is that implementing MLP Classifier takes no more effort than implementing Support Vectors or Naive Bayes or any other classifiers from Scikit-Learn. The multilayer perceptron (MLP) is a feedforward artificial neural network model that maps input data sets to a set of appropriate outputs. An MLP consists of multiple layers and each layer is fully connected to the following one. The nodes of the layers are neurons with nonlinear activation functions, except for the nodes of the input layer. Between the input and the output layer there may be one or more nonlinear hidden layers.



$$u(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i.$$

12. XG BOOST

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.



Original Data

Bootstrapping

Aggregating

Bagging

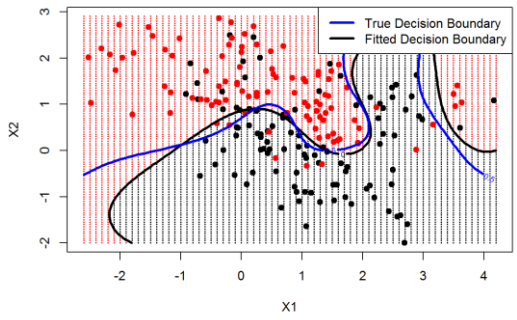
Real value (label) known from the training data-set

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Can be seen as f(x + Δx) where x = $\hat{y_i}^{(t-1)}$

13. RADIAL SVM

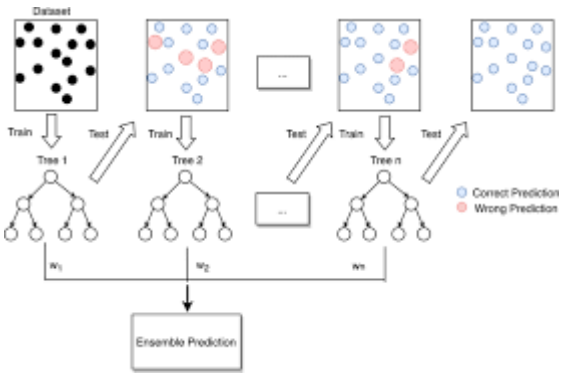
Radial kernel support vector machine is a good approach when the data is not linearly separable. The idea behind generating nonlinear decision boundaries is that we need to do some non linear transformations on the features Xi which transforms them to a higher dimension space. We do this nonlinear transformation using the Kernel trick. Now there are 2 hyperparameters in the SVM i.e the regularization parameter **'c'** and γ. We can implement cross-validation to find the best values of both these tuning parameters which affect our classifier's C(X)C(X) performance. Another way of finding the best value for these hyperparameters is by using certain optimization techniques such as Bayesian Optimization**.**



$$k(x, y) = \exp(-\gamma \sum_{j=1}^{p}(x_{ij} - y_{ij})^2)$$

14. GRADIENT BOOSTING CLASSIFIER

This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the loss function, e.g. binary or multiclass log loss. Binary classification is a special case where only a single regression tree is induced. In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor to *the residual errors made by the previous predictor*



$$\gamma = \frac{Sum\ of\ residuals}{Sum\ of\ each\ p(1-p)\ for\ each\ sample\ in\ the\ leaf}$$

RESULTS

PARAMETERS USED IN ALGORITHMS

1.EXTRA TREES

sklearn.ensemble.**ExtraTreesClassifier**

1. n_estimators : int, default=100
2. Criterion:{“gini”, “entropy”, “log_loss”}, default=”gini”
3. max_depth:int, default=None
4. min_samples_split:int or float, default=2
5. min_samples_leaf:int or float, default=1
6. min_weight_fraction_leaf:float, default=0.0
7. max_features:{“sqrt”, “log2”, None}, int or float, default=”sqrt”
8. max_leaf_nodes:int, default=None
9. min_impurity_decrease:float, default=0.0
10. bootstrap:bool, default=False
11. oob_score:bool, default=False
12. n_jobs:int, default=None
13. random_state:int, RandomState instance or None, default=None
14. verbose:int, default=0
15. warm_start:bool, default=False
16. class_weight{“balanced”, “balanced_subsample”}, dict or list of dicts, default=None
17. ccp_alpha:non-negative float, default=0.0
18. max_samples:int or float, default=None

2.K NEAREST NEIGHBOURS

sklearn.neighbors.**KNeighborsClassifier**

1. n_neighbors:int, default=5
2. weights{‘uniform’, ‘distance’} or callable, default=’uniform’
3. algorithm{‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’}, default=’auto’
4. leaf_size:int, default=30
5. p:int, default=2
6. metric:str or callable, default=’minkowski’
7. metric_params:dict, default=None
8. n_jobs:int, default=None

3.LINEAR SVM

sklearn.svm.**LinearSVC**

1. penalty:{‘l1’, ‘l2’}, default=’l2’
2. loss:{‘hinge’, ‘squared_hinge’}, default=’squared_hinge’
3. dual: bool, default=True
4. tol:float, default=1e-4
5. C:float, default=1.0
6. multi_class:{‘ovr’, ‘crammer_singer’}, default=’ovr’
7. fit_intercept:bool, default=True
8. intercept_scaling : float, default=1
9. class_weight:dict or ‘balanced’, default=None
10. verbose:int, default=0
11. random_state:int, RandomState instance or None, default=None
12. max_iter:int, default=1000

4. NAIVE BYES

sklearn.naive_bayes.**GaussianNB**

1. Priors:array-like of shape (n_classes,)
2. var_smoothing:float, default=1e-9

5. DECISION TREE

sklearn.tree.**DecisionTreeClassifier**

1. criterion{“gini”, “entropy”, “log_loss”}, default=”gini”
2. splitter {“best”, “random”}, default=” best”
3. max_depth:int, default=None
4. min_samples_split:int or float, default=2
5. min_samples_leaf:int or float, default=1
6. min_weight_fraction_leaf:float, default=0.0
7. max_features:int, float or {“auto”, “sqrt”, “log2”}, default=None
8. random_state:int, RandomState instance or None, default=None
9. max_leaf_nodes:int, default=None
10. min_impurity_decrease:float, default=0.0
11. class_weight:dict, list of dict or “balanced”, default=None
12. ccp_alpha:non-negative float, default=0.0

6.RANDOM FOREST

sklearn.ensemble.RandomForestClassifier

1. n_estimators:int, default=100
2. criterion:{“gini”, “entropy”, “log_loss”}, default=”gini”
3. max_depth:int, default=None
4. min_samples_split:int or float, default=2
5. min_samples_leaf:int or float, default=1
6. min_weight_fraction_leaf:float, default=0.0
7. max_features:{“sqrt”, “log2”, None}, int or float, default=”sqrt”
8. max_leaf_nodes:int, default=None
9. min_impurity_decrease:float, default=0.0
10. bootstrap:bool, default=True
11. oob_score:bool, default=False
12. n_jobs:int, default=None
13. random_state:int, RandomState instance or None, default=None
14. verbose:int, default=0

15. warm_start:bool, default=False
16. class_weight:{“balanced”, “balanced_subsample”}, dict or list of dicts, default=None
17. ccp_alpha:non-negative float, default=0.0
18. max_samples:int or float, default=None

7.ADABOOST CLASSIFIER

sklearn.ensemble.**AdaBoostClassifier**

1. base_estimatorobject, default=None
2. n_estimatorsint, default=50
3. learning_ratefloat, default=1.0
4. algorithm{‘SAMME’, ‘SAMME.R’}, default=’SAMME.R’
5. random_stateint, RandomState instance or None, default=None

8.QUADRATIC DISCRIMINANT ANALYSIS

sklearn.qda.QDA

1. priors : array, optional, shape = [n_classes]

2. reg_param : float, optional

9. LIGHT GRADIENT BOOSTING MACHINE

lightgbm.LGBMClassifier

1. boosting_type (str, optional (default='gbdt'))
2. num_leaves (int, optional (default=31))
3. max_depth (int, optional (default=-1))
4. learning_rate (float, optional (default=0.1))
5. n_estimators (int, optional (default=100))
6. subsample_for_bin (int, optional (default=200000))
7. objective (str, callable or None, optional (default=None))
8. class_weight (dict, 'balanced' or None, optional (default=None))
9. min_split_gain (float, optional (default=0.))
10. min_child_weight (float, optional (default=1e-3))
11. min_child_samples (int, optional (default=20))
12. subsample (float, optional (default=1.))
13. subsample_freq (int, optional (default=0))
14. colsample_bytree (float, optional (default=1.))
15. reg_alpha (float, optional (default=0.))
16. reg_lambda (float, optional (default=0.))
17. random_state (int, RandomState object or None, optional (default=None))
18. n_jobs (int or None, optional (default=None))
19. importance_type (str, optional (default='split'))

10. LOGISTIC REGRESSION

sklearn.linear_model.LogisticRegression

1. penalty:{‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default=’l2’
2. dual:bool, default=False
3. tol:float, default=1e-4
4. C:float, default=1.0
5. fit_intercept:bool, default=True
6. intercept_scaling:float, default=1
7. class_weight:dict or ‘balanced’, default=None
8. random_state:int, RandomState instance, default=None
9. solver:{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=’lbfgs’
10. max_iter:int, default=100
11. multi_class:{‘auto’, ‘ovr’, ‘multinomial’}, default=’auto’
12. verbose:int, default=0
13. warm_start:bool, default=False
14. n_jobs:int, default=None
15. l1_ratio:float, default=None

11.MLP CLASSIFIER

`sklearn.neural_network.`**`MLPClassifier`**

1. hidden_layer_sizes:tuple, length = n_layers - 2, default=(100,)
2. activation:{‘identity’, ‘logistic’, ‘tanh’, ‘relu’}, default=’relu’
3. solver:{‘lbfgs’, ‘sgd’, ‘adam’}, default=’adam’
4. alpha:float, default=0.0001
5. batch_size:int, default=’auto’
6. learning_rate:{‘constant’, ‘invscaling’, ‘adaptive’}, default=’constant’
7. learning_rate_init:float, default=0.001
8. power_t:float, default=0.5
9. max_iter:int, default=200
10. shuffle:bool, default=True
11. random_state:int, RandomState instance, default=None
12. tol:float, default=1e-4
13. verbose:bool, default=False
14. warm_start:bool, default=False
15. momentum:float, default=0.9
16. nesterovs_momentum:bool, default=True
17. early_stopping:bool, default=False
18. validation_fraction:float, default=0.1
19. beta_1:float, default=0.9
20. beta_2:float, default=0.999
21. epsilon:float, default=1e-8

22. n_iter_no_change:int, default=10
23. max_fun:int, default=15000

12.XG BOOST

1. booster [default= gbtree]
2. verbosity [default=1]
3. validate_parameters [default to false, except for Python, R and CLI interface]
4. nthread [default to maximum number of threads available if not set]
5. disable_default_eval_metric [default= false]
6. num_feature [set automatically by XGBoost, no need to be set by user]

13.RADIAL SVM

`sklearn.svm.`**SVC**

1. Cfloat:default=1.0
2. Kernel:{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'
3. Degree:int, default=3
4. gamma:{'scale', 'auto'} or float, default='scale'
5. coef0:float, default=0.0
6. shrinking:bool, default=True
7. probability:bool, default=False
8. tol:float, default=1e-3
9. cache_size:float, default=200
10. class_weight:dict or 'balanced', default=None
11. verbose:bool, default=False
12. max_iter:int, default=-1
13. decision_function_shape:{'ovo', 'ovr'}, default='ovr'
14. break_ties:bool, default=False
15. random_state:int, RandomState instance or None, default=None

14.GRADIENT BOOSTIING CLASSIFIER

1. loss{'log_loss', 'deviance', 'exponential'}, default='log_loss'
2. learning_ratefloat, default=0.1
3. n_estimatorsint, default=100
4. subsamplefloat, default=1.0
5. criterion{'friedman_mse', 'squared_error', 'mse'}, default='friedman_mse'
6. min_samples_splitint or float, default=2
7. min_samples_leafint or float, default=1
8. min_weight_fraction_leaffloat, default=0.0
9. max_depthint, default=3
10. min_impurity_decreasefloat, default=0.0

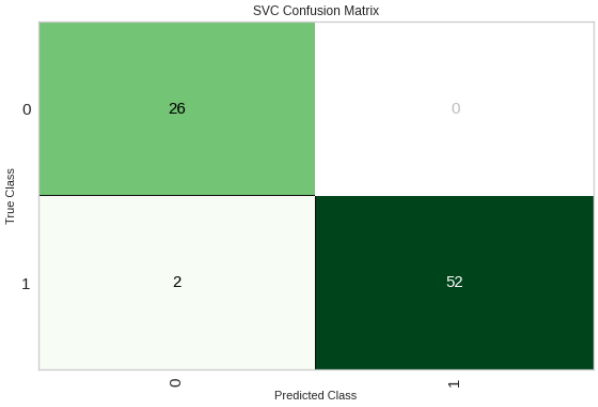
11. initestimator or ‘zero’, default=None
12. random_stateint, RandomState instance or None, default=None
13. max_features{‘auto’, ‘sqrt’, ‘log2’}, int or float, default=None
14. verboseint, default=0
15. max_leaf_nodesint, default=None
16. warm_startbool, default=False
17. validation_fractionfloat, default=0.1
18. n_iter_no_changeint, default=None
19. tolfloat, default=1e-4
20. ccp_alphanon-negative float, default=0.0

TABLE

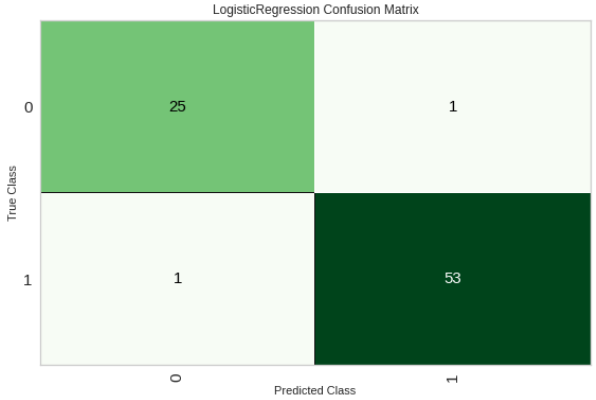
		Accuracy	Precision	Recall	F1 score	Sensitivity	specificity	AUC (ROC)	Error rate	TPR	FPR	Execution time
1.	Logistic regression	0.9969	1	0.99	0.9949	0.9836	0	1	0.0125	0.9836	0	0.017
2.	KNeighborsClassifier	0.9562	1	0.9307	0.9637	0.9508	1	0.9915	0.0375	0.9508	0	0.024
3.	Linear SVM	0.9875	0.9905	0.99	0.99	0.9672	1	0.9928	0.025	0.967	0	0.09
4.	RBF SVM	0.9838	1	0.9895	0.9946	0.9672	1	1	0.025	0.967	0	0.05
5.	Naive Bayes	0.9594	1	0.936	0.96652	0.918	1	1	0.0625	0.918	0	0.01
6	Decision Tree(gini index)	0.9688	0.9852	0.9655	0.9749	0.9836	1	0.9702	0.0125	0.983	0	0.011
7	Random Forest	0..9938	0.9905	1	0.9951	1	1	0.9996	0	1	0	0.919
8	AdaBoost	0.9875	0.9902	0.99	0.99	1	1	0.9992	0	1	0	0.101
9	MLPClassifier	0.9906	1	0.9842	0.9917	1	1	1	0	1	0	0.73
10	Quadratic Discriminant Analysis	0.3688	0	0	0	0	1	0	0.7625	0	0	0.01
11	Gradient Boosting Classifier	0.9812	0.99	0.9992	0.9549	1	1	0.9992	0	1	0	0.162
12	Light Gradient Boosting Machine	0.9875	0.9905	0.99	0.99	1	1	0.9996	0	1	0	0.108
13	XGBoost	0.9875	0.9902	0.99	0.99	1	1	0.9988	0	1	0	0.219
14.	Extra tree classifier	0.9906	1	0.985	0.9922	0.9643	1	1	0.024	0.964	0	1.164

CONFUSION MATRIX

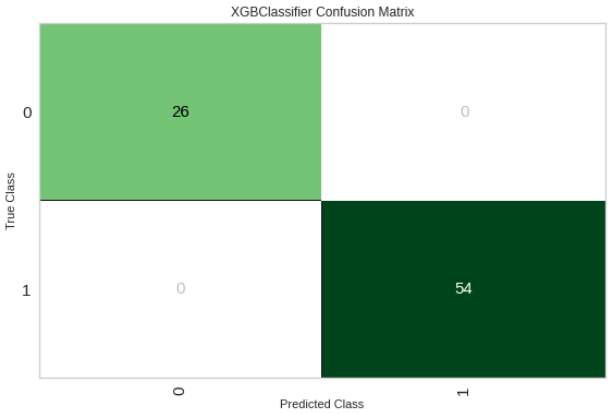
RBF SVM



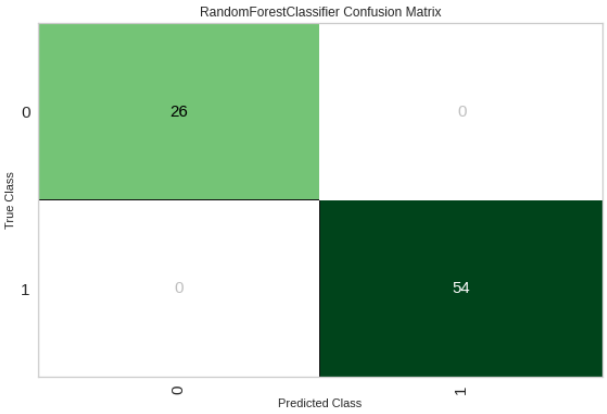
LOGISTIC REGRESSION



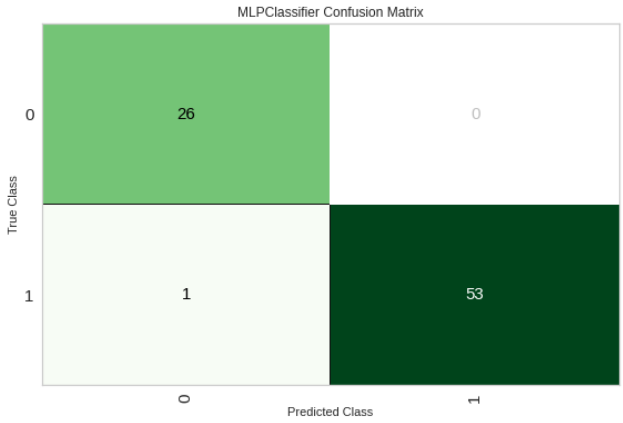
XGBOOST



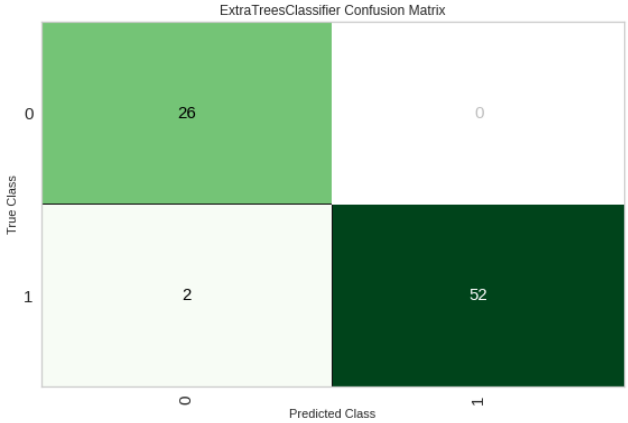
RANDOM FOREST



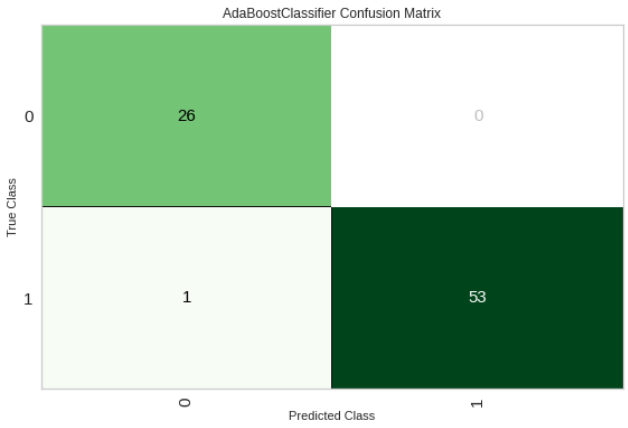
MLP CLASSIFIER



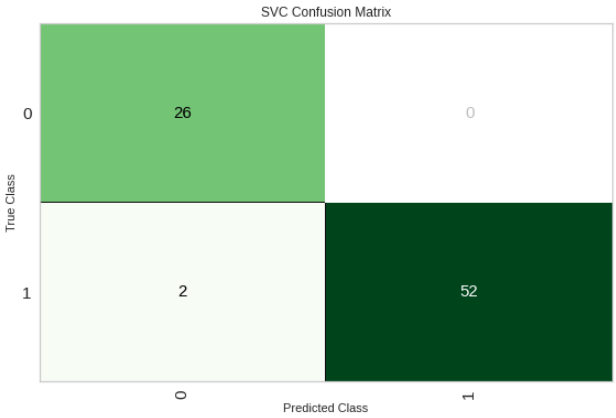
EXTRA TREES



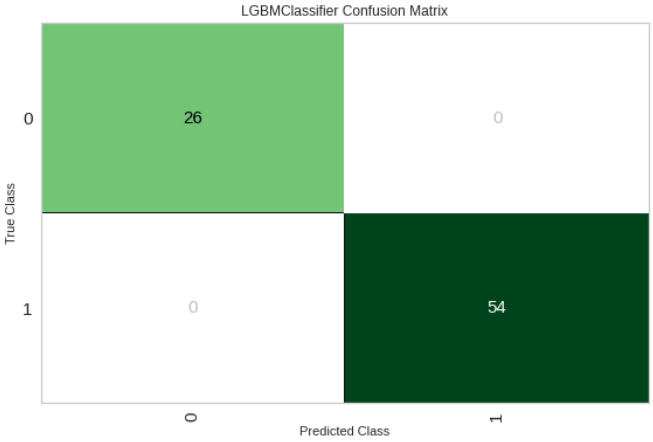
ADABOOST



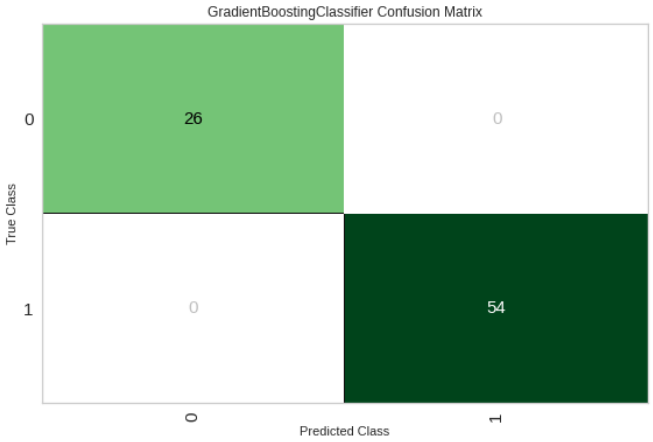
LINEAR SVM



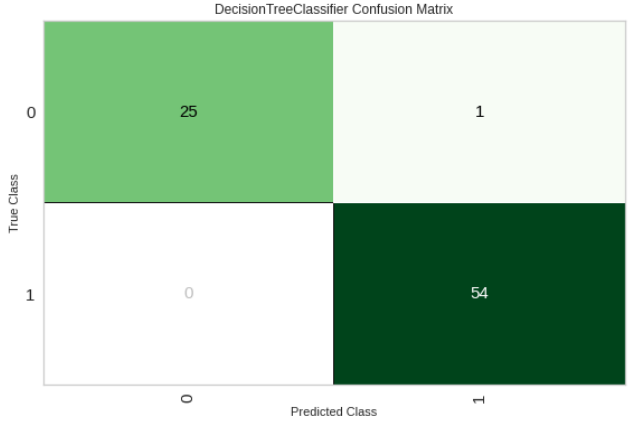
LIGHTGBM



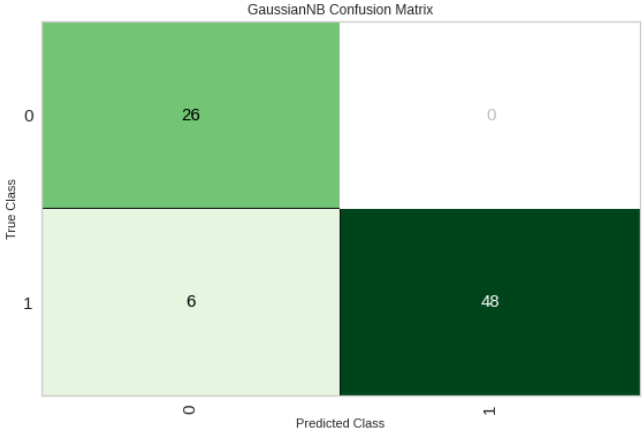
GRADIENT BOOSTING



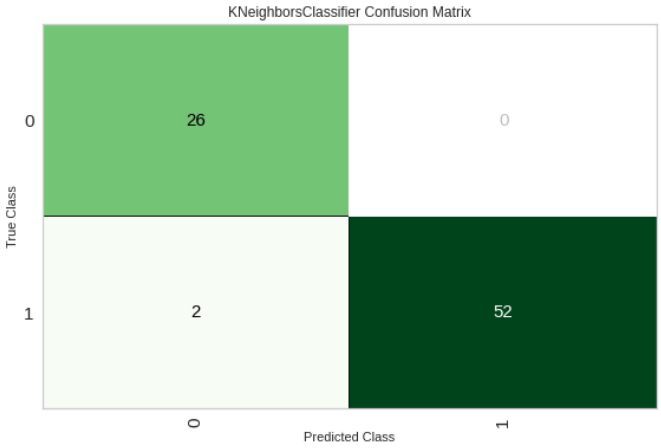
DECISION TREE



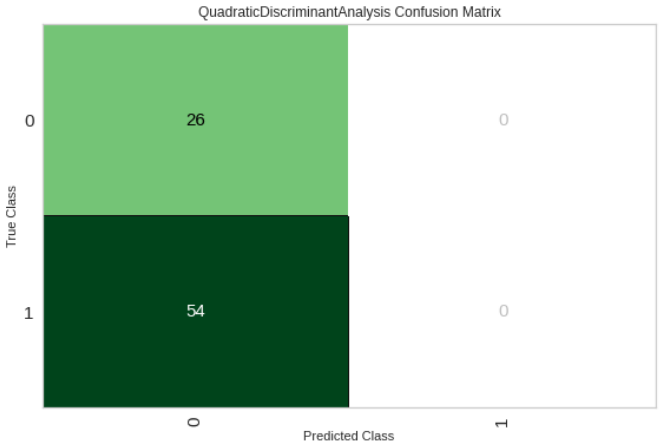
NAÏVE BYES



K NEIGHBOURS CLASSIFIER

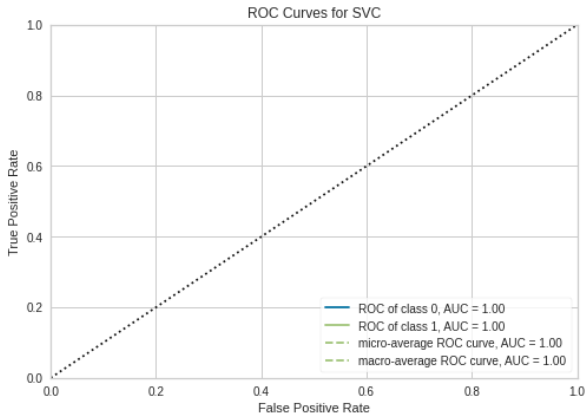


QUADRATIC DISCRIMINANT ANALYSIS

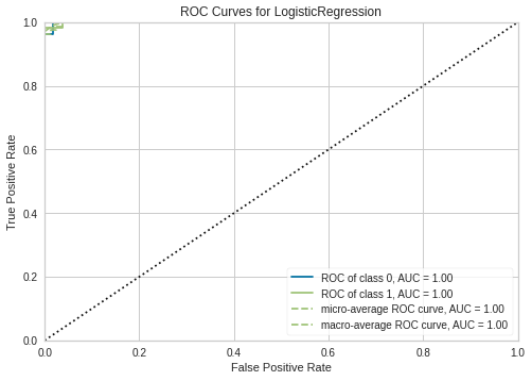


ROC CURVES

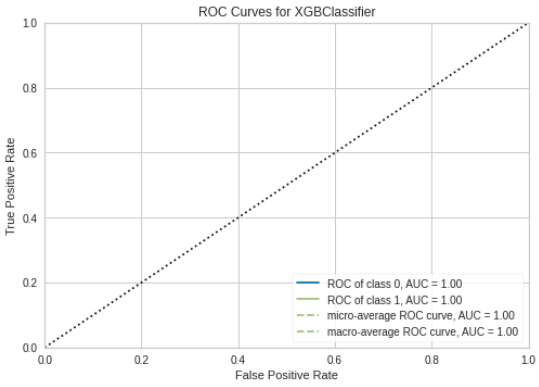
RBF SVM



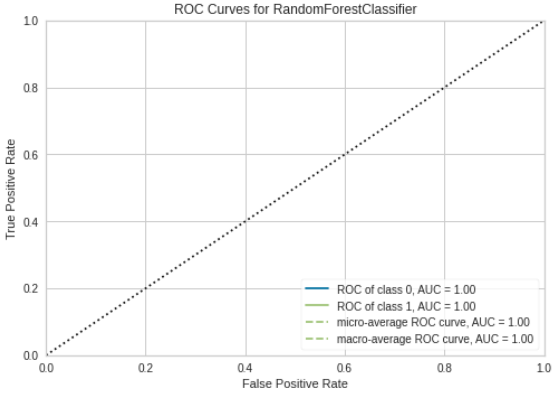
LOGISTIC REGRESSION



XGBOOST

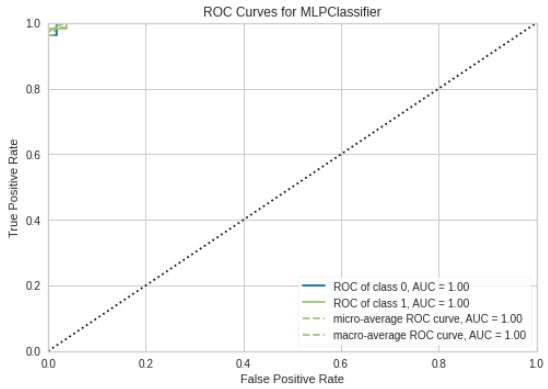


RANDOM FOREST

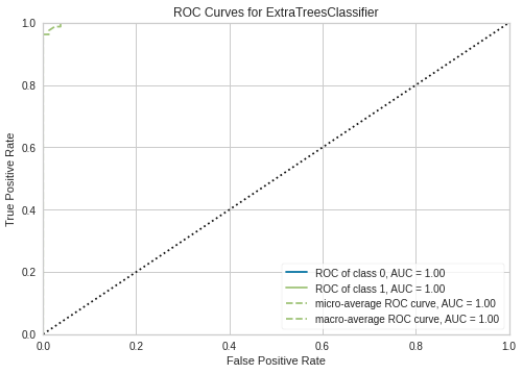


MLP CLASSIFIER

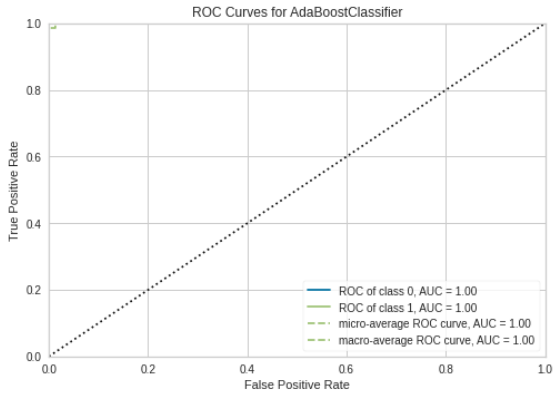
\



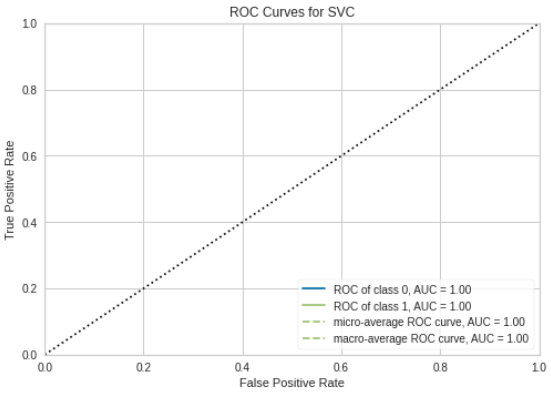
EXTRA TREES



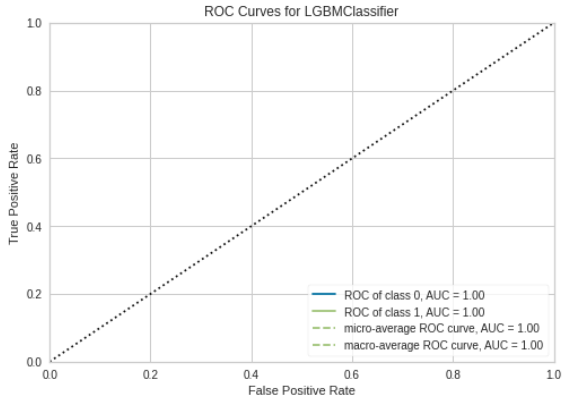
ADABOOST



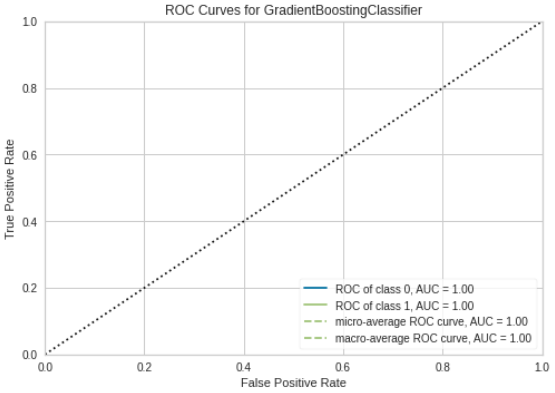
LINEAR SVM



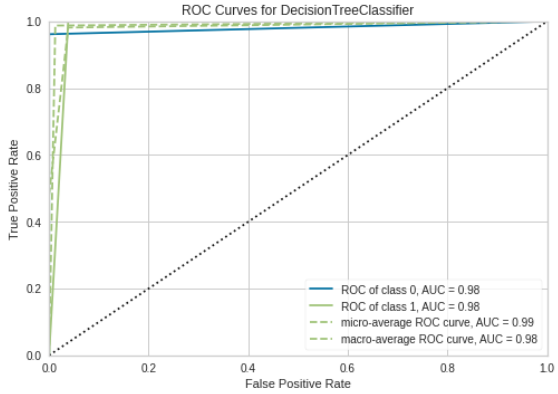
LIGHTGBM



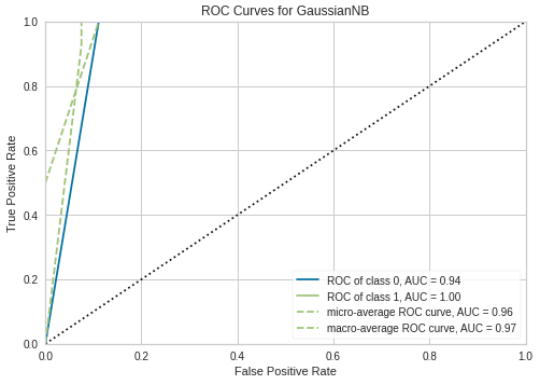
GRADIENT BOOSTING



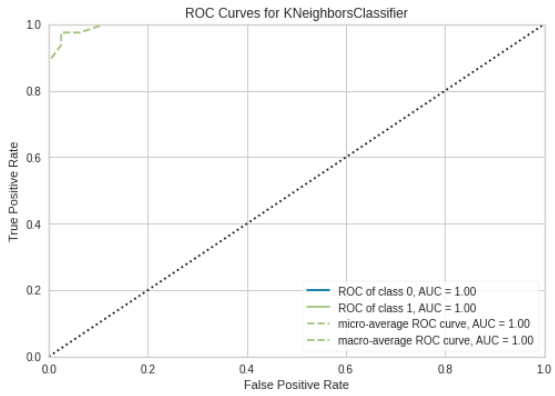
DECISION TREE



NAÏVE BYES



K NEIGHBOURS CLASSIFIER



DISCUSSION SECTION

After training all the above-mentioned algorithms on the Dataset we can conclude that the best-performing algorithm is **LOGISTIC REGRESSION.** This is because out of all algorithms, Logistic Regression has the highest accuracy from the test set with a score of 99.69 %, which is very good. Also, the False Positive Rate is 0 which is very crucial as we do not want to classify patients not having kidney disease as positive. That would lead to more complications in the health of the patient. Logistic Regression performs well in this dataset as the number of quantitative features is more and the categorical features are very easy to be encoded into integers. Also, the recall and the precision scores cross 99 % which makes this model very ideal for Analysis.In the models, a lot of decision tree based algorithms are used that depend on string value. But the string value has less unique values so Logistic Regression performs well here.

CONCLUSION

After acquiring the dataset, we performed cleaning on the dataset with some steps mentioned above. The dataset is thoroughly observed and necessary columns are dropped which won't be part of the ML analysis. After training the dataset on different models, we conclude that the best model for this particular dataset is Logistic Regression. It performs very well on the dataset with an Accuracy of 99.69 %, and another plus point is that the number of false positives is 0 from this model. The ML Model also has a very high Recall score of 0.99 and a very high F1 score of 0.9949. By using this model, we can accurately predict if a person has Chronic Kidney problems and a lot of lives can be saved in the future.

REFERENCES

1. Webster AC, Nagler EV, Morton RL, Masson P. Chronic kidney disease. Lancet. 2016;6736(16):1–15.
2. Serpen AA. Diagnosis rule extraction from patient data for chronic kidney disease using machine learning. Int J Biomed Clin Eng. 2016;5(2):64–72. https://doi.org/10.4018/IJBCE.2016070105.
3. Tekale S, Shingavi P, Wandhekar S. Prediction of chronic kidney disease using machine learning algorithm. Ijarcce. 2018;7(10):92–6. https://doi.org/10.17148/IJARCCE.2018.71021.
4. Ponum M, Hasan O, Khan S. EasyDetectDisease: an android app for early symptom detection and prevention of childhood infectious diseases. Interact J Med Res. 2019;8(2):e12664. https://doi.org/10.2196/12664.
5. Hill NR, Fatoba ST, Oke JL, Hirst JA, O'Callaghan CA, Lasserson DS et. al. (2016) Global prevalence of chronic kidney disease—a systematic review and meta-analysis. PLoS One 11:e0158765, 7, DOI: https://doi.org/10.1371/journal.pone.0158765
6. Ramya S, Radha N. Diagnosis of chronic kidney disease using machine learning algorithms. Int J Innovative Res Comput Commun Eng. 2016;4(1):812–20.
7. Xiao J, et al. Comparison and development of machine learning tools in the prediction of chronic kidney disease progression. J Transl Med. 2019;17(1):1–13.
8. E. H. A. Rady and A. S. Anwar, "Prediction of kidney disease stages using data mining algorithms," Inform Med. Unlocked, vol. 15, no. April, p. 100178, 2019.
9. Teo BW, Xu H, Wang D, Li J, Sinha AK, Shuter B, et al. GFR estimating equations in a multiethnic asian population. Am J Kidney Dis. 2011;58(1):56–63. https://doi.org/10.1053/j.ajkd.2011.02.393.
10. Stevens LA, Claybon MA, Schmid CH, Chen J, Horio M, Imai E, et al. Evaluation of the chronic kidney disease epidemiology collaboration equation for estimating the glomerular filtration rate in multiple ethnicities. Kidney Int. 2011;79(5):555–62. https://doi.org/10.1038/ki.2010.462.