CHRONIC KIDNEY DISEASE PREDICTION USING MACHINE LEARNING

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

EEE300 - MINI PROJECT

Submitted by

Muttumula Sai Vardhan Reddy
(Reg. No.: 123005092, EEE)
Edamadaka Jahnavi
(Reg. No.: 123005032, EEE)

December 2022

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING

THANJAVUR - 613401

Bonafide Certificate

This is to certify that the report titled "**Chronic Kidney Disease Prediction Using Machine Learning**" submitted as a requirement for the course, EEE300: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by

Mr. Muttumula Sai Vardhan Reddy (Reg. No. 123005092, EEE)

Ms. Edamadaka Jahnavi (Reg. No. 123005032, EEE)

during the academic year 2022-23, in the School of ELECTRICAL AND ELECTRONICS ENGINEERING, under my supervision.

Signature of Project Supervisor :

Name with Affiliation : Dr. Venkatesh T, Asst Professor – III, SEEE

Date : 06/12/22

Mini Project *Viva voce* held on _____

Examiner 1 **Examiner 2**

Acknowledgements

First of all, I would like to thank *God Almighty* for his endless blessings.

I would like to express my sincere gratitude to **Dr S. Vaidya Subramaniam,** *Vice-Chancellor* for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and I would like to thank **Dr K. Thenmozhi**, *Dean, School of Electrical and Electronics Engineering* and **R. Chandramouli**, *Registrar* for their overwhelming support provided during my course span in *SASTRA Deemed University*.

I am extremely grateful to Lt. **Dr K. vijayrekha**, *Associate Dean, School of Electrical and Electronics Engineering* for her constant support, motivation and academic help extended for the past three years of my life in *School of Electrical and Electronics Engineering*.

I would specially thank and express my gratitude to **Dr T. Venkatesh**, *Assistant Professor-III, School of Electrical and Electronics Engineering* for providing me an opportunity to do this project and for his guidance and support to successfully complete the project.

I also thank all the *Teaching and Non-teaching faculty,* and all other people who have directly or indirectly help me through their support, encouragement and all other assistance extended for completion of my project and for successful completion of all courses during my academic life at *SASTRA Deemed University*.

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under Curve
CFS	Correlation based Feature Selection
CHAID	Chi-square Automatic Interaction Detection
CKD	Chronic Kidney disease
FN	False Negative
FP	False Positive
FS	Feature Selection
KNN	K-Nearest Neighbor
LASSO	Least Absolute Shrinkage and Selection Operator
LASSO FS	Least Absolute Shrinkage and Selection Operator Feature Selection
LR	Logistic Regression
LSVM	Linear Support Vector Machine
LSVM L1	Linear Support Vector Machine with penalty L1
LSVM L2	Linear Support Vector Machine with penalty L2
ML	Machine Learning
RF	Random Forest
ROC	Receiver Operating a Characteristic
SMOTE	Synthetic Minority class Oversampling Technique
TN	True Negative
TP	True Positive
UCI	UC Irvine repository

Abstract

Illness detection has become a crucial component in the medical industry because treating a disease can only be accomplished once it has been identified. One of the primary diseases that needs to be caught early on is chronic kidney disease (CKD). In today's world, machine learning (ML) is essential for disease identification. Therefore, the major goal of this work is to identify the top machine learning model for CKD prediction. A dataset from the UCI repository was gathered for this viewpoint. We used the K-Nearest Neighbor (KNN), Logistic Regression (LR), Linear Support Vector Machine (LSVM) with penalties L1 and L2, Random Forest (RF), CHAID, C 4.5, and Artificial Neural Network (ANN) classifier methods, as well as ML for prediction. To pick the features, we use feature selection (FS) techniques as full feature selection, correlation-based feature selection (CFS), wrapper FS, and LASSO FS. The target class is balanced using a data balancing technique on a dataset for the SMOTE. Accuracy, precision, recall, error, F-measure, AUC, and GINI index are used to evaluate the performance of classifiers.

The findings for each feature selection approach and all classifier methods are computed in this study. The highest-performing algorithms with the best feature selection techniques are chosen based on the results, and their performances are computed in each instance using SMOTE. We deduced from the data that LR, LSVM L1, LSVM L2, CHAID, C 4.5, and RF outperform Full Features and LASSO Feature Selection in terms of accuracy measure and other evaluation metrics. Random Forest performs best with an accuracy of 98.49% after applying SMOTE with LASSO Feature Selection, whereas LSVM with penalty L2 has an accuracy of 97.73%.

KEY WORDS: CKD, prediction, ML Classifier algorithms, FS methods, SMOTE.

Table of Contents

CHAPTER 1

SUMMARY OF THE BASE PAPER

1.1. Problem Statement

CKD cannot be easily recognised unless the kidney is severely damaged. We are unable to routinely assess our disease status in this high-risk population.

We are unable to anticipate the sickness using current techniques. We are able to accurately predict the disease using ML classifier techniques. Data set collection & Pre-processing

1.1.1. Data Set

The UC Irvine Machine Learning Repository was offering the CKD dataset, which was obtained for this study from the UCI website. 400 instances and 25 attributes make up this dataset. One of those 25 qualities is a goal attribute, while the other 24 are feature attributes. Both CKD and Not CKD are valid values for the goal attribute. Table 1.2 lists the 24 feature qualities and their descriptions.

1.1.2. KNN Imputation

The first and most important thing we need to do with the dataset is find any missing values and replace them. Data cleansing is what this is. Predicting the missing values with an ML model is one of the finest methods for cleaning data. In this study, we predict missing values using the KNN method.

1.1.3. Label Encoder

Label encoding is the process of turning every label into a numeric form that will then be transformed into a machine-readable form. It is among the greatest methods for preparing data. In this paper, all the labels are converted into float or int types.

1.1.4. Train- Test split

Train-Test Split is the most frequently used approach for preprocessing. In this process, the data is divided into two parts, one for training the model and the other for testing. It is mainly used for estimating the performance of ML classifiers. In this paper, we consider the train-test split ratio to be 50:50.

1.2.Proposed Methodology

1.2.1. Feature Selection Methods

1.2.1.1.Full Feature Selection

In this case, we will consider all features as input for ML classifiers for prediction.

1.2.1.2.Correlation based FS (CFS)

CFS is one of the filter methods to select the features based on the correlation

between them. In this process, it will eliminate the highly correlated feature with the other feature. It selects only the remaining features after eliminating the highly correlated ones.

1.2.1.3.Wrapper FS

The feature selection in Wrapper FS is based on an accurate ML classifier. For creating subgroups, it employs a greedy strategy. Both the forward selection approach and the backward elimination method can be used to create subsets. In this paper, "forward selection" is used. Forward selection starts with a null subset and incrementally adds features.

1.2.1.4.LASSO FS

LASSO (least absolute shrinkage and selection operator) (least absolute shrinkage and selection operator) FS is a procedure that is integrated in FS. In the embedded approach, decision trees are used to pick features. This method involves the regularisation and feature selection phases. The coefficients of insignificant features are reduced to zero during regularisation, and the zero-coefficient features are disregarded during feature selection.

1.2.2. Machine Learning Classifiers

1.2.2.1.LR

Logistic regression is one of supervised ML classifiers. It's a mathematical model. It predicts the probability of achieving the target value. It categorizes the target attribute as either successful or unsuccessful. It returns 1 when successful, and 0 when unsuccessful.

1.2.2.2.KNN

KNN is a simplest supervised algorithm. For both classification and regression problems KNN is applicable. It is, however, primarily used for solving classification problems. The distance between two already labelled data sets is calculated. The distance aids in locating the new data nearest neighbor. Distance is calculated using the Euclidian method.

1.2.2.3.LSVM

LSVM is a latest, uniquely fast machine learning algorithm that uses a simple iterative approach to solve multiclass classification problems for large datasets. The SVM model is built in the dataset's linear CPU time. There are two penalty cases, L1 and L2.

1.2.2.4.CHAID

This decision tree technique is chi-square automatic interaction detection (CHAID). It's used to figure out how variables are related to one another. CHAID can find the outcome using nominal, ordinal, and continuous data.

1.2.2.5.C 4.5

C4.5 is a decision tree in the sense that from input it generates the decision tree. The tree will have the specified count of branches. The structure of tree is used to find out the relationship between potential outcomes and features. This process is iterative and continues until no further splits are found.

1.2.2.6.ANN

ANN is a type of AI. It falls under the category of supervised ML. It has the same structure as the human brain. It can solve problems that human or statistical standards have failed to solve.

1.2.2.7.RF

Random forests, also called as random decision forests, is an ensemble learning classifier for regression and classification that works by constructing a big number of decision trees during training. For classification tasks, the random forest output is the class chosen by the majority of trees. The mean prediction value of the different trees is returned for regression tasks. RF outperform decision trees in general.

1.2.3. SMOTE

Data balancing is the major problem one is facing while using dataset for prediction. There are 2 methods for data balancing, oversampling and under sampling. In this case we use oversampling. For oversampling the minority class, the SMOTE is used. It is also referred to as a data balancer. It accepts the entire dataset as input but works only on the minority class. It raises the proportion of minorities in the population. This methodology expands the features available for target class.

1.2.4. Performance Evaluation Metrics

1.2.4.1.Confusion Matrix for prediction of CKD

	CKD (1)	Not CKD (0)
CKD (1)	**TP**	**FN**
Not CKD (0)	**FP**	**TN**

Table. 1.1. Confusion Matrix for prediction of CKD

TP: It indicates that output was CKD and the prediction was classified correctly.

TN: It indicates that output was not CKD and the prediction was classified correctly.

FP: It indicates that output was CKD and the prediction was classified incorrectly.

FN: It indicates that output was not CKD and the prediction was classified incorrectly.

1.2.4.2.Accuracy:

It shows the right rate of prediction findings.

$$Accuracy = ((TP+TN)/(TP+TN+FP+FN)) *100$$

1.2.4.3.Precision:

It is the proportion of related instances in the total number of retrieved instances. It is a predicted positive value.

$$Precision= (TP/(TP+FP)) *100$$

1.2.4.4.Recall:

It is the proportion of related instances among all retrieved instances.

$$Recall= (TP/(TP+FN)) *100$$

1.2.4.5.F- Measure:

It is referred to as the F Score. It is used to determine the test accuracy.

$$F\text{- Measure}= 2* ((precision*recall)/ (precision + recall))$$

1.2.4.6.AUC:

ROC curve, a graph that compares the true positive rate to the false positive rate at various classification thresholds. The whole area under the ROC curve is referred to as the AUC.

1.2.4.7.GINI Index:

It is used to assess the disparity in attribute value distributions. It calculates the impurity of a specific attribute as probability.

$$GINI= 2* AUC-1$$

1.2.4.8.Error:

The rate of incorrect predicted results is shown by the classification error.

$$Error = 1- accuracy$$

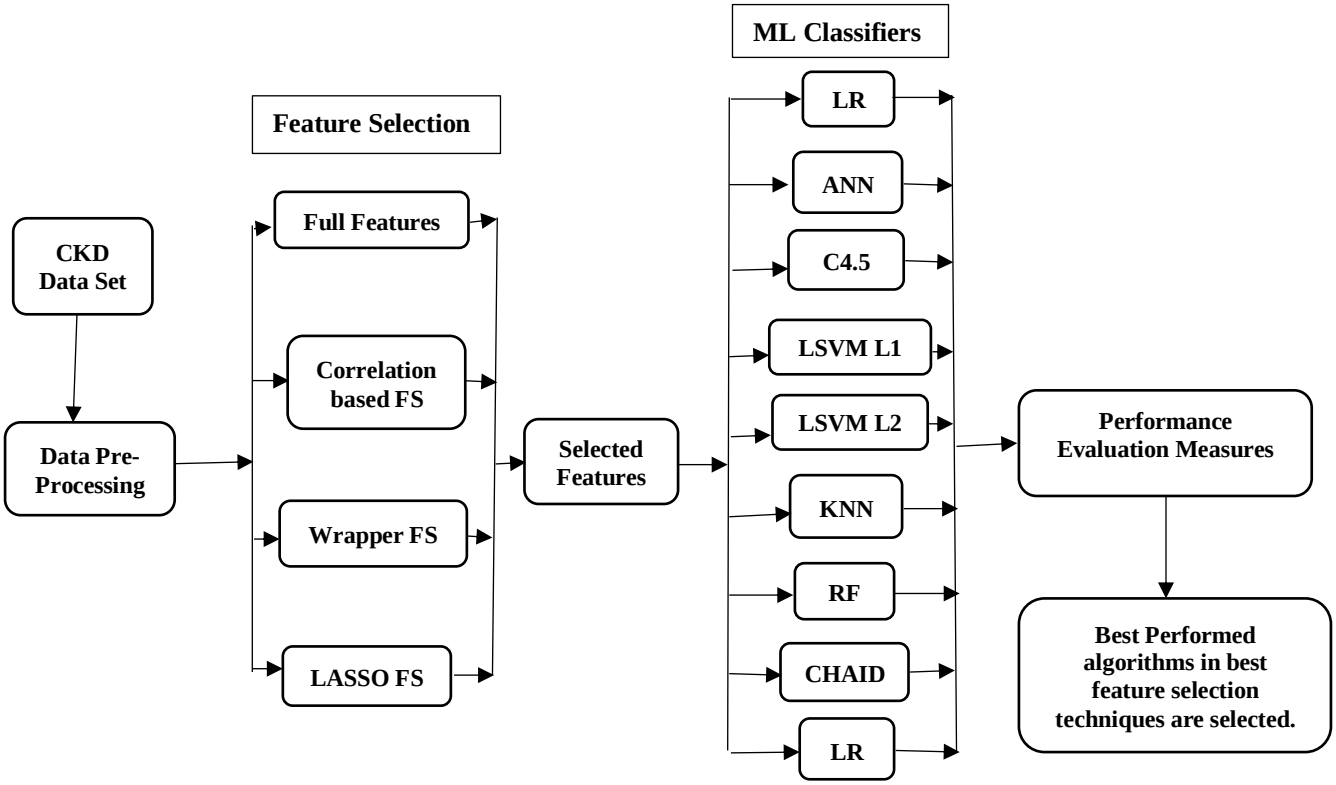


Fig. 1.1. Work Flow model for CKD Prediction before applying SMOTE

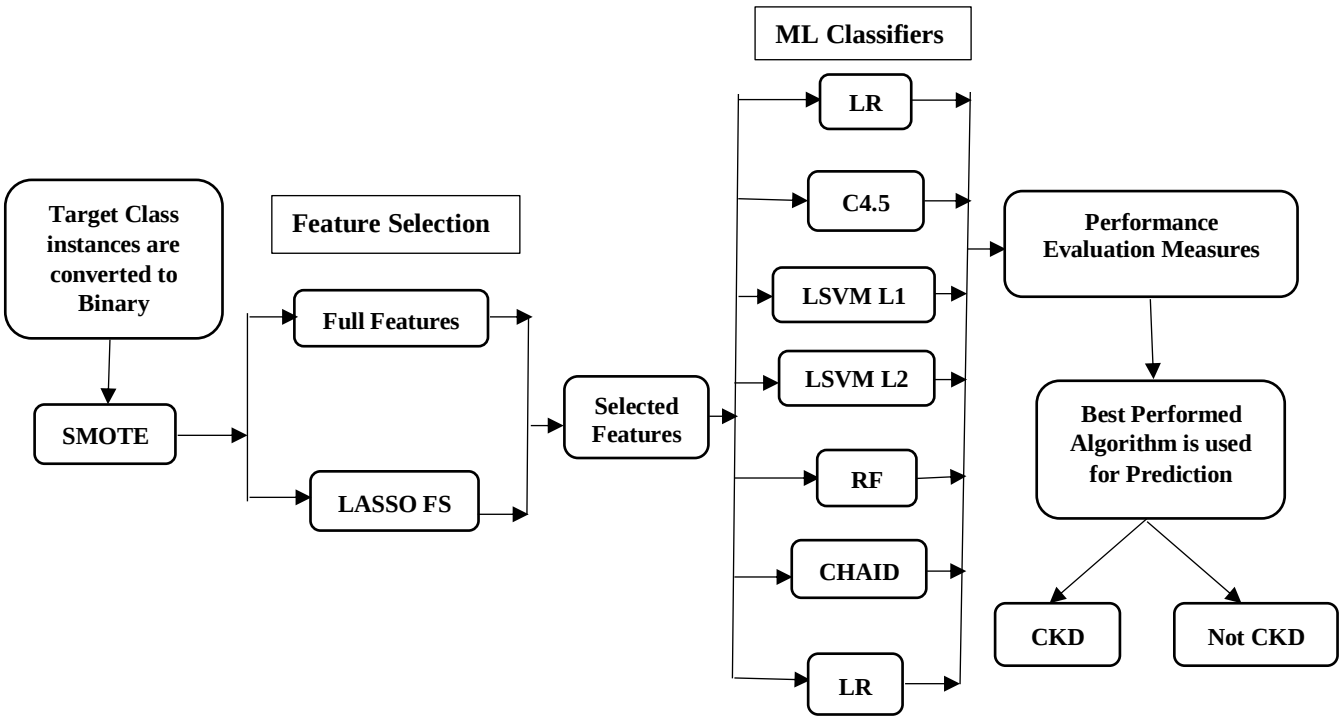


Fig. 1.2. Work Flow model for CKD Prediction after applying SMOTE

S. No.	Attribute Name	Description
1	Age	Age of patient
2	Bp	Blood pressure of patient
3	Sg	Urine specific gravity of patient
4	Al	Albumin level of patient
5	Su	Sugar level of patient
6	Rbc	Red blood cells normality in patient
7	Pc	Pus cell normality in patient
8	Pcc	Pus cell clumps existence in patient
9	Ba	Bacteria existence in patient
10	Bgr	Blood glucose random value of patient
11	Bu	Blood urea value of patient
12	Sc	Serum creatinine value of patient
13	Sod	Sodium level of patient
14	Pot	Potassium level of patient
15	Hemo	Hemoglobin level of patient
16	Pcv	Packed cell volume % of RBCs in circulating blood of patient
17	Wc	White blood cells count in patient
18	Rc	Red blood cells count in patient
19	Htn	Hypertension Yes or No for patient
20	Dm	Diabetes mellitus Yes or No for patient
21	Cad	Coronary artery disease Yes or No for patient
22	Appet	Appetite Yes or No for patient
23	Pe	Pedal edema Yes or No for patient
24	Ane	Anemia Yes or No for patient
25	Class	Target Variable (CKD or Not CKD)

Table. 1.2. Description of Attributes in the Dataset.

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

2.1. Literature Review

G. Parthiban and K. R. A. Padmanaban et al. [9] identified Kidney disorder in its early stages by using Nave Bayes and Decision tree techniques on the selected features. However, the neural network prediction system's accuracy could be improved.

K. D. M. Perera, W. Gunarathne and K. A. D. C. P Kahandawaarachchi et al. [8] conducted a performance evaluation on Machine Learning Classification Techniques for classification of disease and Forecasting via Data Analytics for CKD and discovered that the Multiclass Decision Forest algorithm has the best accuracy of 98.1 percent.

The effects of using clinical features to classify patients with chronic kidney disease using the SVM algorithm were investigated by S. Shamiluulu, Y. Amirgaliyev and A. Serek et al. [7]. The implemented classifier is capable of achieving an overall performance of 94.602 percent.

M. Almasoud and T. E. Ward et al. [6] used 10-fold cross-validation to train and test logistic regression, support vector machines, random forest, and gradient boosting algorithms in this study. The gradient boosting algorithm outperformed other algorithms in terms of F1-measure (99.1 percent), sensitivity (98.8 percent), and specificity (98.8 percent) (99.3 percent).

L. Lessa, A. Peixoto, L. Kilvia De Almeida, R. Gomes, and J. Celestino, among others, [3] They devised a method for detecting kidney failure early on, increasing the chances of successful treatment for these patients. Several Machine Learning techniques (DT, SVM, and RF) were employed for this purpose. However, the accuracy of the prediction was low and could be improved further.

S. Elavarthy, T. Kiran, S. Shankar, S. Verma, P. Ghuli, and colleagues [4] Logistic Regression, Random Forest Tree, K-Nearest Neighbor, and Neural Network are four ML techniques that they used. To determine the correct classifier for CKD diagnosis, they conducted a systematic study of the outcomes of all three classifiers. However, certain noisy values are missing from the data collection.

2.2. Merits

ML classifiers are commonly used for prediction. We used several FS methods to improve accuracy. Not every algorithm performs well with a well-balanced data set. As a result, we used SMOTE to balance the dataset.

2.3. Demerits

In general, ANN and KNN are better performing classifiers, but their performance on our data set was poor.

2.4. Results

In case of Full feature selection, for LR the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97, 100, 94.96, 0.96, 97.47, 0.94 and 3.0 respectively. for KNN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 66.5, 66.25, 89.07, 0.44, 61.20, 0.22 and 33.5 respectively. for LSVM penalty L1 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.0, 99.1, 95.7, 0.96, 97.2, 0.94 and 3.0 respectively. for LSVM penalty L2 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.5,100, 95.8, 0.97, 97.9, 0.95 and 2.5 respectively. for RF the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 99.5, 99.1, 100, 0.99, 99.3, 0.98 and 0.5

respectively. for CHAID the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.2, 97.5, 96.3, 0.96, 96.1, 0.92 and 3.78 respectively. for C 4.5 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.9, 97.5, 97.5, 0.97, 96.7, 0.93 and 3.03 respectively. for ANN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 83, 94.3, 61.7, 0.74, 79.6, 0.59 and 17.0 respectively.

In case of CFS, for LR the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 94.5, 97.3,93.27, 0.93, 94.7, 0.89 and 5.5 respectively. for KNN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 64.5, 68.4, 74.7, 0.52, 62.08, 0.24 and 35.5 respectively. for LSVM penalty L1 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 95.5, 97.4, 94.9, 0.94, 95.6, 0.91 and 4.5 respectively. for LSVM penalty L2 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 91.5, 97.2, 88.2, 0.90, 92.2, 0.84 and 8.5 respectively. for RF the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97, 95.1, 100, 0.96, 96.29, 0.92 and 3.0 respectively. for CHAID the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 95.45, 95.23, 97.56, 0.96, 94.7, 0.89 and 4.54 respectively. for C 4.5 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.9, 98.75, 96.34, 0.97, 97.1, 0.94 and 3.0 respectively. for ANN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 67.5, 60.0, 59.25, 0.59, 66.1, 032 and 32.5 respectively.

In case of Wrapper FS, for LR the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 90.5, 92.3, 91.5, 0.88, 90.24, 0.80 and 9.5 respectively. for KNN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 64.0, 68.2, 73.9, 0.52, 61.6, 0.23 and 36 respectively. for LSVM penalty L1 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 90.0, 93.04, 89.9, 0.87, 90.0, 0.8 and 10 respectively. for LSVM penalty L2 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 90.5, 92.3, 91.5, 0.8, 90.2, 0.8 and 9.5 respectively. for RF the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 91.5, 91.8, 94.1, 0.89, 90.8, 0.81 and 8.5 respectively. for CHAID the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 94.6, 94.1, 97.5, 0.95, 93.7, 0.87 and 5.3 respectively. for C 4.5 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.2, 96.3, 97.5, 0.96, 95.7, 0.91 and 3.7 respectively. for ANN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 79.5, 66.94, 97.5, 0.88, 90.88, 0.81 and 20.5 respectively.

In case of LASSO FS, for LR the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.5, 100, 95.7, 0.97, 97.8, 0.95 and 2.5 respectively. for KNN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 67.0, 70.8, 75.6, 0.57, 64.97, 0.29 and 33 respectively. for LSVM penalty L1 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.5, 99.1, 96.6, 0.96, 97.7, 0.95 and 2.5 respectively. for LSVM penalty L2 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.5, 99.13, 96.63, 0.96, 97.7, 0.95 and 2.5 respectively. for RF the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 98.0, 98.3, 98.3, 0.97, 97.9, 0.95 and 2 respectively. for CHAID the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 98.4, 98.7, 98.7, 0.98, 98.3, 0.96 and 1.5 respectively. for C 4.5 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 93.1, 95.0, 93.9, 0.94, 92.9, 0.85 and 6.8 respectively. for ANN the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 87.5, 87.8, 80.2, 0.83, 86.3, 0.72 and 12.5 respectively.

In case of Full features with SMOTE, for LR the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.5, 100, 94.1, 0.95, 97.0, 0.94 and 3.5 respectively. for LSVM penalty L1 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 95.5, 98.3, 98.3, 0.97, 85.8, 0.91 and 4.5 respectively. for LSVM penalty L2 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.5, 99.1, 94.9, 0.95, 96.8, 0.93 and 3.5 respectively. for RF the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.5, 98.3, 97.4, 0.96, 97.5, 9.95 and 2.5 respectively. for CHAID the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 95.4, 98.7, 93.9, 0.96, 95.9, 0.9 and 4.5 respectively. for C 4.5 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 62.1, 62.1, 100, 0.76, 50, 0 and 37.8 respectively.

In case of LASSO FS with SMOTE, for LR the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.9, 98.75, 96.3, 0.97, 97.1, 0.91 and 3.03 respectively. for LSVM penalty L1 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 96.9, 98.7, 96.3, 0.97, 97.1, 0.91 and 3.03 respectively. for LSVM penalty L2 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 97.7, 98.7, 97.5, 0.98, 97.7, 0.93 and 2.27 respectively. for RF the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 98.4, 97.6, 100, 0.98, 98, 0.95 and 1.51respectively. for CHAID the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 95.45, 96.34, 96.34, 0.96, 95.17, 0.91 and 4.5 respectively. for C 4.5 the accuracy, precision, recall, F-measure, AUC, GINI Index and Error are 62.12, 62.12, 100, 0.76, 50, 0 and 37.8 respectively.

In the Table. 2.1. the accuracies of all classifiers in all cases are compared and for each case the best accurate algorithm is marked.

Classifiers / FS	**Full Features**	**CFS**	**Wrapper**	**LASSO**	**Full Features with SMOTE**	**LASSO with SMOTE**
LR	97.0%	94.5%	90.5%	97.5%	96.5%	96.97%
LSVM L1	97.0%	95.5%	90.0%	97.5%	95.5%	96.97%
LSVM L2	97.5%	91.5%	90.5%	97.5%	96.5%	97.73%
CHAID	96.22%	95.45%	94.67%	**98.49%**	95.46%	95.46%
C 4.5	96.97%	96.97%	**96.22%**	93.19%	62.13%	62.13%
RF	**99.5%**	**97.0%**	91.5%	98.0%	**97.5%**	**98.49%**
KNN	66.5%	64.5%	64.0%	67.0%	----	----
ANN	83.0%	67.5%	79.5%	87.5%	----	----
Best Accuracy	**RF**	**RF**	**C 4.5**	**CHAID**	**RF**	**RF**

Table. 2.1. Accuracy Table of Classifiers in case of different FS

CHAPTER 3

SOURCE CODE

```
# Data Pre-Processing


import pandas as pd
data = pd.read_csv(&quot;DataSet.csv&quot;)
data.head()
data.info()
dec = data[&#39;DECID&#39;]
list1 = []
for i in data:
# data[i].dtype == &#39;object&#39;:
list1.append(i)
list1
# Label Encoder
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
for i in list1:
data[i] = labelencoder.fit_transform(data[i])
data
type(data)
list2 = [x for x in data]
data.info()
# KNN Imputer
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=20,metric = &#39;nan_euclidean&#39;) #
DataNoNull= pd.DataFrame(imputer.fit_transform(data))
Data = pd.DataFrame()
for i in range(len(list2)):
Data[list2[i]] = DataNoNull[i]
Data[list2[i]] = Data[list2[i]].astype(data[list2[i]].dtype)
Data.head()
Data.info()
```

```
x_all_features = Data.iloc[:,0:24]
y_all_features = Data[&#39;DECID&#39;]
# Train-Test Split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split( x_all_features, y_all_features, test_size =0.50)
display(x_train.shape,y_train.shape,x_test.shape,y_test.shape)
# Heat Map for Full Features
import seaborn as sns
sns.set(rc={&#39;figure.figsize&#39;:(24,14)})
sns.heatmap(data.corr().annot = True)
# LR in case of Full Features
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, accuracy_score, recall_score,confusion_matrix,
f1_score, roc_auc_score
import sys
lr = LogisticRegression(solver = &#39;lbfgs&#39;, max_iter = sys.maxsize ) #liblong
lr.fit(x_train, y_train)
y_predict = lr.predict(x_test)
accuracy_lr = accuracy_score(y_test,y_predict)
precison_lr = precision_score(y_test, y_predict,pos_label= 0 )
recall_lr = recall_score(y_test, y_predict,pos_label=0)
fmes_lr = f1_score(y_test,y_predict,zero_division =1)
accuracy_lr*=100
precison_lr*=100
recall_lr*=100
error_lr = 100 - accuracy_lr
print(&quot;Accuracy :&quot;,accuracy_lr)
print(&quot;Precision&quot;,precison_lr)
print(&quot;Recall :&quot;,recall_lr)
print(&quot;Error : &quot;,error_lr)
print(&quot;F1 Score : &quot;,fmes_lr)
```

```
auc_lr = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_lr)
gini_lr = 2*(auc_lr/100) - 1
print(&quot;GINI : &quot;,gini_lr)
print(&quot;Confusion matrix &quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
# KNN in case of Full features
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(x_train, y_train)
y_predict = knn.predict(x_test)
accuracy_knn = accuracy_score(y_test,y_predict)
precison_knn = precision_score(y_test, y_predict,pos_label= 0 )
recall_knn = recall_score(y_test, y_predict,pos_label= 0)
fmes_knn = f1_score(y_test,y_predict,zero_division =1)
accuracy_knn*= 100
precison_knn*= 100
recall_knn *= 100
error_knn = 100 - accuracy_knn
print(&quot;Accuracy :&quot;,accuracy_knn)
print(&quot;Precision :&quot;,(precison_knn))
print(&quot;Recall :&quot;,recall_knn )
print(&quot;Error : &quot;,error_knn)
print(&quot;F1 Score : &quot;,fmes_knn)
auc_knn = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_knn)
gini_knn = 2*(auc_knn/100) - 1
print(&quot;GINI : &quot;,gini_knn)
print(&quot;Confusion matrix &quot;)
```

```
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
# LSVM L1 in case of Full Features
from sklearn.svm import LinearSVC
lsvmL1 = LinearSVC(dual = False, penalty = &#39;l1&#39;,max_iter = 10000)
lsvmL1.fit(x_train, y_train)
y_predict = lsvmL1.predict(x_test)
accuracy_lsvmL1 = accuracy_score(y_test,y_predict)
precison_lsvmL1 = precision_score(y_test, y_predict,pos_label=0 )
recall_lsvmL1 = recall_score(y_test, y_predict,pos_label=0)
fmes_lsvmL1 = f1_score(y_test,y_predict,zero_division =1)
accuracy_lsvmL1 *= 100
precison_lsvmL1 *= 100
recall_lsvmL1*= 100
error_lsvmL1 = 100 - accuracy_lsvmL1
print(&quot;Accuracy :&quot;,accuracy_lsvmL1)
print(&quot;Precision&quot;,precison_lsvmL1)
print(&quot;Recall :&quot;,recall_lsvmL1 ) #max_iter = sys.maxsize
print(&quot;F1 Score : &quot;,fmes_lsvmL1)
print(&quot;Error : &quot;,error_lsvmL1)
auc_lsvmL1 = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_lsvmL1)
gini_lsvmL1 = 2*(auc_lsvmL1/100) - 1
print(&quot;GINI : &quot;,gini_lsvmL1)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
# LSVM L2 in case of Full Features
from sklearn.svm import LinearSVC
```

```
import sys
lsvmL2 = LinearSVC( dual=False,penalty=&#39;l2&#39;,max_iter = 10000)
lsvmL2.fit(x_train, y_train)
y_predict = lsvmL2.predict(x_test)
accuracy_lsvmL2 = accuracy_score(y_test,y_predict)
precison_lsvmL2 = precision_score(y_test, y_predict,pos_label=0 )
recall_lsvmL2 = recall_score(y_test, y_predict,pos_label=0)
fmes_lsvmL2 = f1_score(y_test,y_predict,zero_division =1)
accuracy_lsvmL2 *= 100
precison_lsvmL2 *= 100
recall_lsvmL2 *= 100
error_lsvmL2 = 100 - accuracy_lsvmL2
print(&quot;Accuracy :&quot;,accuracy_lsvmL2)
print(&quot;Precision&quot;,precison_lsvmL2)
print(&quot;Recall :&quot;,recall_lsvmL2 )
print(&quot;Error : &quot;,error_lsvmL2)
auc_lsvmL2 = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_lsvmL2)
gini_lsvmL2 = 2*(auc_lsvmL2/100) - 1
print(&quot;GINI : &quot;,gini_lsvmL2)
print(&quot;F1 Score : &quot;,fmes_lsvmL2)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
# RF in case of Full features
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 20, criterion=&quot;entropy&quot;)
classifier.fit(x_train, y_train)
y_predict= classifier.predict(x_test)
accuracy_rf = accuracy_score(y_test,y_predict)
```

```
precision_rf = precision_score(y_test, y_predict,pos_label=0)
recall_rf = recall_score(y_test, y_predict,pos_label=0)
fmes_rf = f1_score(y_test,y_predict,zero_division =1)
accuracy_rf *= 100
recall_rf *= 100
precision_rf *=100
error_lsvmL2 = 100 - accuracy_rf
print(&quot;Accuracy :&quot;,accuracy_rf)
print(&quot;Precision&quot;,precision_rf)
print(&quot;Recall :&quot;,recall_rf )
print(&quot;Error : &quot;,error_lsvmL2)
print(&quot;F Score : &quot;,fmes_rf)
auc_rf = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_rf)
gini_rf = 2*(auc_rf/100) - 1
print(&quot;GINI : &quot;,gini_rf)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
Data.columns = ["Data.columns[:-1], &#39;Decision&#39;]
Data.head()
DataTree = pd.DataFrame()
DataTree = Data
DataTree[&#39;Decision&#39;] = dec
DataTree
X_all_features = DataTree.iloc[:,0:24]
Y_all_features = DataTree[&#39;Decision&#39;]
x1_train, x1_test, y1_train, y1_test = train_test_split( X_all_features, Y_all_features, test_size =
0.33)
display(x1_train.shape,y1_train.shape,x1_test.shape,y1_test.shape)
```

```
x1_train
y1_train
df = pd.DataFrame()
df = x1_train
print(type(df))
df.join(y1_train)
df.info()
# CHAID in case of Full Features
from chefboost import Chefboost as cf
config = {&quot;algorithm&quot; : &quot;CHAID&quot;}
ChaidTree = cf.fit(df.join(y1_train), config)
print(type(ChaidTree))
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test.index:
pred = cf.predict(ChaidTree,x1_test.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
tn += 1
y_predict.append(0)
y_testing.append(0)
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
accuracy_chaid = (tp+tn)/(tp+tn+fp+fn)
precision_chaid = tp/(tp+fp)
recall_chaid = tp/(tp+fn)
accuracy_chaid *= 100
precision_chaid *= 100
recall_chaid *= 100
print(accuracy_chaid)
print(precision_chaid)
print(recall_chaid)
error_chaid = 100 - accuracy_chaid
print(&quot;Error : &quot;,error_chaid)
fmes_chaid = 2*((precision_chaid*recall_chaid)/(precision_chaid+recall_chaid))/100
print(fmes_chaid)
auc_chaid = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_chaid)
gini_chaid = 2*(auc_chaid/100) - 1
print(&quot;GINI : &quot;,gini_chaid)
# C4.5 in case of Full Features
config = {&quot;algorithm&quot; : &quot;C4.5&quot;}
C45Tree = cf.fit(df.join(y1_train), config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test.index:
pred = cf.predict(C45Tree,x1_test.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
fp+=1
y_predict.append(1)
y_testing.append(0)
else:
fn+=1
y_predict.append(0)
y_testing.append(1)
accuracy_c45 = (tp+tn)/(tp+tn+fp+fn)
accuracy_c45 *= 100
print(accuracy_c45)
precision_c45 = tp/(tp+fp)
precision_c45 *= 100
recall_c45 = tp/(tp+fn)
recall_c45 *= 100
print(precision_c45)
print(recall_c45)
error_c45 = 100 - accuracy_c45
print(&quot;Error : &quot;,error_c45)
fmes_c45 = 2*((precision_c45*recall_c45)/(precision_c45+recall_c45))/100
print(fmes_c45)
auc_c45 = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_c45)
gini_c45 = 2*(auc_c45/100) - 1
print(&quot;GINI : &quot;,gini_c45)
# ANN in case of Full Features
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import confusion_matrix, accuracy_score,precision_score,recall_score
classifier = Sequential()
classifier.add(Dense(units = 6, kernel_initializer = &#39;uniform&#39;, activation = &#39;relu&#39;, input_dim =24))
classifier.add(Dense(units = 6, kernel_initializer = &#39;uniform&#39;, activation = &#39;relu&#39;))
classifier.add(Dense(units = 1, kernel_initializer = &#39;uniform&#39;, activation = &#39;sigmoid&#39;))
classifier.compile(optimizer = &#39;adam&#39;, loss = &#39;binary_crossentropy&#39;, metrics = [&#39;accuracy&#39;])
Y_train = y_train
Y_test = y_test
y_train = y_train.astype(&#39;category&#39;)
y_train = y_train.cat.codes
y_test = y_test.astype(&#39;category&#39;)
y_test = y_test.cat.codes
classifier.fit(x_train, y_train, batch_size = 10, epochs = 100)
y_pred = classifier.predict(x_test)
y_pred = (y_pred &gt; 0.5)
11
accuracy_ann = accuracy_score(y_test,y_pred)
accuracy_ann *= 100
precision_ann = precision_score(y_test,y_pred)
precision_ann *= 100
recall_ann = recall_score(y_test,y_pred)
recall_ann *= 100
print(&quot;Accuracy :&quot;,accuracy_ann)
print(&quot;Precision&quot;,precision_ann)
print(&quot;Recall :&quot;,recall_ann)
error_ann = 100 - accuracy_ann
print(&quot;Error : &quot;,error_ann)
fmes_ann = f1_score(y_test,y_pred,zero_division =1)
print(&quot;F1 Score : &quot;,fmes_ann)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_pred))
auc_ann = roc_auc_score(y_test, y_pred)*100
print(&quot;AUC : &quot;,auc_ann)
gini_ann = 2*(auc_ann/100) - 1
print(&quot;GINI : &quot;,gini_ann)
# Evaluation Measures Plotting in case of Full Features
import matplotlib.pyplot as plt
import numpy as np
barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))
accuracy = [accuracy_lr,accuracy_knn,accuracy_lsvmL1
,accuracy_lsvmL2,accuracy_chaid,accuracy_c45, accuracy_ann, accuracy_rf ]
precision = [precison_lr,precison_knn,precison_lsvmL1,precison_lsvmL2,
precision_chaid,precision_c45,precision_ann, precision_rf ]
recall = [recall_lr, recall_knn, recall_lsvmL1,recall_lsvmL2,recall_chaid, recall_c45, recall_ann,
recall_rf ]
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br1, accuracy, color =&#39;r&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Accuracy&#39;)
plt.bar(br2, precision,color =&#39;g&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Precision&#39;)
plt.bar(br3, recall, color =&#39;b&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Recall&#39;)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Performance&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
[&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;, &#39;C4.5&#39;, &#39;ANN&#39;, &#39;RF&#39; ] )
plt.title(&quot;Accuracy Precision Recall for all features&quot;)
plt.legend()
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;, &#39;RF&#39;]
f_mesures = [fmes_lr, fmes_knn, fmes_lsvmL1,fmes_lsvmL2,fmes_chaid, fmes_c45, fmes_ann,
fmes_rf]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;F Measure&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of F Measure for Various Classifiers using Full Features&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;, &#39;RF&#39;]
auc = [auc_lr, auc_knn, auc_lsvmL1,auc_lsvmL2,auc_chaid, auc_c45, auc_ann, auc_rf]
```

```
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;AUC&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of AUC for Various Classifiers using Full Features&quot;)
plt.show()
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
gini = [gini_lr, gini_knn, gini_lsvmL1,gini_lsvmL2,gini_chaid, gini_c45, gini_ann, gini_rf]
plt.rcParams[&quot;figure.figsize&quot;] = (30,10)
plt.plot(algo, gini)
plt.xlabel(&quot;Algorithm&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.ylabel(&quot;GINI&quot;,fontweight =&#39;bold&#39;, fontsize = 25)
plt.title(&quot;Variation of GINI for Various Classifiers using Full Features&quot;, fontweight
=&#39;bold&#39;,
fontsize = 25)
plt.show()
err = plt.figure()
x = err.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
errors = [100-accuracy_lr,100-accuracy_knn,100-accuracy_lsvmL1 ,100
accuracy_lsvmL2,100-accuracy_chaid,100-accuracy_c45, 100-accuracy_ann, 100-
accuracy_rf ]
x.bar(algo,errors)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Error&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of Error for Various Classifiers using Full Features&quot;)
plt.show()
# CFS
def correlation(dataSeT, threshold):
col_corr = set()
corr_matrix = dataSeT.corr()
for i in range(len(corr_matrix.columns)):
for j in range(i):
if abs(corr_matrix.iloc[i,j])&gt;threshold :
col_corr.add(corr_matrix.columns[i])
return col_corr
corr_features = correlation(x_train,0.5)
corr_features
x_train_corr = x_train.drop(corr_features, axis = 1)
x1_train_corr = x1_train.drop(corr_features, axis = 1)
x_test_corr = x_test.drop(corr_features, axis = 1)
x1_test_corr = x1_test.drop(corr_features, axis = 1)
x1_train_corr
y1_train
x_train_corr
y_train
sns.heatmap(x_train_corr.corr(),annot = True)
x_train_corr
y1_train
# LR
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, accuracy_score, recall_score
import sys
lr = LogisticRegression(solver = &#39;lbfgs&#39;, max_iter = sys.maxsize )
lr.fit(x_train_corr, y_train)
y_predict = lr.predict(x_test_corr)
accuracy_lr_corr = accuracy_score(y_test,y_predict)
precison_lr_corr = precision_score(y_test, y_predict,pos_label= 0 )
recall_lr_corr = recall_score(y_test, y_predict,pos_label=0)
fmes_lr_corr = f1_score(y_test,y_predict,zero_division =1)
accuracy_lr_corr *= 100
precison_lr_corr *= 100
recall_lr_corr *= 100
error_lr_corr = 100 - accuracy_lr_corr
print(&quot;Error : &quot;,error_lr_corr)
print(&quot;Accuracy :&quot;,accuracy_lr_corr)
print(&quot;Precision&quot;,precison_lr_corr)
print(&quot;Recall :&quot;,recall_lr_corr )
print(&quot;F1 Score : &quot;,fmes_lr_corr)
auc_lr_corr = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_lr_corr)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
auc_lr_corr = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_lr_corr)
gini_lr_corr = 2*(auc_lr_corr/100) - 1
print(&quot;GINI : &quot;,gini_lr_corr)
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train_corr, y_train)
y_predict_corr = knn.predict(x_test_corr)
accuracy_knn_corr = accuracy_score(y_test,y_predict_corr)
precison_knn_corr = precision_score(y_test, y_predict_corr,pos_label= 0 )
recall_knn_corr = recall_score(y_test, y_predict_corr,pos_label= 0)
fmes_knn_corr = f1_score(y_test,y_predict_corr,zero_division =1)
accuracy_knn_corr *= 100
precison_knn_corr *= 100
recall_knn_corr *= 100
print(&quot;Accuracy :&quot;,accuracy_knn_corr)
print(&quot;Precision&quot;,precison_knn_corr)
print(&quot;Recall :&quot;,recall_knn_corr )
error_knn_corr = 100 - accuracy_knn_corr
print(&quot;Error : &quot;,error_knn_corr)
print(&quot;F1 Score : &quot;,fmes_knn_corr)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_corr))
print(&quot;Correct Predictions :&quot;,sum(y_predict_corr == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_corr != y_test))
auc_knn_corr = roc_auc_score(y_test, y_predict_corr)*100
print(&quot;AUC : &quot;,auc_knn_corr)
gini_knn_corr = 2*(auc_knn_corr/100) - 1
print(&quot;GINI : &quot;,gini_knn_corr)
# LSVM L1
from sklearn.svm import LinearSVC
lsvmL1 = LinearSVC(dual = False, penalty = &#39;l1&#39;,max_iter = 10000)
lsvmL1.fit(x_train_corr, y_train)
y_predict_corr = lsvmL1.predict(x_test_corr)
accuracy_lsvmL1_corr = accuracy_score(y_test,y_predict_corr)
precison_lsvmL1_corr = precision_score(y_test, y_predict_corr,pos_label=0 )
recall_lsvmL1_corr = recall_score(y_test, y_predict_corr,pos_label=0)
fmes_lsvmL1_corr = f1_score(y_test,y_predict_corr,zero_division =1)
accuracy_lsvmL1_corr *= 100
precison_lsvmL1_corr *= 100
recall_lsvmL1_corr *= 100
error_lsvmL1_corr = 100 - accuracy_lsvmL1_corr
print(&quot;Error : &quot;,error_lsvmL1_corr)
print(&quot;Accuracy :&quot;,accuracy_lsvmL1_corr)
print(&quot;Precision&quot;,precison_lsvmL1_corr)
print(&quot;Recall :&quot;,recall_lsvmL1_corr ) #max_iter = sys.maxsize
print(&quot;F1 Score : &quot;,fmes_lsvmL1_corr)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_corr))
print(&quot;Correct Predictions :&quot;,sum(y_predict_corr == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_corr != y_test))
auc_lsvmL1_corr = roc_auc_score(y_test, y_predict_corr)*100
print(&quot;AUC : &quot;,auc_lsvmL1_corr)
gini_lsvmL1_corr = 2*(auc_lsvmL1_corr/100) - 1
print(&quot;GINI : &quot;,gini_lsvmL1_corr)
# LSVM L2
from sklearn.svm import LinearSVC
import sys
lsvmL2 = LinearSVC( dual=False,penalty=&#39;l2&#39;,max_iter = 10000)
lsvmL2.fit(x_train_corr, y_train)
y_predict_corr = lsvmL2.predict(x_test_corr)
accuracy_lsvmL2_corr = accuracy_score(y_test,y_predict_corr)
precison_lsvmL2_corr = precision_score(y_test, y_predict_corr,pos_label=0 )
recall_lsvmL2_corr = recall_score(y_test, y_predict_corr,pos_label=0)
fmes_lsvmL2_corr = f1_score(y_test,y_predict_corr,zero_division =1)
accuracy_lsvmL2_corr *= 100
precison_lsvmL2_corr *= 100
recall_lsvmL2_corr *= 100
print(&quot;Accuracy :&quot;,accuracy_lsvmL2_corr)
print(&quot;Precision&quot;,precison_lsvmL2_corr)
print(&quot;Recall :&quot;,recall_lsvmL2_corr )
error_lsvmL2_corr = 100 - accuracy_lsvmL2_corr
print(&quot;Error : &quot;,error_lsvmL2_corr)
print(&quot;F1 Score : &quot;,fmes_lsvmL2_corr)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_corr))
print(&quot;Correct Predictions :&quot;,sum(y_predict_corr == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_corr != y_test))
auc_lsvmL2_corr = roc_auc_score(y_test, y_predict_corr)*100
print(&quot;AUC : &quot;,auc_lsvmL2_corr)
gini_lsvmL2_corr = 2*(auc_lsvmL2_corr/100) - 1
print(&quot;GINI : &quot;,gini_lsvmL2_corr)
# RF
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion=&quot;entropy&quot;)
classifier.fit(x_train_corr, y_train)
y_predict_corr= classifier.predict(x_test_corr)
accuracy_rf_corr = accuracy_score(y_test,y_predict_corr)
precision_rf_corr = precision_score(y_test, y_predict_corr,pos_label=0)
recall_rf_corr = recall_score(y_test, y_predict_corr,pos_label=0)
fmes_rf_corr = f1_score(y_test,y_predict_corr,zero_division =1)
accuracy_rf_corr *= 100
precision_rf_corr *= 100
recall_rf_corr *= 100
print(&quot;Accuracy :&quot;,accuracy_rf_corr)
print(&quot;Precision&quot;,precision_rf_corr)
print(&quot;Recall :&quot;,recall_rf_corr )
error_rf_corr = 100 - accuracy_rf_corr
print(&quot;Error : &quot;,error_rf_corr)
print(&quot;F1 Score : &quot;,fmes_rf_corr)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_corr))
print(&quot;Correct Predictions :&quot;,sum(y_predict_corr == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_corr != y_test))
auc_rf_corr = roc_auc_score(y_test, y_predict_corr)*100
print(&quot;AUC : &quot;,auc_rf_corr)
gini_rf_corr = 2*(auc_rf_corr/100) - 1
print(&quot;GINI : &quot;,gini_rf_corr)
# ANN
import keras
# C4.5
config = {&quot;algorithm&quot; : &quot;C4.5&quot;}
C45Tree = cf.fit(df.join(y1_train), config)
#print(type(ChaidTree))
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test_corr.index:
pred = cf.predict(C45Tree,x1_test_corr.loc[i, :])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
accuracy_c45_corr = (tp+tn)/(tp+tn+fp+fn)
accuracy_c45_corr *=100
print(accuracy_c45_corr)
precision_c45_corr = tp/(tp+fp)
precision_c45_corr*=100
recall_c45_corr = tp/(tp+fn)
recall_c45_corr*=100
print(precision_c45_corr)
print(recall_c45_corr)
error_c45_corr = 100 - accuracy_c45_corr
print(&quot;Error : &quot;,error_c45_corr)
fmes_c45_corr =
2*((precision_c45_corr*recall_c45_corr)/(precision_c45_corr+recall_c45_corr))/100
print(fmes_c45_corr)
auc_c45_corr = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_c45_corr)
gini_c45_corr = 2*(auc_c45_corr/100) - 1
# Evaluation Measures Plotting in case of CFS
import matplotlib.pyplot as plt
import numpy as np
barWidth = 0.25
fig2 = plt.subplots(figsize =(12, 8))
accuracy = [accuracy_lr_corr,accuracy_knn_corr,accuracy_lsvmL1_corr
,accuracy_lsvmL2_corr,accuracy_ann_corr,accuracy_chaid_corr,accuracy_c45_corr,accuracy_rf
_corr ]
precision = [precison_lr_corr,precison_knn_corr,precison_lsvmL1_corr,precison_lsvmL2_corr,
precision_ann_corr,precision_chaid_corr,precision_c45_corr,precision_rf_corr ]
recall = [recall_lr_corr, recall_knn_corr, recall_lsvmL1_corr,recall_lsvmL2_corr,
recall_ann_corr,recall_chaid_corr, recall_c45_corr,recall_rf_corr ]
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br1, accuracy, color =&#39;r&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Accuracy&#39;)
plt.bar(br2, precision,color =&#39;g&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Precision&#39;)
plt.bar(br3, recall, color =&#39;b&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Recall&#39;)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Performance&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
[&#39;LR&#39;,&#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;ANN&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;, &#39;RF&#39; ])
plt.legend()
plt.title(&quot;Accuracy Precision Recall for correlation based feature selection&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
f_mesures = [fmes_lr_corr, fmes_knn_corr,
fmes_lsvmL1_corr,fmes_lsvmL2_corr,fmes_chaid_corr, fmes_c45_corr, fmes_ann_corr,
fmes_rf_corr]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;F Measure&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of F Measure for Various Classifiers using Correlation Based Feature
selection&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
auc = [auc_lr_corr, auc_knn_corr, auc_lsvmL1_corr,auc_lsvmL2_corr,auc_chaid_corr,
```

```
auc_c45_corr, auc_ann_corr, auc_rf_corr]

x.bar(algo,auc)

plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)

plt.ylabel(&#39;AUC&#39;, fontweight =&#39;bold&#39;, fontsize = 15)

plt.title(&quot;Variation of AUC for Various Classifiers using Correlation Based Feature selection&quot;)

plt.show()

algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]

gini = [gini_lr_corr, gini_knn_corr, gini_lsvmL1_corr,gini_lsvmL2_corr,gini_chaid_corr,

gini_c45_corr, gini_ann_corr,gini_rf_corr]

plt.rcParams[&quot;figure.figsize&quot;] = [30,10]

plt.plot(algo, gini)

plt.xlabel(&quot;Algorithm&quot;, fontweight =&#39;bold&#39;, fontsize = 25)

plt.ylabel(&quot;GINI&quot;, fontweight =&#39;bold&#39;, fontsize = 25)

plt.title(&quot;Variation of GINI for Various Classifiers using Correlation Based Feature selection&quot;,

fontweight =&#39;bold&#39;, fontsize = 25)

plt.show()

error = plt.figure()

x = error.add_axes([0,0,0.3,0.3])

algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]

auc = [100-accuracy_lr_corr,100-accuracy_knn_corr,100-accuracy_lsvmL1_corr ,100-

accuracy_lsvmL2_corr,100-accuracy_chaid_corr,100-accuracy_c45_corr,100-

accuracy_ann_corr,100-accuracy_rf_corr ]

x.bar(algo,auc)

plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)

plt.ylabel(&#39;Error&#39;, fontweight =&#39;bold&#39;, fontsize = 15)

plt.title(&quot;Variation of Error for Various Classifiers using Correlation Based Feature selection&quot;)

plt.show()

Data.info()
```

```
# Wrapper FS

x_lasso = Data.iloc[:,0:24]

y_lasso = Data[&#39;Decision&#39;]

y_lasso = y_lasso.astype(&#39;category&#39;)

y_lasso = y_lasso.cat.codes

import statsmodels.api as sm

def forward_selection(data, target, significance_level=0.05):

initial_features = data.columns.tolist()

best_features = []

while (len(initial_features)&gt;0):

remaining_features = list(set(initial_features)-set(best_features))

new_pval = pd.Series(index=remaining_features,dtype = &quot;float64&quot;)

pd.to_numeric(new_pval, errors = &#39;coerce&#39;)

for new_column in remaining_features:

model = sm.OLS(target.astype(float),

sm.add_constant(data[best_features+[new_column]]).astype(float)).fit()

new_pval[new_column] = model.pvalues[new_column]

min_p_value = new_pval.min()

if(float(min_p_value)&lt;significance_level):

best_features.append(new_pval.idxmin())

else:

break

return best_features

X = forward_selection(x_lasso,y_lasso)

print(X)

x_train_wrp = x_train.drop(X, axis = 1)

x1_train_wrp = x1_train.drop(X, axis = 1)

x_test_wrp = x_test.drop(X, axis = 1)

x1_test_wrp = x1_test.drop(X, axis = 1)

x_train_wrp

y_train
```

```
sns.heatmap(x_train_wrp.corr(),annot = True)

# LR

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import precision_score, accuracy_score, recall_score

import sys

lr = LogisticRegression(solver = &#39;lbfgs&#39;, max_iter = sys.maxsize )

lr.fit(x_train_wrp, y_train)

y_predict = lr.predict(x_test_wrp)

accuracy_lr_wrp = accuracy_score(y_test,y_predict)*100

precison_lr_wrp = precision_score(y_test, y_predict,pos_label= 0 )*100

recall_lr_wrp = recall_score(y_test, y_predict,pos_label=0)*100

fmes_lr_wrp = f1_score(y_test,y_predict,zero_division =1)

print(&quot;Accuracy :&quot;,accuracy_lr_wrp)

print(&quot;Precision&quot;,precison_lr_wrp)

print(&quot;Recall :&quot;,recall_lr_wrp )

error_lr_wrp = 100 - accuracy_lr_wrp

print(&quot;Error : &quot;,error_lr_wrp)

print(&quot;F1 Score : &quot;,fmes_lr_wrp)

print(&quot;Confusion matrix:&quot;)

print(confusion_matrix(y_test,y_predict))

print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))

print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))

auc_lr_wrp = roc_auc_score(y_test, y_predict)*100

print(&quot;AUC : &quot;,auc_lr_wrp)

gini_lr_wrp = 2*(auc_lr_wrp/100) - 1

print(&quot;GINI : &quot;,gini_lr_wrp)

# KNN

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(x_train_wrp, y_train)

y_predict_wrp = knn.predict(x_test_wrp)
```

```
accuracy_knn_wrp = accuracy_score(y_test,y_predict_wrp)*100

precison_knn_wrp = precision_score(y_test, y_predict_wrp,pos_label= 0 )*100

recall_knn_wrp = recall_score(y_test, y_predict_wrp,pos_label= 0)*100

fmes_knn_wrp = f1_score(y_test,y_predict_wrp,zero_division =1)

print(&quot;Accuracy :&quot;,accuracy_knn_wrp)

print(&quot;Precision&quot;,precison_knn_wrp)

print(&quot;Recall :&quot;,recall_knn_wrp )

error_knn_wrp = 100 - accuracy_knn_wrp

print(&quot;Error : &quot;,error_knn_wrp)

print(&quot;F1 Score : &quot;,fmes_knn_wrp)

print(&quot;Confusion matrix:&quot;)

print(confusion_matrix(y_test,y_predict_wrp))

print(&quot;Correct Predictions :&quot;,sum(y_predict_wrp == y_test))

print(&quot;Incorrect Predictions :&quot;,sum(y_predict_wrp != y_test))

auc_knn_wrp = roc_auc_score(y_test, y_predict_wrp)*100

print(&quot;AUC : &quot;,auc_knn_wrp)

gini_knn_wrp = 2*(auc_knn_wrp/100) - 1

print(&quot;GINI : &quot;,gini_knn_wrp)

# LSVM L1

from sklearn.svm import LinearSVC

lsvmL1 = LinearSVC(dual = False, penalty = &#39;l1&#39;,max_iter = 10000)

lsvmL1.fit(x_train_wrp, y_train)

y_predict_wrp = lsvmL1.predict(x_test_wrp)

accuracy_lsvmL1_wrp = accuracy_score(y_test,y_predict_wrp)*100

precison_lsvmL1_wrp = precision_score(y_test, y_predict_wrp,pos_label=0 )*100

recall_lsvmL1_wrp = recall_score(y_test, y_predict_wrp,pos_label=0)*100

fmes_lsvmL1_wrp = f1_score(y_test,y_predict_wrp,zero_division =1)

print(&quot;Accuracy :&quot;,accuracy_lsvmL1_wrp)

print(&quot;Precision&quot;,precison_lsvmL1_wrp)

print(&quot;Recall :&quot;,recall_lsvmL1_wrp ) #max_iter = sys.maxsize

error_lsvmL1_wrp = 100 - accuracy_lsvmL1_wrp
```

```
print(&quot;Error : &quot;,error_lsvmL1_wrp)

print(&quot;F1 Score : &quot;,fmes_lsvmL1_wrp)

print(&quot;Confusion matrix:&quot;)

print(confusion_matrix(y_test,y_predict_wrp))

print(&quot;Correct Predictions :&quot;,sum(y_predict_wrp == y_test))

print(&quot;Incorrect Predictions :&quot;,sum(y_predict_wrp != y_test))

auc_lsvmL1_wrp = roc_auc_score(y_test, y_predict_wrp)*100

print(&quot;AUC : &quot;,auc_lsvmL1_wrp)

gini_lsvmL1_wrp = 2*(auc_lsvmL1_wrp/100) - 1

print(&quot;GINI : &quot;,gini_lsvmL1_wrp)

# LSVM L2

from sklearn.svm import LinearSVC

import sys

lsvmL2 = LinearSVC( dual=False,penalty=&#39;l2&#39;,max_iter = 120000)

lsvmL2.fit(x_train_wrp, y_train)

y_predict_wrp = lsvmL2.predict(x_test_wrp)

accuracy_lsvmL2_wrp = accuracy_score(y_test,y_predict_wrp)*100

precison_lsvmL2_wrp = precision_score(y_test, y_predict_wrp,pos_label=0 )*100

recall_lsvmL2_wrp = recall_score(y_test, y_predict_wrp,pos_label=0)*100

fmes_lsvmL2_wrp = f1_score(y_test,y_predict_wrp,zero_division =1)

print(&quot;Accuracy :&quot;,accuracy_lsvmL2_wrp)

print(&quot;Precision&quot;,precison_lsvmL2_wrp)

print(&quot;Recall :&quot;,recall_lsvmL2_wrp )

error_lsvmL2_wrp = 100 - accuracy_lsvmL2_wrp

print(&quot;Error : &quot;,error_lsvmL2_wrp)

print(&quot;F1 Score : &quot;,fmes_lsvmL2_wrp)

print(&quot;Confusion matrix:&quot;)

print(confusion_matrix(y_test,y_predict_wrp))

auc_lsvmL2_wrp = roc_auc_score(y_test, y_predict_wrp)*100

print(&quot;AUC : &quot;,auc_lsvmL2_wrp)

print(&quot;Correct Predictions :&quot;,sum(y_predict_wrp == y_test))
```

```
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_wrp != y_test))

gini_lsvmL2_wrp = 2*(auc_lsvmL2_wrp/100) - 1

print(&quot;GINI : &quot;,gini_lsvmL2_wrp)

# RF

from sklearn.ensemble import RandomForestClassifier

classifier= RandomForestClassifier(n_estimators= 10, criterion=&quot;entropy&quot;)

classifier.fit(x_train_wrp, y_train)

y_predict_wrp= classifier.predict(x_test_wrp)

accuracy_rf_wrp = accuracy_score(y_test,y_predict_wrp)*100

precision_rf_wrp = precision_score(y_test, y_predict_wrp,pos_label=0)*100

recall_rf_wrp = recall_score(y_test, y_predict_wrp,pos_label=0)*100

fmes_rf_wrp = f1_score(y_test,y_predict_wrp,zero_division =1)

print(&quot;Accuracy :&quot;,accuracy_rf_wrp)

print(&quot;Precision&quot;,precision_rf_wrp)

print(&quot;Recall :&quot;,recall_rf_wrp )

error_rf_wrp = 100 - accuracy_rf_wrp

print(&quot;Error : &quot;,error_rf_wrp)

print(&quot;F1 Score : &quot;,fmes_rf_wrp)

print(&quot;Confusion matrix:&quot;)

print(confusion_matrix(y_test,y_predict_wrp))

print(&quot;Correct Predictions :&quot;,sum(y_predict_wrp == y_test))

print(&quot;Incorrect Predictions :&quot;,sum(y_predict_wrp != y_test))

auc_rf_wrp = roc_auc_score(y_test, y_predict_wrp)*100

print(&quot;AUC : &quot;,auc_rf_wrp)

gini_rf_wrp = 2*(auc_rf_wrp/100) - 1

print(&quot;GINI : &quot;,gini_rf_wrp)

# ANN

import keras

from keras.models import Sequential

from keras.layers import Dense

from sklearn.metrics import confusion_matrix, accuracy_score,precision_score,recall_score
```

```
classifier = Sequential()

classifier.add(Dense(units = 5, kernel_initializer = &#39;uniform&#39;, activation = &#39;relu&#39;, input_dim = 14))

classifier.add(Dense(units = 5, kernel_initializer = &#39;uniform&#39;, activation = &#39;relu&#39;))

classifier.add(Dense(units = 1, kernel_initializer = &#39;uniform&#39;, activation = &#39;sigmoid&#39;))

classifier.compile(optimizer = &#39;adam&#39;, loss = &#39;binary_crossentropy&#39;, metrics = [&#39;accuracy&#39;])

Y_train = y_train

Y_test = y_test

y_train = y_train.astype(&#39;category&#39;)

y_train = y_train.cat.codes

y_test = y_test.astype(&#39;category&#39;)

y_test = y_test.cat.codes

classifier.fit(x_train_wrp, y_train, batch_size = 10, epochs = 100)

y_pred = classifier.predict(x_test_wrp)

y_pred = (y_pred &gt; 0.5)

accuracy_ann_wrp = accuracy_score(y_test,y_pred)*100

precision_ann_wrp = precision_score(y_test,y_pred)*100

recall_ann_wrp = recall_score(y_test,y_pred)*100

print(&quot;Accuracy :&quot;,accuracy_ann_wrp)

print(&quot;Precision&quot;,precision_ann_wrp)

print(&quot;Recall :&quot;,recall_ann_wrp)

error_ann_wrp = 100 - accuracy_ann_wrp

print(&quot;Error : &quot;,error_ann_wrp)

fmes_ann_wrp = f1_score(y_test,y_predict,zero_division =1)

print(&quot;F1 Score : &quot;,fmes_ann_wrp)

print(&quot;Confusion matrix:&quot;)

print(confusion_matrix(y_test,y_pred))

print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))

print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))

auc_ann_wrp = roc_auc_score(y_test, y_predict_wrp)*100
```

```
print(&quot;AUC : &quot;,auc_ann_wrp)

gini_ann_wrp = 2*(auc_ann_wrp/100) - 1

print(&quot;GINI : &quot;,gini_ann_wrp)

# CHAID

from chefboost import Chefboost as cf

from pandas.api.types import CategoricalDtype

config = [&quot;algorithm&quot; : &quot;CHAID&quot;]

df = pd.DataFrame()

df = x1_train_wrp

print(df)

ChaidTree = cf.fit(df.join(y1_train),config)

tp = 0

tn = 0

fp = 0

fn = 0

y_predict = []

y_testing = []

for i in x1_test.index:

pred = cf.predict(ChaidTree,x1_test_wrp.loc[i, :])

if pred == y1_test[i]:

if pred == &#39;ckd&#39;:

y_predict.append(1)

y_testing.append(1)

tp += 1

else:

y_predict.append(0)

y_testing.append(0)

tn += 1

else:

if pred == &#39;ckd&#39;:

y_predict.append(1)
```

```
y_testing.append(0)

fp+=1

else:

y_predict.append(0)

y_testing.append(1)

fn+=1

accuracy_chaid_wrp = ((tp+tn)/(tp+tn+fp+fn))*100

precision_chaid_wrp = (tp/(tp+fp))*100

recall_chaid_wrp = (tp/(tp+fn))*100

print(accuracy_chaid_wrp)

print(precision_chaid_wrp)

print(recall_chaid_wrp)

error_chaid_wrp = 100 - accuracy_chaid_wrp

print(&quot;Error : &quot;,error_chaid_wrp)

fmes_chaid_wrp =

2*(precision_chaid_wrp*recall_chaid_wrp)/((precision_chaid_wrp+recall_chaid_wrp)*100)

print(fmes_chaid_wrp)

auc_chaid_wrp = roc_auc_score(y_testing, y_predict)*100

print(&quot;AUC : &quot;,auc_chaid_wrp)

gini_chaid_wrp = 2*(auc_chaid_wrp/100) - 1

print(&quot;GINI : &quot;,gini_chaid_wrp)

# C4.5

config = [&quot;algorithm&quot; : &quot;C4.5&quot;]

C45Tree = cf.fit(df.join(y1_train), config)

tp = 0

tn = 0

fp = 0

fn = 0

y_predict = []

y_testing = []

for i in x1_test_corr.index:
```

```
pred = cf.predict(C45Tree,x1_test_wrp.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
accuracy_c45_wrp = (tp+tn)*100/(tp+tn+fp+fn)
print(accuracy_c45_wrp)
precision_c45_wrp = tp*100/(tp+fp)
recall_c45_wrp = tp*100/(tp+fn)
print(precision_c45_wrp)
print(recall_c45_wrp )
error_c45_wrp = 100 - accuracy_c45_wrp
print(&quot;Error : &quot;,error_c45_wrp)
fmes_c45_wrp =
2*(precision_c45_wrp*recall_c45_wrp)/((precision_c45_wrp+recall_c45_wrp)*100)
print(fmes_c45_wrp)
auc_c45_wrp = roc_auc_score(y_testing, y_predict)*100
```

```
print(&quot;AUC : &quot;,auc_c45_wrp)
gini_c45_wrp = 2*(auc_c45_wrp/100) - 1
print(&quot;GINI : &quot;,gini_c45_wrp)
# Evaluation Measures Plotting in case of Wrapper FS
fig2 = plt.subplots(figsize =(12, 8))
accuracy = [accuracy_lr_wrp,accuracy_knn_wrp,accuracy_lsvmL1_wrp
,accuracy_lsvmL2_wrp,accuracy_ann_wrp,accuracy_chaid_wrp,accuracy_c45_wrp,accuracy_rf
_wrp ]
precision = [precison_lr_wrp,precison_knn_wrp,precison_lsvmL1_wrp,precison_lsvmL2_wrp,
precision_ann_wrp,precision_chaid_wrp,precision_c45_wrp,precision_rf_wrp]
recall = [recall_lr_wrp, recall_knn_wrp, recall_lsvmL1_wrp,recall_lsvmL2_wrp,
recall_ann_wrp,recall_chaid_wrp, recall_c45_wrp,recall_rf_wrp ]
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br1, accuracy, color =&#39;r&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Accuracy&#39;)
plt.bar(br2, precision,color =&#39;g&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Precision&#39;)
plt.bar(br3, recall, color =&#39;b&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Recall&#39;)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Performance&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
[&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;ANN&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;, &#39;RF&#39; ] )
plt.title(&quot;Accuracy Precision Recall for Wrapper feature selection&quot;)
plt.legend()
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
```

```
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
f_mesures = [fmes_lr_wrp, fmes_knn_wrp,
fmes_lsvmL1_wrp,fmes_lsvmL2_wrp,fmes_chaid_wrp, fmes_c45_wrp,
fmes_ann_wrp,fmes_rf_wrp]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;F Measure&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of F Measure for Various Classifiers using Wrapper Feature
Selection&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
auc = [auc_lr_wrp, auc_knn_wrp, auc_lsvmL1_wrp,auc_lsvmL2_wrp,auc_chaid_wrp,
auc_c45_wrp, auc_ann_wrp,auc_rf_wrp]
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;AUC&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of AUC for Various Classifiers using Wrapper Feature Selection&quot;)
plt.show()
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
gini = [gini_lr_wrp, gini_knn_wrp, gini_lsvmL1_wrp,gini_lsvmL2_wrp,gini_chaid_wrp,
gini_c45_wrp, gini_ann_wrp,gini_rf_wrp]
plt.rcParams[&quot;figure.figsize&quot;] = (30,10)
plt.plot(algo, gini)
plt.xlabel(&quot;Algorithm&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.ylabel(&quot;GINI&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.title(&quot;Variation of GINI for Various Classifiers using Wrapper Feature Selection&quot;,
fontweight =&#39;bold&#39;, fontsize = 25)
```

```
plt.show()
error = plt.figure()
x = error.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
auc = [100-accuracy_lr_wrp,100-accuracy_knn_wrp,100-accuracy_lsvmL1_wrp ,100-
accuracy_lsvmL2_wrp,100-accuracy_chaid_wrp,100-accuracy_c45_wrp,100-
accuracy_ann_wrp,100-accuracy_rf_wrp ]
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Error&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of Error for Various Classifiers using Wrapper Feature Selection&quot;)
plt.show()
# LASSO FS
from sklearn.linear_model import Lasso
features = [i for i in data]
features.remove(&#39;sCKD&#39;)
print(features)
lasso = Lasso(alpha = 0.01)
lasso.fit(x_train,y_train)
coeff=lasso.coef_
for i in range(len(coeff)):
coeff[i] = abs(coeff[i])
df_lasso = pd.DataFrame({&quot;Features&quot; :features,&quot;Coefficient&quot; :coeff})
df_lasso.sort_values(&quot;Coefficient&quot;)
list_lasso = []
for i in range(len(df_lasso)):
if df_lasso[&#39;Coefficient&#39;][i] == 0:

11
list_lasso.append(df_lasso[&#39;Features&#39;][i])
```

```
list_lasso
x_train_lsso = x_train.drop(list_lasso, axis = 1)
x1_train_lsso = x1_train.drop(list_lasso, axis = 1)
x_test_lsso = x_test.drop(list_lasso, axis = 1)
x1_test_lsso = x1_test.drop(list_lasso, axis = 1)
sns.heatmap(x_train_lsso.corr(),annot = True)
# LR
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, accuracy_score, recall_score
import sys
lr = LogisticRegression(solver = &#39;lbfgs&#39;, max_iter = sys.maxsize )
lr.fit(x_train_lsso, y_train)
y_predict = lr.predict(x_test_lsso)
accuracy_lr_lsso = accuracy_score(y_test,y_predict)*100
precison_lr_lsso = precision_score(y_test, y_predict,pos_label= 0 )*100
recall_lr_lsso = recall_score(y_test, y_predict,pos_label=0)*100
fmes_lr_lsso = f1_score(y_test,y_predict,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_lr_lsso)
print(&quot;Precision&quot;,precison_lr_lsso)
print(&quot;Recall :&quot;,recall_lr_lsso )
error_lr_lsso = 100 - accuracy_lr_lsso
print(&quot;Error : &quot;,error_lr_lsso)
print(&quot;F1 Score : &quot;,fmes_lr_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
auc_lr_lsso = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_lr_lsso)
gini_lr_lsso = 2*(auc_lr_lsso/100) - 1
print(&quot;GINI : &quot;,gini_lr_lsso)
```

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train_lsso, y_train)
y_predict_lsso = knn.predict(x_test_lsso)
accuracy_knn_lsso = accuracy_score(y_test,y_predict_lsso)*100
precison_knn_lsso = precision_score(y_test, y_predict_lsso,pos_label= 0 )*100
recall_knn_lsso = recall_score(y_test, y_predict_lsso,pos_label= 0)*100
fmes_knn_lsso = f1_score(y_test,y_predict_lsso,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_knn_lsso)
print(&quot;Precision&quot;,precison_knn_lsso)
print(&quot;Recall :&quot;,recall_knn_lsso )
error_knn_lsso = 100 - accuracy_knn_lsso
print(&quot;Error : &quot;,error_knn_lsso)
print(&quot;F1 Score : &quot;,fmes_knn_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y_test))
auc_knn_lsso = roc_auc_score(y_test, y_predict_lsso)*100
print(&quot;AUC : &quot;,auc_knn_lsso)
gini_knn_lsso = 2*(auc_knn_lsso/100) - 1
print(&quot;GINI : &quot;,gini_knn_lsso)
# LSVM L1
from sklearn.svm import LinearSVC
lsvmL1 = LinearSVC(dual = False, penalty = &#39;l1&#39;,max_iter = 10000)
lsvmL1.fit(x_train_lsso, y_train)
y_predict_lsso = lsvmL1.predict(x_test_lsso)
accuracy_lsvmL1_lsso = accuracy_score(y_test,y_predict_lsso)*100
precison_lsvmL1_lsso= precision_score(y_test, y_predict_lsso,pos_label=0 )*100
recall_lsvmL1_lsso = recall_score(y_test, y_predict_lsso,pos_label=0)*100
```

```
fmes_lsvmL1_lsso = f1_score(y_test,y_predict_lsso,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_lsvmL1_lsso)
print(&quot;Precision&quot;,precison_lsvmL1_lsso)
print(&quot;Recall :&quot;,recall_lsvmL1_lsso) #max_iter = sys.maxsize
error_lsvmL1_lsso = 100 - accuracy_lsvmL1_lsso
print(&quot;Error : &quot;,error_lsvmL1_lsso)
print(&quot;F1 Score : &quot;,fmes_lsvmL1_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y_test))
auc_lsvmL1_lsso = roc_auc_score(y_test, y_predict_lsso)*100
print(&quot;AUC : &quot;,auc_lsvmL1_lsso)
gini_lsvmL1_lsso = 2*(auc_lsvmL1_lsso/100) - 1
print(&quot;GINI : &quot;,gini_lsvmL1_lsso)
# LSVM L2
from sklearn.svm import LinearSVC
import sys
lsvmL2 = LinearSVC( dual=False,penalty=&#39;l2&#39;,max_iter = 120000)
lsvmL2.fit(x_train_lsso, y_train)
y_predict_lsso = lsvmL2.predict(x_test_lsso)
accuracy_lsvmL2_lsso = accuracy_score(y_test,y_predict_lsso)*100
precison_lsvmL2_lsso = precision_score(y_test, y_predict_lsso,pos_label=0 )*100
recall_lsvmL2_lsso = recall_score(y_test, y_predict_lsso,pos_label=0)*100
fmes_lsvmL2_lsso = f1_score(y_test,y_predict_lsso,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_lsvmL2_lsso)
print(&quot;Precision&quot;,precison_lsvmL2_lsso)
print(&quot;Recall :&quot;,recall_lsvmL2_lsso)
error_lsvmL2_lsso = 100 - accuracy_lsvmL2_lsso
print(&quot;Error : &quot;,error_lsvmL2_lsso)
print(&quot;F1 Score : &quot;,fmes_lsvmL2_lsso)
```

```
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y_test))
auc_lsvmL2_lsso = roc_auc_score(y_test, y_predict_lsso)*100
print(&quot;AUC : &quot;,auc_lsvmL2_lsso)
gini_lsvmL2_lsso = 2*(auc_lsvmL2_lsso/100) - 1
print(&quot;GINI : &quot;,gini_lsvmL2_lsso)
# RF
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion=&quot;entropy&quot;)
classifier.fit(x_train_lsso, y_train)
y_predict_lsso= classifier.predict(x_test_lsso)
accuracy_rf_lsso = accuracy_score(y_test,y_predict_lsso)*100
precision_rf_lsso= precision_score(y_test, y_predict_lsso,pos_label=0)*100
recall_rf_lsso = recall_score(y_test, y_predict_lsso,pos_label=0)*100
fmes_rf_lsso = f1_score(y_test,y_predict_lsso,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_rf_lsso)
print(&quot;Precision&quot;,precision_rf_lsso)
print(&quot;Recall :&quot;,recall_rf_lsso)
error_rf_lsso = 100 - accuracy_rf_lsso
print(&quot;Error : &quot;,error_rf_lsso)
print(&quot;F1 Score : &quot;,fmes_rf_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y_test))
auc_rf_lsso = roc_auc_score(y_test, y_predict_lsso)*100
print(&quot;AUC : &quot;,auc_rf_lsso)
gini_rf_lsso = 2*(auc_rf_lsso/100) - 1
print(&quot;GINI : &quot;,gini_rf_lsso)
```

```
# ANN
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import confusion_matrix, accuracy_score,precision_score,recall_score
classifier = Sequential()
classifier.add(Dense(units = 6, kernel_initializer = &#39;uniform&#39;, activation = &#39;relu&#39;,
input_dim = 16))
classifier.add(Dense(units = 6, kernel_initializer = &#39;uniform&#39;, activation = &#39;relu&#39;))
classifier.add(Dense(units = 1, kernel_initializer = &#39;uniform&#39;, activation =
&#39;sigmoid&#39;))
classifier.compile(optimizer = &#39;adam&#39;, loss = &#39;binary_crossentropy&#39;, metrics =
[&#39;accuracy&#39;])
Y_train = y_train
Y_test = y_test
y_train = y_train.astype(&#39;category&#39;)
y_train = y_train.cat.codes
y_test = y_test.astype(&#39;category&#39;)
y_test = y_test.cat.codes
classifier.fit(x_train_lsso, y_train, batch_size = 10, epochs = 100)
y_pred = classifier.predict(x_test_lsso)
y_pred = (y_pred &gt; 0.5)
accuracy_ann_lsso = accuracy_score(y_test,y_pred)*100
precision_ann_lsso = precision_score(y_test,y_pred)*100
recall_ann_lsso = recall_score(y_test,y_pred)*100
print(&quot;Accuracy :&quot;,accuracy_ann_lsso)
print(&quot;Precision&quot;,precision_ann_lsso)
print(&quot;Recall :&quot;,recall_ann_lsso)
error_ann_lsso = 100 - accuracy_ann_lsso
print(&quot;Error : &quot;,error_ann_lsso)
fmes_ann_lsso = f1_score(y_test,y_pred,zero_division =1)
print(&quot;F1 Score : &quot;,fmes_ann_lsso)
```

```
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_pred))
auc_ann_lsso = roc_auc_score(y_test, y_pred)*100
print(&quot;AUC : &quot;,auc_ann_lsso)
gini_ann_lsso = 2*(auc_ann_lsso/100) - 1
print(&quot;GINI : &quot;,gini_ann_lsso)
# CHAID
from chefboost import Chefboost as cf
from pandas.api.types import CategoricalDtype
config = [&quot;algorithm&quot; : &quot;CHAID&quot;]
df = pd.DataFrame()
df = x1_train_lsso
ChaidTree = cf.fit(df.join(y1_train),config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test.index:
pred = cf.predict(ChaidTree,x1_test_lsso.loc[i, :])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
```

```
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
accuracy_chaid_lsso = (tp+tn)*100/(tp+tn+fp+fn)
precision_chaid_lsso = tp*100/(tp+fp)
recall_chaid_lsso = tp*100/(tp+fn)
print(accuracy_chaid_lsso)
print(precision_chaid_lsso)
print(recall_chaid_lsso)
error_chaid_lsso = 100 - accuracy_chaid_lsso
print(&quot;Error : &quot;,error_chaid_lsso)
fmes_chaid_lsso =
2*(precision_chaid_lsso*recall_chaid_lsso)/((precision_chaid_lsso+recall_chaid_lsso)*100)
print(fmes_chaid_lsso)
auc_chaid_lsso = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_chaid_lsso)
gini_chaid_lsso = 2*(auc_chaid_lsso/100) - 1
print(&quot;GINI : &quot;,gini_chaid_lsso)
# C4.5
config = [&quot;algorithm&quot; : &quot;C4.5&quot;]
C45Tree = cf.fit(df.join(y1_train), config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
```

```
y_testing = []
for i in x1_test_corr.index:
pred = cf.predict(C45Tree,x1_test_lsso.loc[i, :])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
accuracy_c45_lsso = (tp+tn)*100/(tp+tn+fp+fn)
print(accuracy_c45_lsso)
precision_c45_lsso = tp*100/(tp+fp)
recall_c45_lsso = tp*100/(tp+fn)
print(precision_c45_lsso)
print(recall_c45_lsso )
error_c45_lsso = 100 - accuracy_c45_lsso
print(&quot;Error : &quot;,error_c45_lsso)
fmes_c45_lsso =
2*(precision_c45_lsso*recall_c45_lsso)/((precision_c45_lsso+recall_c45_lsso)*100)
```

```
print(fmes_c45_lsso)
auc_c45_lsso = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_c45_lsso)
gini_c45_lsso = 2*(auc_c45_lsso/100) - 1
print(&quot;GINI : &quot;,gini_c45_lsso)
# Evaluation Measures Plotting in case of LASSO FS
fig2 = plt.subplots(figsize =(12, 8))
accuracy = [accuracy_lr_lsso,accuracy_knn_lsso,accuracy_lsvml1_lsso
,accuracy_lsvml2_lsso,accuracy_ann_lsso,accuracy_chaid_lsso,accuracy_c45_lsso,accuracy_rf
_lsso]
precision = [precison_lr_lsso,precison_knn_lsso,precison_lsvml1_lsso,precison_lsvml2_lsso,
precision_ann_lsso,precision_chaid_lsso,precision_c45_lsso,precision_rf_lsso ]
recall = [recall_lr_lsso, recall_knn_lsso, recall_lsvml1_lsso,recall_lsvml2_lsso,
recall_ann_lsso,recall_chaid_lsso, recall_c45_lsso,recall_rf_lsso ]
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br1, accuracy, color =&#39;r&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Accuracy&#39;)
plt.bar(br2, precision,color =&#39;g&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Precision&#39;)
plt.bar(br3, recall, color =&#39;b&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Recall&#39;)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Performance&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
[&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;ANN&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;RF&#39; ] )
plt.title(&quot;Accuracy Precision Recall for Lasso feature selection&quot;)
plt.legend()
plt.show()
```

```
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
f_mesures = [fmes_lr_lsso, fmes_knn_lsso,
fmes_lsvml1_lsso,fmes_lsvml2_lsso,fmes_chaid_lsso, fmes_c45_lsso,
fmes_ann_lsso,fmes_rf_lsso]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;F Measure&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of F Measure for Various Classifiers using Lasso Feature selection&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
auc = [auc_lr_lsso, auc_knn_lsso, auc_lsvml1_lsso,auc_lsvml2_lsso,auc_chaid_lsso,
auc_c45_lsso, auc_ann_lsso,auc_rf_lsso]
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;AUC&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of AUC for Various Classifiers using Lasso Feature selection&quot;)
plt.show()
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
gini = [gini_lr_lsso, gini_knn_lsso, gini_lsvml1_lsso,gini_lsvml2_lsso,gini_chaid_lsso,
gini_c45_lsso, gini_ann_lsso,gini_rf_lsso]
plt.rcParams[&quot;figure.figsize&quot;] = (30,10)
plt.plot(algo, gini)
plt.xlabel(&quot;Algorithm&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.ylabel(&quot;GINI&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
```

```
plt.title(&quot;Variation of GINI for Various Classifiers using Lasso Feature selection&quot;,
fontweight
=&#39;bold&#39;, fontsize = 25)
plt.show()
error = plt.figure()
x = error.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;KNN&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;,
&#39;CHAID&#39;, &#39;C4.5&#39;,&#39;ANN&#39;,&#39;RF&#39;]
auc = [100-accuracy_lr_lsso,100-accuracy_knn_lsso,100-accuracy_lsvml1_lsso ,100-
accuracy_lsvml2_lsso,100-accuracy_chaid_lsso,100-accuracy_c45_lsso,100-
accuracy_ann_lsso,100-accuracy_rf_lsso]
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Error&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of Error for Various Classifiers using Lasso Feature selection&quot;)
plt.show()
# SMOTE
Data.head()
Data
y_train
np.bincount(y_train)
np.bincount(y_test)
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
x1_smote_train, y1_smote_train = sm.fit_resample(x1_train,y1_train)
x_smote_train, y_smote_train = sm.fit_resample(x_train, y_train)
np.bincount(y_smote_train)
x_smote_train
# Full features with SMOTE
sns.heatmap(x_smote_train.corr(),annot = True)
# LR
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import precision_score, accuracy_score, recall_score
import sys
lr = LogisticRegression(solver = &#39;lbfgs&#39;, max_iter = sys.maxsize )
lr.fit(x_smote_train, y_smote_train)
y_predict = lr.predict(x_test)
accuracy_smote_lr = accuracy_score(y_test,y_predict)*100
precison_smote_lr = precision_score(y_test, y_predict,pos_label= 0 )*100
recall_smote_lr = recall_score(y_test, y_predict,pos_label=0)*100
fmes_smote_lr = f1_score(y_test,y_predict,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_smote_lr)
print(&quot;Precision&quot;,precison_smote_lr)
print(&quot;Recall :&quot;,recall_smote_lr )
error_smote_lr = 100 - accuracy_smote_lr
print(&quot;Error : &quot;,error_smote_lr)
print(&quot;F1 Score : &quot;,fmes_smote_lr)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
auc_smote_lr = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_lr)
gini_smote_lr = 2*(auc_smote_lr/100) - 1
print(&quot;GINI : &quot;,gini_smote_lr)
# LSVM L1
from sklearn.svm import LinearSVC
lsvmL1 = LinearSVC(dual = False, penalty = &#39;l1&#39;,max_iter = 10000)
lsvmL1.fit(x_smote_train, y_smote_train)
y_predict = lsvmL1.predict(x_test)
accuracy_smote_lsvmL1 = accuracy_score(y_test,y_predict)*100
precison_smote_lsvmL1 = precision_score(y_test, y_predict_lsso,pos_label=0 )*100
recall_smote_lsvmL1 = recall_score(y_test, y_predict_lsso,pos_label=0)*100
```

```
fmes_smote_lsvmL1 = f1_score(y_test,y_predict_lsso,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_smote_lsvmL1)
print(&quot;Precision&quot;,precison_smote_lsvmL1)
print(&quot;Recall :&quot;,recall_smote_lsvmL1) #max_iter = sys.maxsize
error_smote_lsvmL1 = 100 - accuracy_smote_lsvmL1
print(&quot;Error : &quot;,error_smote_lsvmL1)
print(&quot;F1 Score : &quot;,fmes_smote_lsvmL1)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
auc_smote_lsvmL1 = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_lsvmL1)
gini_smote_lsvmL1 = 2*(auc_smote_lsvmL1/100) - 1
print(&quot;GINI : &quot;,gini_smote_lsvmL1)
# LSVM L2
from sklearn.svm import LinearSVC
import sys
lsvmL2 = LinearSVC( dual=False,penalty=&#39;l2&#39;,max_iter = 120000)
lsvmL2.fit(x_smote_train, y_smote_train)
y_predict = lsvmL2.predict(x_test)
accuracy_smote_lsvmL2 = accuracy_score(y_test,y_predict)*100
precison_smote_lsvmL2 = precision_score(y_test, y_predict,pos_label=0 )*100
recall_smote_lsvmL2 = recall_score(y_test, y_predict,pos_label=0)*100
fmes_smote_lsvmL2 = f1_score(y_test,y_predict,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_smote_lsvmL2)
print(&quot;Precision&quot;,precison_smote_lsvmL2)
print(&quot;Recall :&quot;,recall_smote_lsvmL2)
error_smote_lsvmL2 = 100 - accuracy_smote_lsvmL2
print(&quot;Error : &quot;,error_smote_lsvmL2)
print(&quot;F1 Score : &quot;,fmes_smote_lsvmL2)
```

```
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
auc_smote_lsvmL2 = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_lsvmL2)
gini_smote_lsvmL2 = 2*(auc_smote_lsvmL2/100) - 1
print(&quot;GINI : &quot;,gini_smote_lsvmL2)
# RF
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion=&quot;entropy&quot;)
classifier.fit(x_smote_train, y_smote_train)
y_predict = classifier.predict(x_test)
accuracy_smote_rf = accuracy_score(y_test,y_predict)*100
precision_smote_rf= precision_score(y_test, y_predict,pos_label=0)*100
recall_smote_rf = recall_score(y_test, y_predict,pos_label=0)*100
fmes_smote_rf = f1_score(y_test,y_predict,zero_division =1)
print(&quot;Accuracy :&quot;,accuracy_smote_rf)
print(&quot;Precision&quot;,precision_smote_rf)
print(&quot;Recall :&quot;,recall_smote_rf)
error_smote_rf = 100 - accuracy_smote_rf
print(&quot;Error : &quot;,error_smote_rf)
print(&quot;F1 Score : &quot;,fmes_smote_rf)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y_test))
auc_smote_rf = roc_auc_score(y_test, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_rf)
gini_smote_rf = 2*(auc_smote_rf/100) - 1
print(&quot;GINI : &quot;,gini_smote_rf)
```

```
x1_smote_train
df1 = pd.DataFrame()
df1 = x1_smote_train
y1_smote_train.rename = &quot;Decision&quot;
df1 = df1.join(y1_smote_train)
df1
df1.info()
df1[&quot;Hemo&quot;] = pd.to_numeric(df1.Hemo, errors = &#39;coerce&#39;)
print(df1.info())
y1_test
x1_test
# CHAID
from chefboost import Chefboost as cf
from pandas.api.types import CategoricalDtype
config = {&quot;algorithm&quot; : &quot;CHAID&quot;}
ChaidTree = cf.fit(df1,config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test.index:
pred = cf.predict(ChaidTree,x1_test.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
tn += 1
```

```
y_predict.append(0)
y_testing.append(0)
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
print(&quot;TP : &quot;+str(tp))
print(&quot;TN : &quot;+str(tn))
print(&quot;FP : &quot;+str(fp))
print(&quot;FN : &quot;+str(fn))
accuracy_smote_chaid = (tp+tn)*100/(tp+tn+fp+fn)
precision_smote_chaid = tp*100/(tp+fp)
recall_smote_chaid = tp*100/(tp+fn)
print(accuracy_smote_chaid)
print(precision_smote_chaid)
print(recall_smote_chaid)
error_smote_chaid = 100 - accuracy_smote_chaid
print(&quot;Error : &quot;,error_smote_chaid)
fmes_smote_chaid =
2*(recall_smote_chaid*precision_smote_chaid)/(recall_smote_chaid+precision_smote_chaid)*
100)
print(&quot;F MEasure : &quot;,fmes_smote_chaid)
auc_smote_chaid = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_chaid)
gini_smote_chaid = 2*(auc_smote_chaid/100) - 1
print(&quot;GINI : &quot;,gini_smote_chaid)
```

```
# C4.5
from chefboost import Chefboost as cf
from pandas.api.types import CategoricalDtype
config = {&quot;algorithm&quot; : &quot;C4.5&quot;}
C45Tree = cf.fit(df1, config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test.index:
pred = cf.predict(C45Tree,x1_test.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
```

```
print(&quot;TP : &quot;+str(tp))
print(&quot;TN : &quot;+str(tn))
print(&quot;FP : &quot;+str(fp))
print(&quot;FN : &quot;+str(fn))
accuracy_smote_c45 = (tp+tn)*100/(tp+tn+fp+fn)
print(accuracy_smote_c45)
precision_smote_c45 = tp*100/(tp+fp)
recall_smote_c45 = tp*100/(tp+fn)
print(precision_smote_c45)
print(recall_smote_c45 )
error_smote_c45 = 100 - accuracy_smote_c45
print(&quot;Error : &quot;,error_smote_c45)
fmes_smote_c45 =
2*(recall_smote_c45*precision_smote_c45)/(recall_smote_c45+precision_smote_c45)*100)
print(&quot;F MEasure : &quot;,fmes_smote_c45)
auc_smote_c45 = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_c45)
gini_smote_c45 = 2*(auc_smote_c45/100) - 1
print(&quot;GINI : &quot;,gini_smote_c45)
# Evaluation Measures Plotting in case of Full Features with SMOTE
fig2 = plt.subplots(figsize =(12, 8))
accuracy = [accuracy_smote_lr,accuracy_smote_lsvmL1
,accuracy_smote_lsvmL2,accuracy_smote_rf,accuracy_smote_chaid,accuracy_smote_c45]
precision = [precison_smote_lr,precison_smote_lsvmL1,precison_smote_lsvmL2,
precision_smote_rf,precision_smote_chaid,precision_smote_c45 ]
recall = [recall_smote_lr, recall_smote_lsvmL1,recall_smote_lsvmL2,
recall_smote_rf,recall_smote_chaid, recall_smote_c45]
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br1, accuracy, color =&#39;r&#39;, width = barWidth,
```

```
edgecolor =&#39;grey&#39;, label =&#39;Accuracy&#39;)
plt.bar(br2, precision,color =&#39;g&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Precision&#39;)
plt.bar(br3, recall, color =&#39;b&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Recall&#39;)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Performance&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
[&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;RF&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39; ] )
plt.title(&quot;Accuracy Precision Recall using Smote for Full features&quot;)
plt.legend()
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
f_mesures = [fmes_smote_lr, fmes_smote_lsvmL1,fmes_smote_lsvmL2,fmes_smote_chaid,
fmes_smote_c45, fmes_smote_rf]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;F Measure&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of F Measure for Various Classifiers using Full Features with Smote&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
auc = [auc_smote_lr, auc_smote_lsvmL1,auc_smote_lsvmL2,auc_smote_chaid,
auc_smote_c45, auc_smote_rf]
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
```

```
plt.ylabel(&#39;AUC&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of AUC for Various Classifiers using Full Features with Smote&quot;)
plt.show()
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
gini = [gini_smote_lr, gini_smote_lsvmL1,gini_smote_lsvmL2,gini_smote_chaid,
gini_smote_c45, gini_smote_rf]
plt.rcParams[&quot;figure.figsize&quot;] = [30,10]
plt.plot(algo, gini)
plt.xlabel(&quot;Algorithm&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.ylabel(&quot;GINI&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.title(&quot;Variation of GINI for Various Classifiers using Full Features with Smote&quot;,
fontweight
=&#39;bold&#39;, fontsize = 25)
plt.show()
error = plt.figure()
x = error.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
auc =[100-accuracy_smote_lr,100-accuracy_smote_lsvmL1 ,100-accuracy_smote_lsvmL2,100-
accuracy_smote_chaid,100-accuracy_smote_c45,100-accuracy_smote_rf]
x.bar(algo,auc)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Error&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of Error for Various Classifiers using Full Features with Smote&quot;)
plt.show()
# LASSO with SMOTE
x1_smote_train_lsso = x1_smote_train.drop(list_lasso, axis = 1)
x1_train_lsso = x1_train.drop(list_lasso, axis = 1)
x_test_lsso = x_test.drop(list_lasso, axis = 1)
x1_test_lsso = x1_test.drop(list_lasso, axis = 1)
x1_smote_train_lsso
```

```
sns.heatmap(x1_smote_train_lsso.corr(),annot = True)
# LR
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, accuracy_score, recall_score
import sys
lr = LogisticRegression(solver = &#39;lbfgs&#39;, max_iter = sys.maxsize )
lr.fit(x1_smote_train_lsso, y1_smote_train)
y_predict = lr.predict(x1_test_lsso)
accuracy_smote_lr_lsso = accuracy_score(y1_test,y_predict)*100
precison_smote_lr_lsso = precision_score(y1_test, y_predict,pos_label= &#39;ckd&#39; )*100
recall_smote_lr_lsso = recall_score(y1_test, y_predict,pos_label=&#39;ckd&#39;)*100
fmes_smote_lr_lsso = f1_score(y1_test,y_predict, pos_label=&#39;ckd&#39;)
print(&quot;Accuracy :&quot;,accuracy_smote_lr_lsso)
print(&quot;Precision&quot;,precison_smote_lr_lsso)
print(&quot;Recall :&quot;,recall_smote_lr_lsso )
error_smote_lr_lsso = 100 - accuracy_smote_lr_lsso
print(&quot;Error : &quot;,error_smote_lr_lsso)
print(&quot;F1 Score : &quot;,fmes_smote_lr_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y1_test,y_predict))
print(&quot;Correct Predictions :&quot;,sum(y_predict == y1_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict != y1_test))
print(y_predict.dtype)
y2_predict = pd.Series(y_predict,dtype = &quot;category&quot;)
y2_predict = y2_predict.cat.codes
auc_smote_lr_lsso = roc_auc_score(y1_test, y2_predict)*100
print(&quot;AUC : &quot;,auc_smote_lr_lsso)
gini_smote_lr_lsso = 2*(auc_smote_lr_lsso/100) - 1
print(&quot;GINI : &quot;,gini_smote_lr_lsso)
# LSVM L1
from sklearn.svm import LinearSVC
```

```
lsvmL1 = LinearSVC(dual = False, penalty = &#39;l1&#39;,max_iter = 10000)
lsvmL1.fit(x1_smote_train_lsso, y1_smote_train)
y_predict_lsso = lsvmL1.predict(x1_test_lsso)
accuracy_smote_lsvmL1_lsso = accuracy_score(y1_test,y_predict_lsso)*100
precison_smote_lsvmL1_lsso= precision_score(y1_test, y_predict_lsso,pos_label=&#39;ckd&#39;
)*100
recall_smote_lsvmL1_lsso = recall_score(y1_test, y_predict_lsso,pos_label=&#39;ckd&#39;)*100
fmes_smote_lsvmL1_lsso = f1_score(y1_test,y_predict_lsso,zero_division
=1,pos_label=&#39;ckd&#39;)
print(&quot;Accuracy :&quot;,accuracy_smote_lsvmL1_lsso)
print(&quot;Precision&quot;,precison_smote_lsvmL1_lsso)
print(&quot;Recall :&quot;,recall_smote_lsvmL1_lsso) #max_iter = sys.maxsize
error_smote_lsvmL1_lsso = 100 - accuracy_smote_lsvmL1_lsso
print(&quot;Error : &quot;,error_smote_lsvmL1_lsso)
print(&quot;F1 Score : &quot;,fmes_smote_lsvmL1_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y1_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y1_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y1_test))
y2_predict = pd.Series(y_predict_lsso,dtype = &quot;category&quot;)
y2_predict = y2_predict.cat.codes
auc_smote_lsvmL1_lsso = roc_auc_score(y1_test, y2_predict)*100
print(&quot;AUC : &quot;,auc_smote_lsvmL1_lsso)
gini_smote_lsvmL1 = 2*(auc_smote_lsvmL1_lsso/100) - 1
print(&quot;GINI : &quot;,gini_smote_lsvmL1)
# LSVM L2
from sklearn.svm import LinearSVC
import sys
lsvmL2 = LinearSVC( dual=False,penalty=&#39;l2&#39;,max_iter = 120000)
lsvmL2.fit(x1_smote_train_lsso, y1_smote_train)
y_predict_lsso = lsvmL2.predict(x1_test_lsso)
accuracy_smote_lsvmL2_lsso = accuracy_score(y1_test,y_predict_lsso)*100
```

```
precison_smote_lsvmL2_lsso = precision_score(y1_test, y_predict_lsso,pos_label=&#39;ckd&#39;
)*100
recall_smote_lsvmL2_lsso = recall_score(y1_test, y_predict_lsso,pos_label=&#39;ckd&#39;)*100
fmes_smote_lsvmL2_lsso = f1_score(y1_test,y_predict_lsso,zero_division
=1,pos_label=&#39;ckd&#39;)
print(&quot;Accuracy :&quot;,accuracy_smote_lsvmL2_lsso)
print(&quot;Precision&quot;,precison_smote_lsvmL2_lsso)
print(&quot;Recall :&quot;,recall_smote_lsvmL2_lsso)
error_smote_lsvmL2_lsso = 100 - accuracy_smote_lsvmL2_lsso
print(&quot;Error : &quot;,error_smote_lsvmL2_lsso)
print(&quot;F1 Score : &quot;,fmes_smote_lsvmL2_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y1_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y1_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y1_test))
y2_predict = pd.Series(y_predict_lsso,dtype = &quot;category&quot;)
y2_predict = y2_predict.cat.codes
auc_smote_lsvmL2_lsso = roc_auc_score(y1_test, y2_predict)*100
print(&quot;AUC : &quot;,auc_smote_lsvmL2_lsso)
gini_smote_lsvmL2 = 2*(auc_smote_lsvmL2_lsso/100) - 1
print(&quot;GINI : &quot;,gini_smote_lsvmL2)
# RF
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion=&quot;entropy&quot;)
classifier.fit(x1_smote_train_lsso, y1_smote_train)
y_predict_lsso= classifier.predict(x1_test_lsso)
accuracy_smote_rf_lsso = accuracy_score(y1_test,y_predict_lsso)*100
precision_smote_rf_lsso= precision_score(y1_test, y_predict_lsso,pos_label=&#39;ckd&#39;)*100
recall_smote_rf_lsso = recall_score(y1_test, y_predict_lsso,pos_label=&#39;ckd&#39;)*100
fmes_smote_rf_lsso = f1_score(y1_test,y_predict_lsso,zero_division =1,pos_label=&#39;ckd&#39;)
print(&quot;Accuracy :&quot;,accuracy_smote_rf_lsso)
print(&quot;Precision&quot;,precision_smote_rf_lsso)
```

```
print(&quot;Recall :&quot;,recall_smote_rf_lsso)
error_smote_rf_lsso = 100 - accuracy_smote_rf_lsso
print(&quot;Error : &quot;,error_smote_rf_lsso)
print(&quot;F1 Score : &quot;,fmes_smote_rf_lsso)
print(&quot;Confusion matrix:&quot;)
print(confusion_matrix(y1_test,y_predict_lsso))
print(&quot;Correct Predictions :&quot;,sum(y_predict_lsso == y1_test))
print(&quot;Incorrect Predictions :&quot;,sum(y_predict_lsso != y1_test))
y2_predict = pd.Series(y_predict_lsso,dtype = &quot;category&quot;)
y2_predict = y2_predict.cat.codes
auc_smote_rf_lsso = roc_auc_score(y1_test, y2_predict)*100
print(&quot;AUC : &quot;,auc_smote_rf_lsso)
gini_smote_rf = 2*(auc_smote_rf/100) - 1
print(&quot;GINI : &quot;,gini_smote_rf)
# CHAID
from chefboost import Chefboost as cf
from pandas.api.types import CategoricalDtype
config = {&quot;algorithm&quot; : &quot;CHAID&quot;}
df = pd.DataFrame()
df = x1_smote_train_lsso
df = df.join(y1_smote_train)
df
ChaidTree = cf.fit(df,config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test.index:
pred = cf.predict(ChaidTree,x1_test_lsso.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fp+=1
else:
y_predict.append(0)
y_testing.append(1)
fn+=1
print(&quot;TP : &quot;+str(tp))
print(&quot;TN : &quot;+str(tn))
print(&quot;FP : &quot;+str(fp))
print(&quot;FN : &quot;+str(fn))
accuracy_smote_chaid_lsso = (tp+tn)*100/(tp+tn+fp+fn)
precision_smote_chaid_lsso = tp*100/(tp+fp)
recall_smote_chaid_lsso = tp*100/(tp+fn)
print(accuracy_smote_chaid_lsso)
print(precision_smote_chaid_lsso)
print(recall_smote_chaid_lsso)
error_smote_chaid_lsso = 100 - accuracy_smote_chaid_lsso
print(&quot;Error : &quot;,error_smote_chaid_lsso)
fmes_smote_chaid_lsso =
2*(precision_smote_chaid_lsso*recall_smote_chaid_lsso)/(precision_smote_chaid_lsso+re
smote_chaid_lsso)
fmes_smote_chaid_lsso/=100.0
print(&quot;F Measure : &quot;,fmes_smote_chaid_lsso)
auc_smote_chaid_lsso = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_chaid_lsso)
gini_smote_chaid = 2*(auc_smote_chaid/100) - 1
print(&quot;GINI : &quot;,gini_smote_chaid)
# C4.5
config = {&quot;algorithm&quot; : &quot;C4.5&quot;}
C45Tree = cf.fit(df, config)
tp = 0
tn = 0
fp = 0
fn = 0
y_predict = []
y_testing = []
for i in x1_test_corr.index:
pred = cf.predict(C45Tree,x1_test_lsso.loc[i,:])
if pred == y1_test[i]:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(1)
tp += 1
else:
y_predict.append(0)
y_testing.append(0)
tn += 1
else:
if pred == &#39;ckd&#39;:
y_predict.append(1)
y_testing.append(0)
fn+=1
print(&quot;TP : &quot;+str(tp))
print(&quot;TN : &quot;+str(tn))
print(&quot;FP : &quot;+str(fp))
print(&quot;FN : &quot;+str(fn))
accuracy_smote_c45_lsso = (tp+tn)*100/(tp+tn+fp+fn)
print(accuracy_smote_c45_lsso)
precision_smote_c45_lsso = tp*100/(tp+fp)
recall_smote_c45_lsso = tp*100/(tp+fn)
print(precision_smote_c45_lsso)
print(recall_smote_c45_lsso )
error_smote_c45_lsso = 100 - accuracy_smote_c45_lsso
print(&quot;Error : &quot;,error_smote_c45_lsso)
fmes_smote_c45_lsso =
2*(precision_smote_c45_lsso*recall_smote_c45_lsso)/((precision_smote_c45_lsso+recall_smot
e_c45_lsso)*100)
print(&quot;F Measure : &quot;,fmes_smote_c45_lsso)
auc_smote_c45_lsso = roc_auc_score(y_testing, y_predict)*100
print(&quot;AUC : &quot;,auc_smote_c45_lsso)
gini_smote_c45 = 2*(auc_smote_c45/100) - 1
print(&quot;GINI : &quot;,gini_smote_c45)
# Evaluation Measures Plotting in case of LASSO with SMOTE
fig2 = plt.subplots(figsize =(12, 8))
accuracy = [accuracy_smote_lr_lsso,accuracy_smote_lsvmL1_lsso
,accuracy_smote_lsvmL2_lsso,accuracy_smote_rf_lsso,accuracy_smote_chaid_lsso,accuracy_s
mote_c45_lsso ]
precision =
[precison_smote_lr_lsso,precison_smote_lsvmL1_lsso,precison_smote_lsvmL2_lsso,
precision_smote_rf_lsso,precision_smote_chaid_lsso,precision_smote_c45_lsso ]
recall = [recall_smote_lr_lsso, recall_smote_lsvmL1_lsso,recall_smote_lsvmL2_lsso,
recall_smote_rf_lsso,recall_smote_chaid_lsso, recall_smote_c45_lsso ]
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br1, accuracy, color =&#39;r&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Accuracy&#39;)
plt.bar(br2, precision,color =&#39;g&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Precision&#39;)
plt.bar(br3, recall, color =&#39;b&#39;, width = barWidth,
edgecolor =&#39;grey&#39;, label =&#39;Recall&#39;)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;Performance&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
[&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;RF&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39; ] )
plt.title(&quot;Accuracy Precision Recall using Smote for Lasso feature selection&quot;)
plt.legend()
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,1,0.3])
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
f_mesures = [fmes_smote_lr_lsso,
fmes_smote_lsvmL1_lsso,fmes_smote_lsvmL2_lsso,fmes_smote_chaid_lsso,
fmes_smote_c45_lsso, fmes_smote_rf_lsso]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;F Measure&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of F Measure for Various Classifiers using Lasso Feature selection with
Smote&quot;)
plt.show()
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
f_mesures = [auc_smote_lr_lsso,
auc_smote_lsvmL1_lsso,auc_smote_lsvmL2_lsso,auc_smote_chaid_lsso, auc_smote_c45_lsso,
auc_smote_rf_lsso]
x.bar(algo,f_mesures)
plt.xlabel(&#39;Algorithm&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.ylabel(&#39;AUC&#39;, fontweight =&#39;bold&#39;, fontsize = 15)
plt.title(&quot;Variation of AUC for Various Classifiers using Lasso Feature selection with
Smote&quot;)
plt.show()
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
gini = [gini_smote_lr_lsso, gini_smote_lsvmL1,gini_smote_lsvmL2,gini_smote_chaid,
gini_smote_c45, gini_smote_rf]
plt.rcParams[&quot;figure.figsize&quot;] = (30,10)
plt.plot(algo, gini)
plt.xlabel(&quot;Algorithm&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.ylabel(&quot;GINI&quot;, fontweight =&#39;bold&#39;, fontsize = 25)
plt.title(&quot;Variation of GINI for Various Classifiers using Lasso Feature selection with
Smote&quot;,
fontweight =&#39;bold&#39;, fontsize = 25)
plt.show()
error = plt.figure()
x = error.add_axes([0,0,0.3,0.3])
algo = [&#39;LR&#39;, &#39;LSVML1&#39;, &#39;LSVML2&#39;, &#39;CHAID&#39;,
&#39;C4.5&#39;,&#39;RF&#39;]
errorures = [100-accuracy_smote_lr_lsso,100-accuracy_smote_lsvmL1_lsso ,100-
```


accuracy_smote_lsvmL2_lsso,100-accuracy_smote_chaid_lsso,100-
accuracy_smote_c45_lsso,100-accuracy_smote_rf_lsso]
x.bar(algo,errorures)
plt.xlabel('Algorithm', fontweight ='bold', fontsize = 15)
plt.ylabel('Error', fontweight ='bold', fontsize = 15)
plt.title("Variation of Error for Various Classifiers using Lasso Feature selection with Smote")
plt.show()

CHAPTER 4

SNAP SHOTS

jupyter Mini Project Last Checkpoint: 2 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

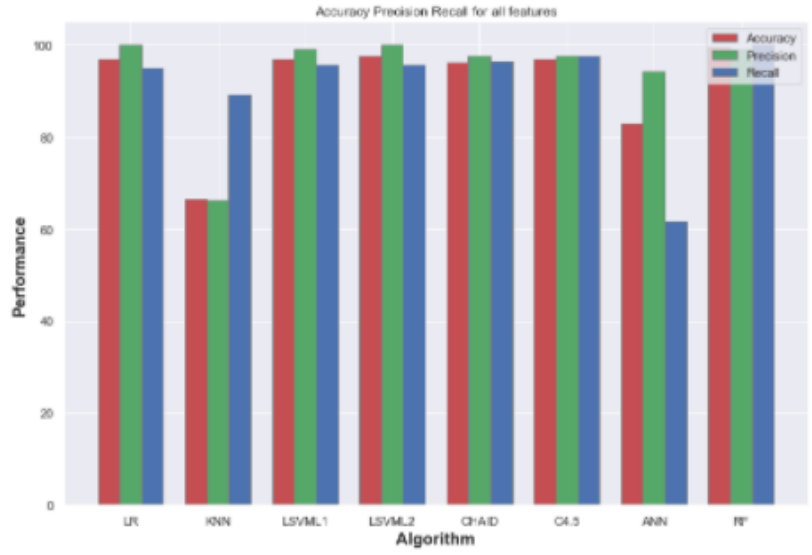
Not Trusted | Python 3

Run Code

```
br1 = np.arange(len(accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]

plt.bar(br1, accuracy, color ='r', width = barWidth,
        edgecolor ='grey', label ='Accuracy')
plt.bar(br2, precision,color ='g', width = barWidth,
        edgecolor ='grey', label ='Precision')
plt.bar(br3, recall, color ='b', width = barWidth,
        edgecolor ='grey', label ='Recall')

# Adding Xticks
plt.xlabel('Algorithm', fontweight ='bold', fontsize = 15)
plt.ylabel('Performance', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(accuracy))],
        ['LR', 'KNN', 'LSVML1', 'LSVML2', 'CHAID', 'C4.5','ANN', 'RF' ] )
plt.title("Accuracy Precision Recall for all features")
plt.legend()
plt.show()
```



In [ ]:

In [33]:
```
f_mes = plt.figure()
x = f_mes.add_axes([0,0,0.3,0.3])
algo = ['LR', 'KNN', 'LSVML1', 'LSVML2', 'CHAID', 'C4.5','ANN','RF']
f_mesures = [fmes_lr, fmes_knn, fmes_lsvmL1,fmes_lsvmL2,fmes_chaid, fmes_c45, fmes_ann, fmes_rf]
x.bar(algo,f_mesures)
plt.xlabel('Algorithm', fontweight ='bold', fontsize = 15)
plt.ylabel('F Measure', fontweight ='bold', fontsize = 15)
plt.title("Variation of F Measure for Various Classifiers using Full Features")
plt.show()
```

Fig. 4.1. Sample Image of Execution of Source code in Jupyter Notebook

Heat Maps:

Here are the heat maps of correlation between selected features in case of Full feature selection, CFS, Wrapper FS, LASSO FS, Full features with SMOTE, LASSO with SMOTE respectively.

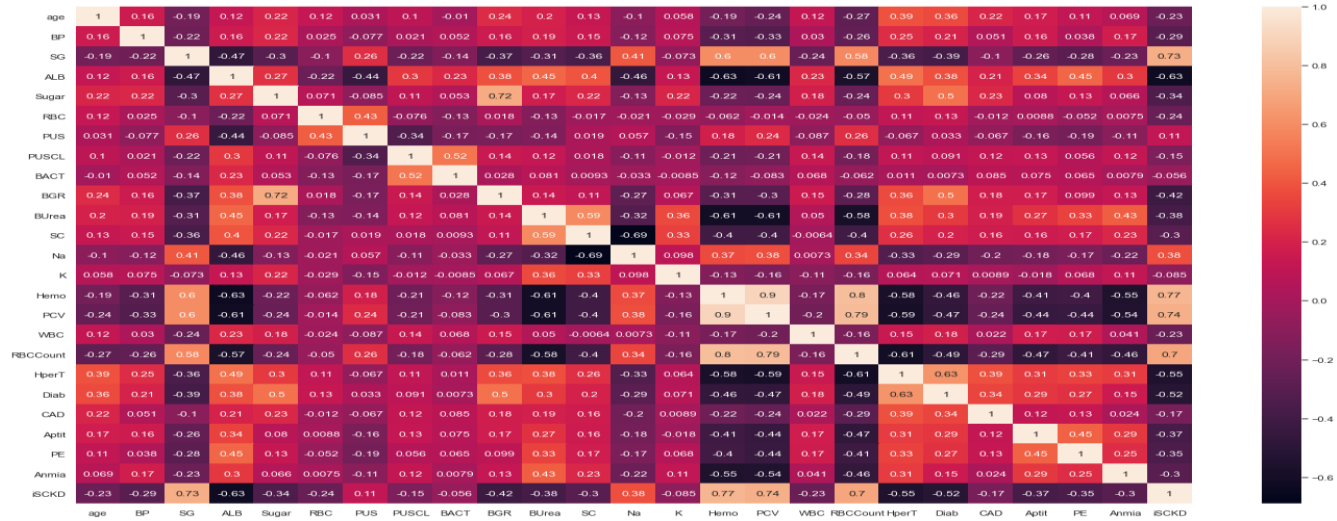


Fig. 4.2. Heat Maps of Correlation between Selected Features in case of Full feature selection

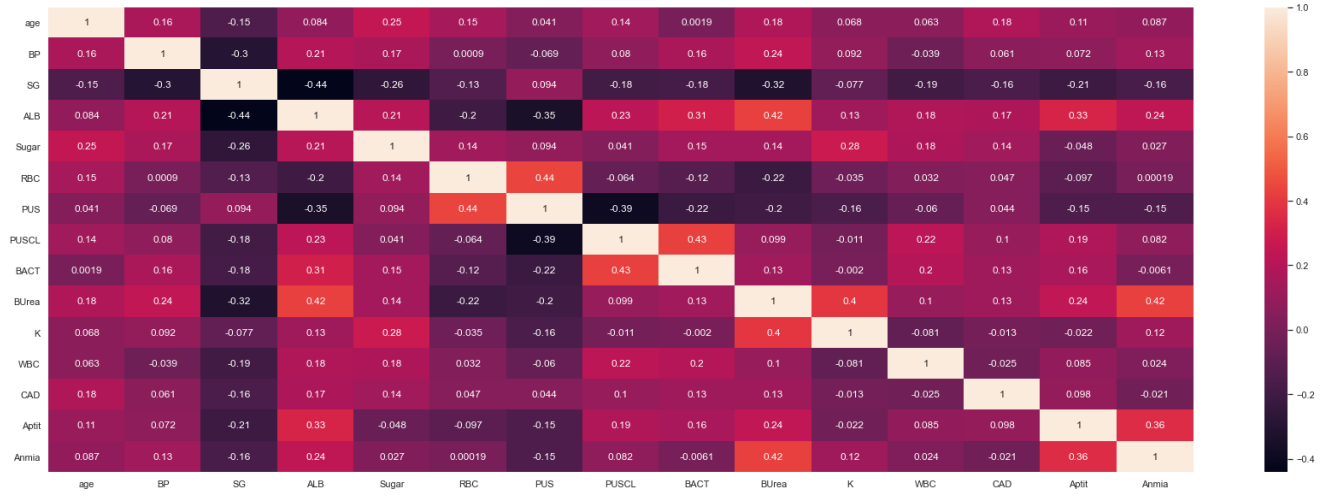


Fig. 4.3. Heat Maps of Correlation between Selected Features in case of CFS

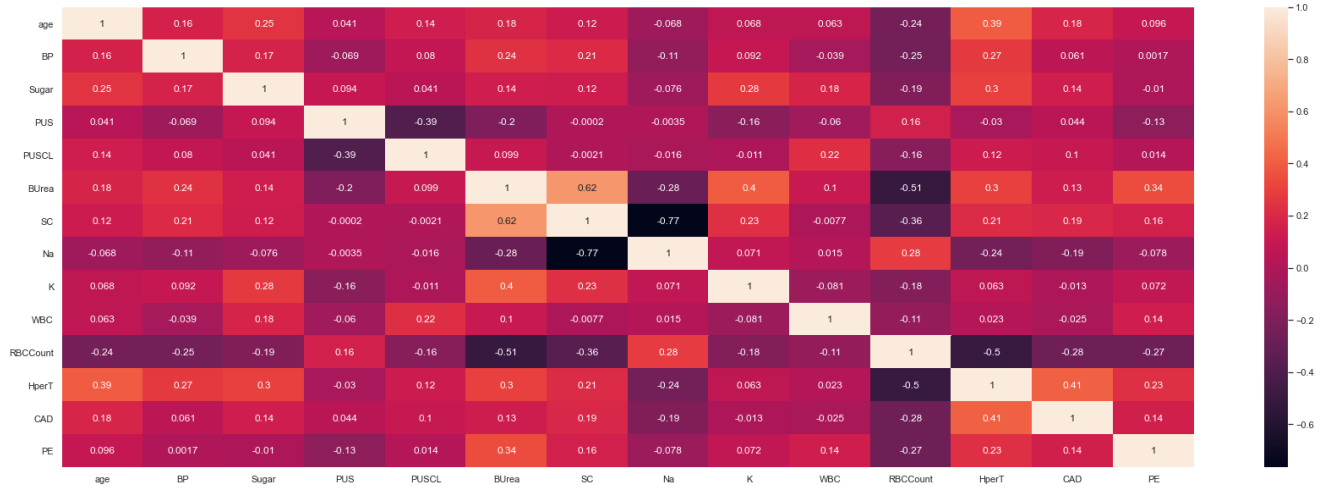


Fig. 4.4. Heat Maps of Correlation between Selected Features in case of Wrapper FS

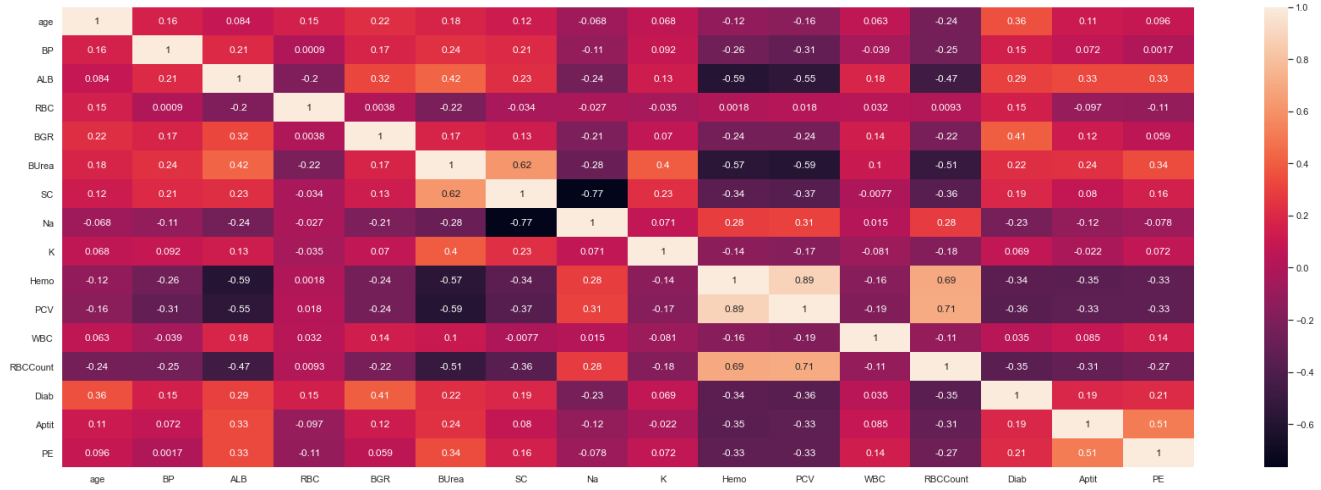


Fig. 4.5. Heat Maps of Correlation between Selected Features in case of LASSO FS

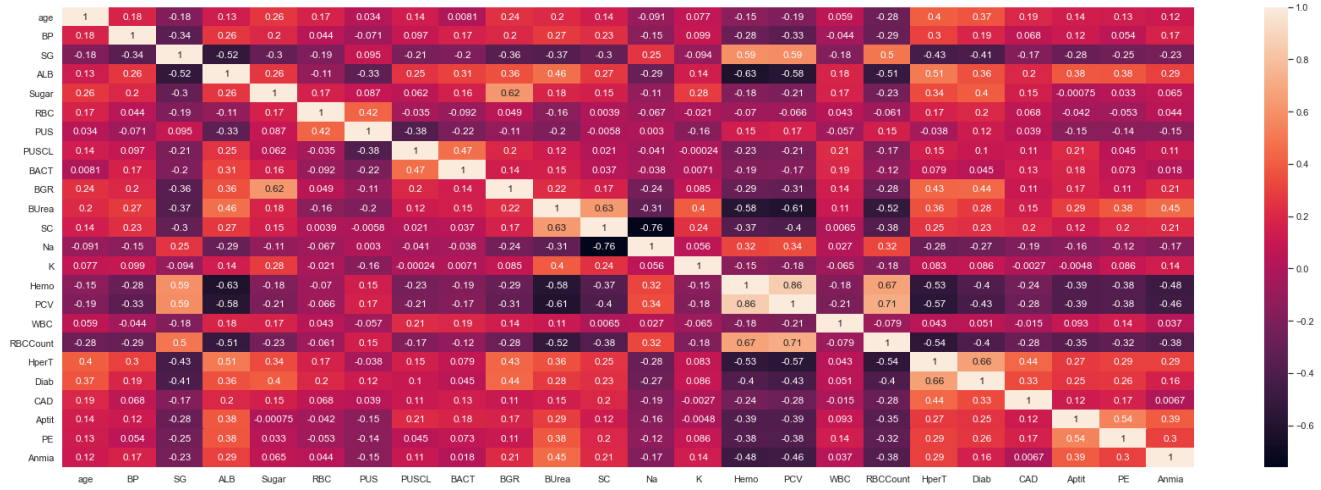


Fig. 4.6. Heat Maps of Correlation between Selected Features in case of Full features with SMOTE

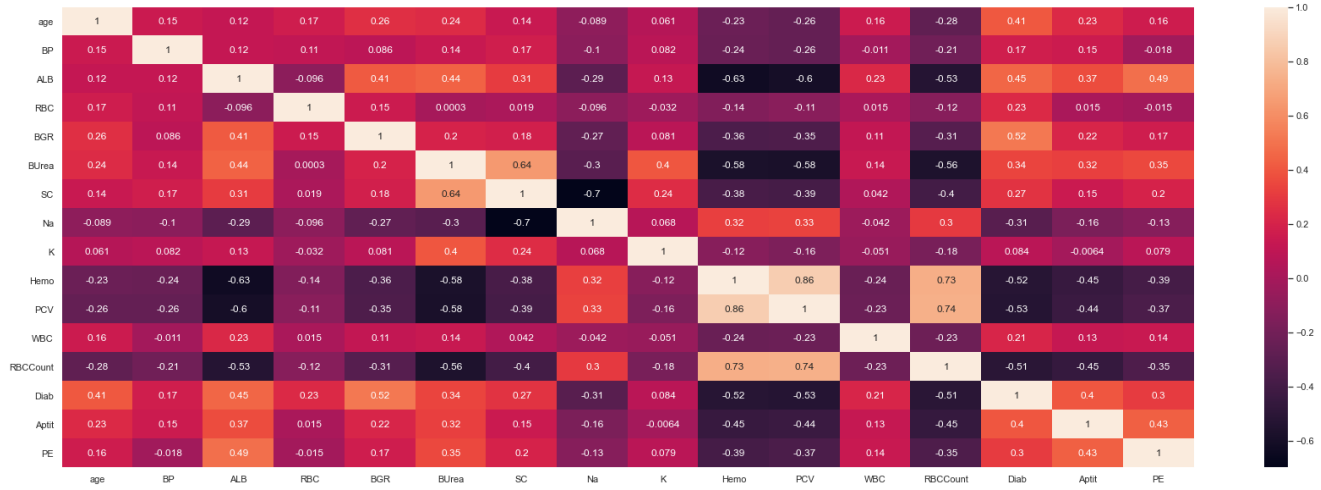


Fig. 4.7. Heat Maps of Correlation between Selected Features in case of LASSO FS with SMOTE

Graphs of Accuracy, Precision, Recall:

Here are the plotted graphs of Accuracy, Precision, Recall respectively for the Machine Learning classifiers in case of Full feature selection, CFS, Wrapper FS, LASSO FS, Full features with SMOTE, LASSO with SMOTE respectively.

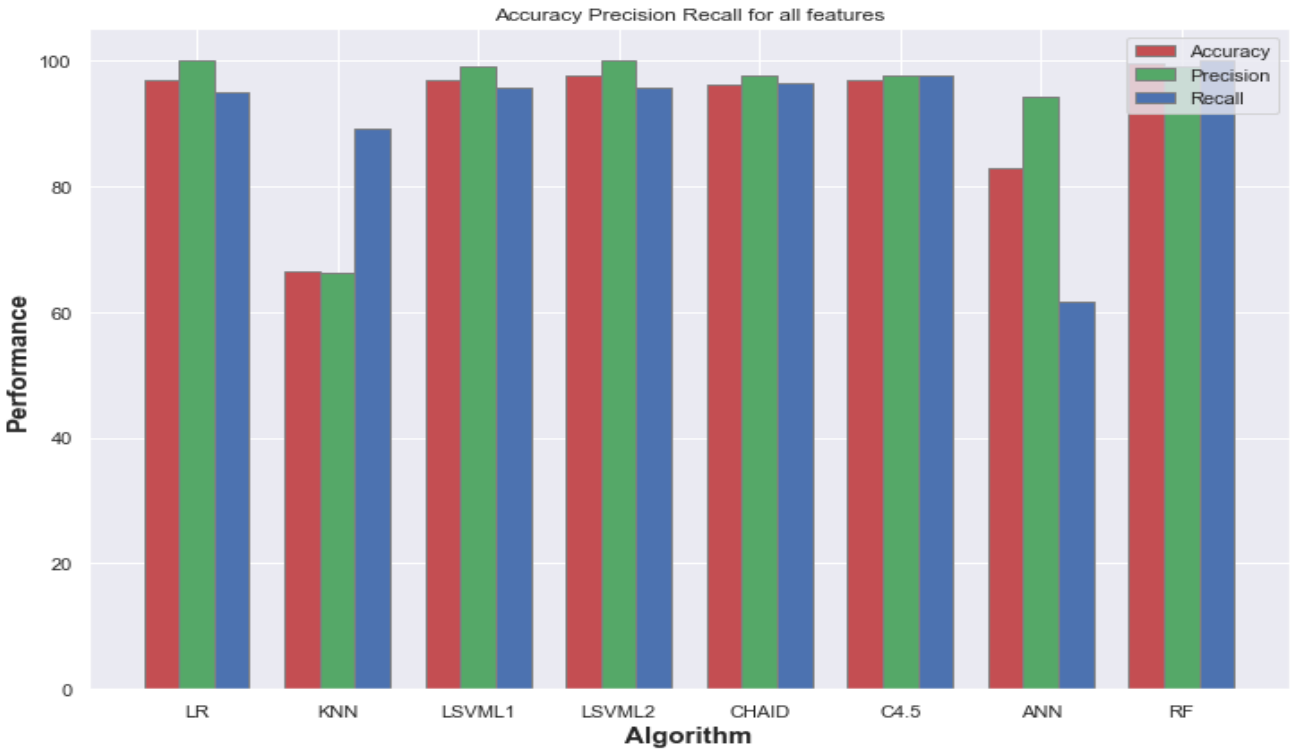


Fig. 4.8. Accuracy, Precision, Recall for ML classifiers in case of Full feature selection

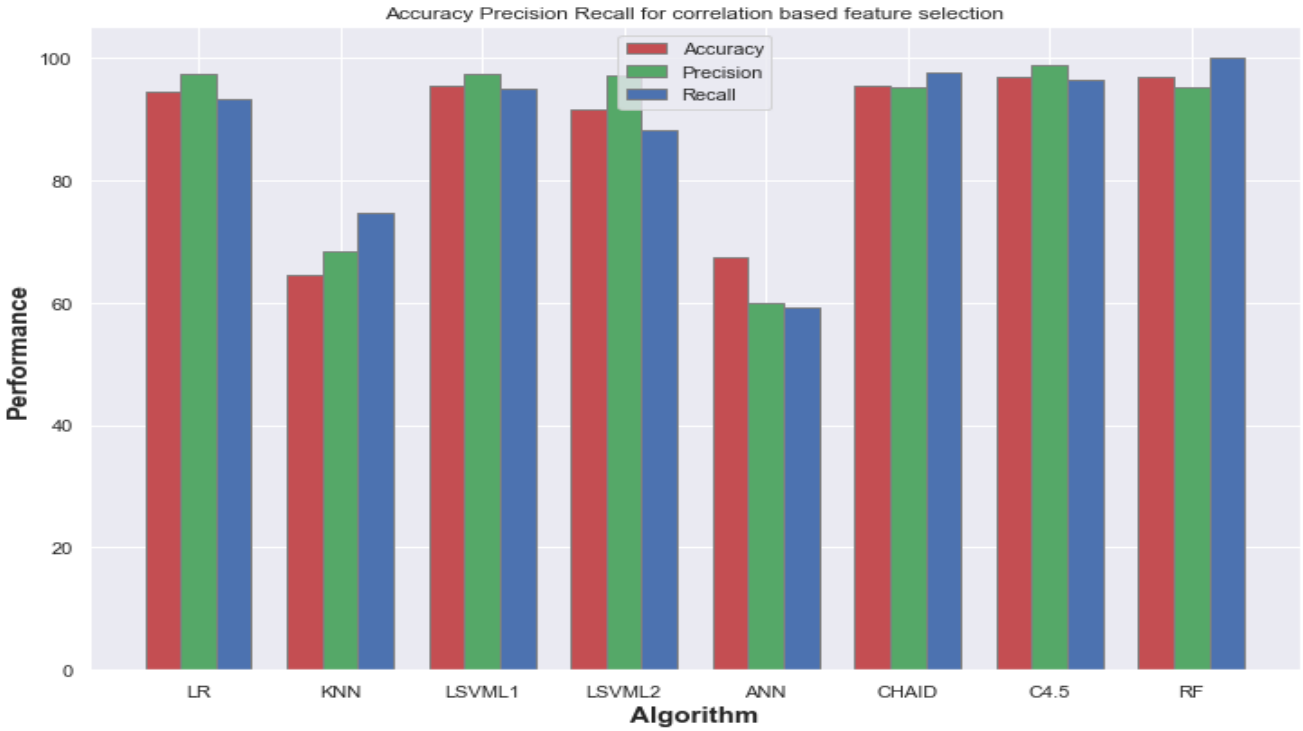


Fig. 4.9. Accuracy, Precision, Recall for ML classifiers in case of CFS

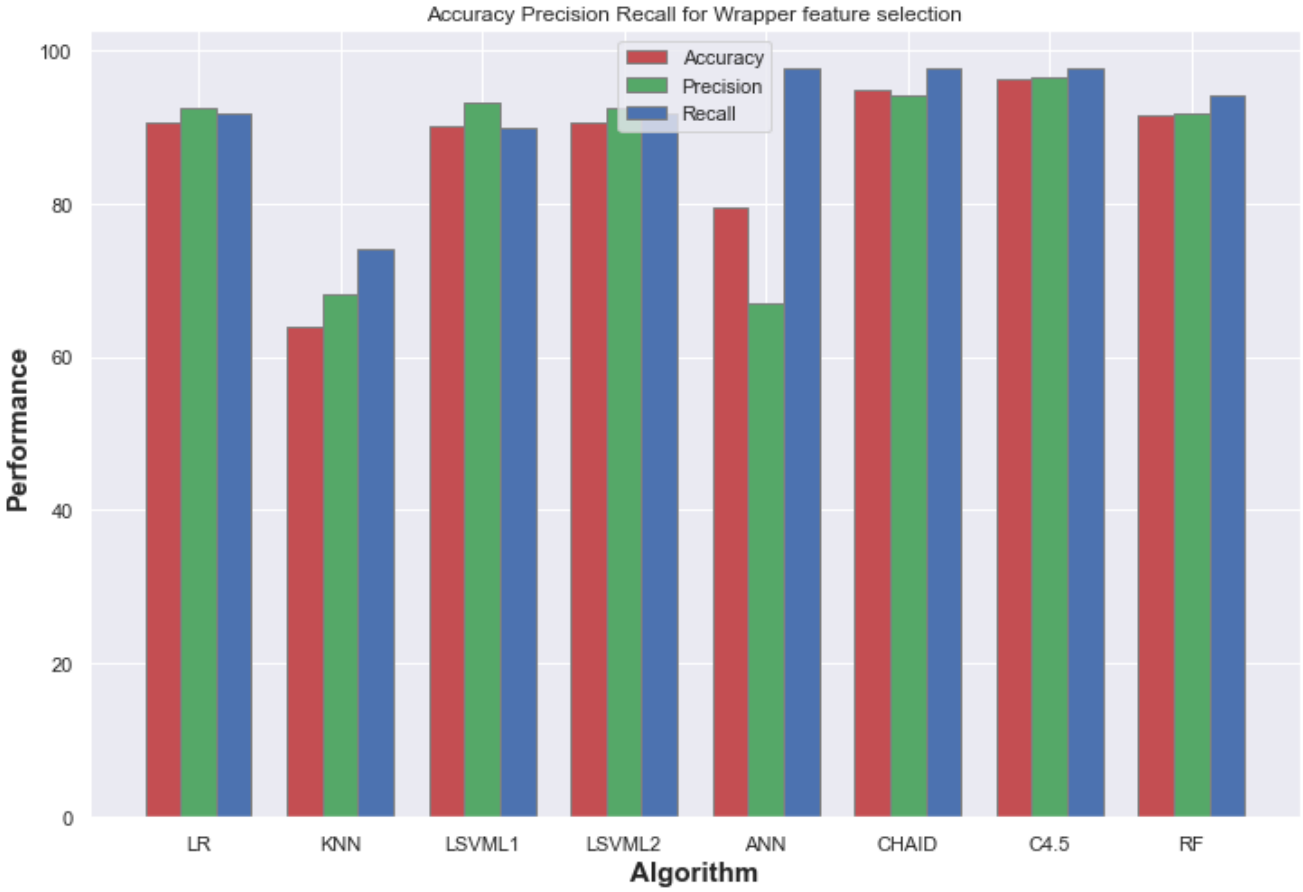


Fig. 4.10. Accuracy, Precision, Recall for ML classifiers in case of Wrapper FS

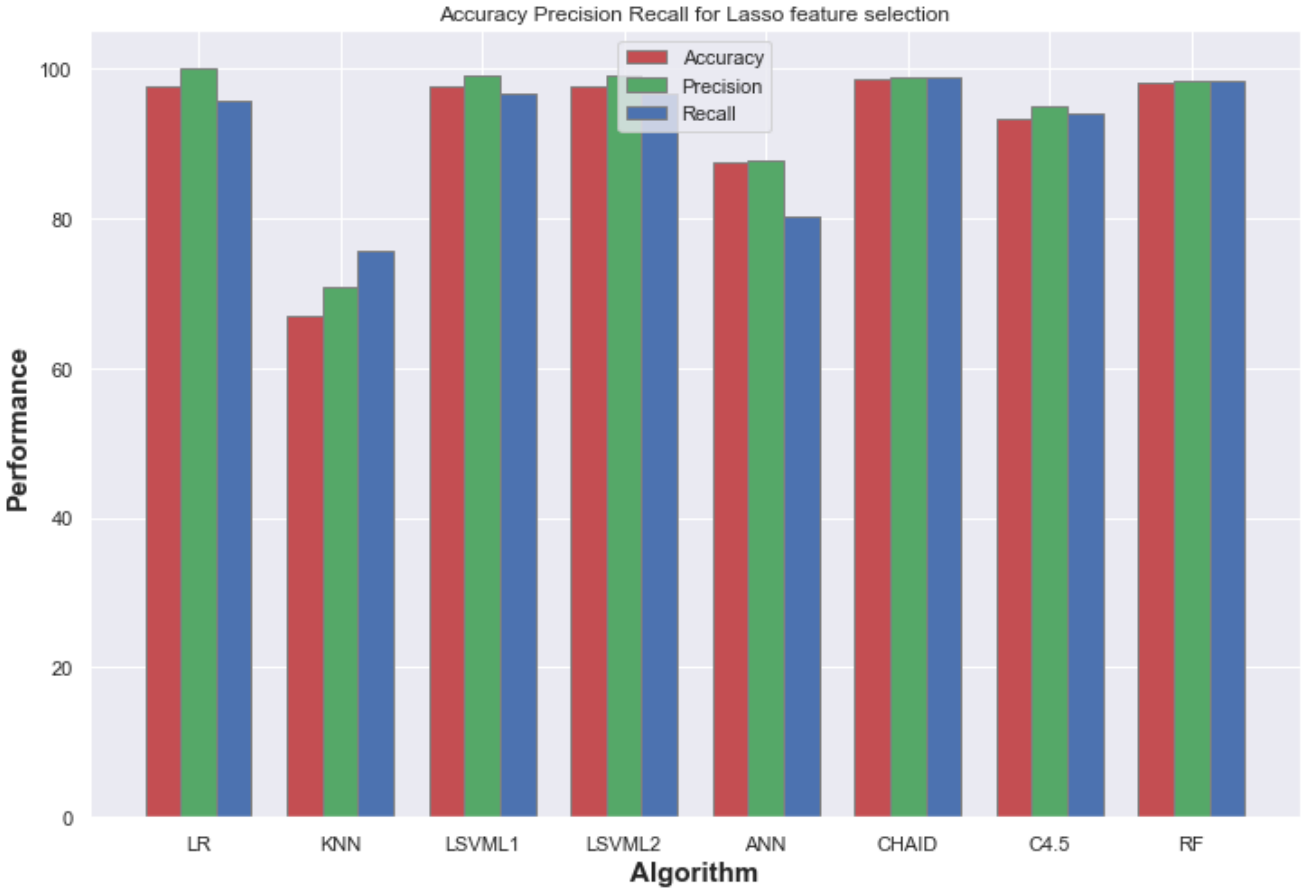


Fig. 4.11. Accuracy, Precision, Recall for ML classifiers in case of LASSO FS

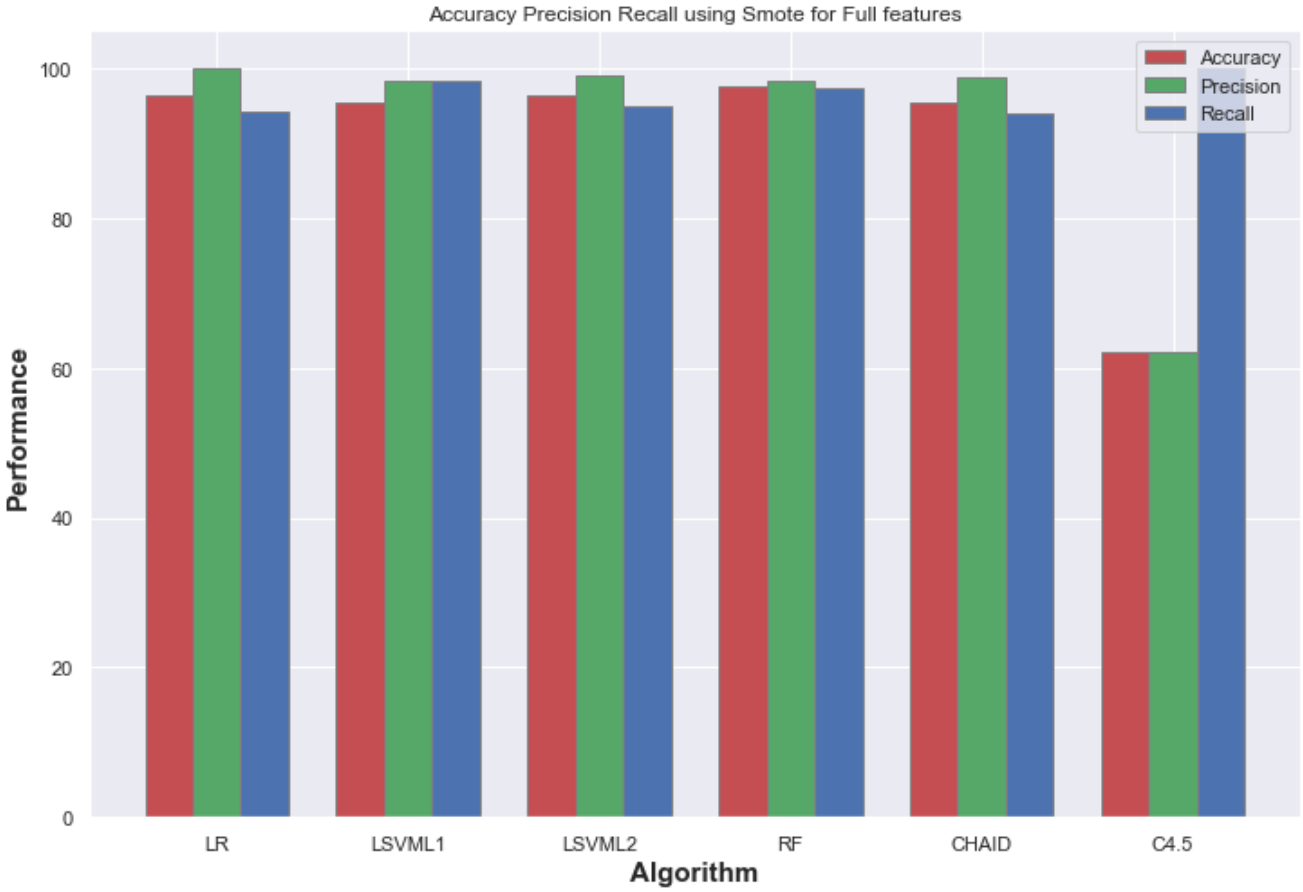


Fig. 4.12. Accuracy, Precision, Recall for ML classifiers in case of Full features with SMOTE

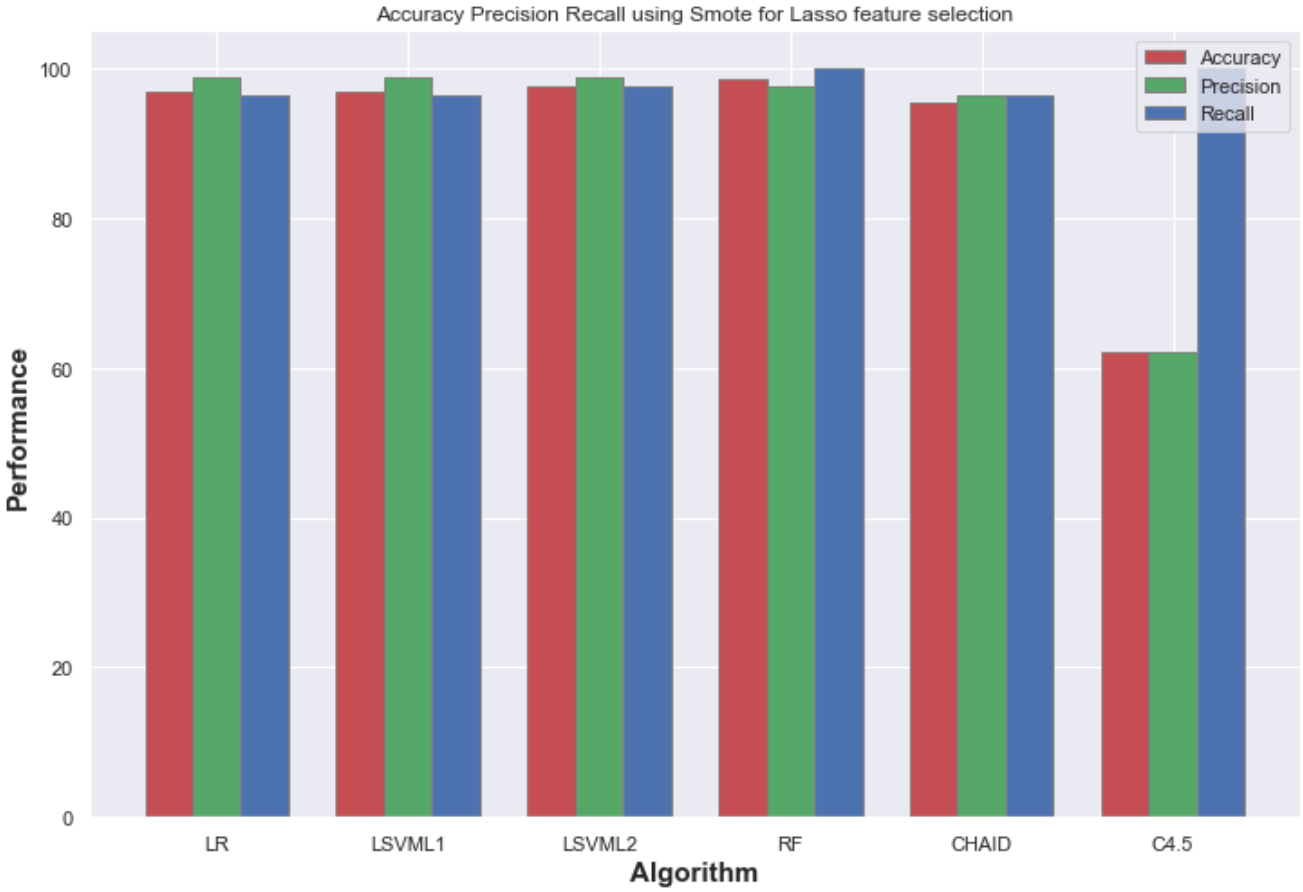


Fig. 4.13. Accuracy, Precision, Recall for ML classifiers in case of LASSO FS with SMOTE

Classification Error:

Here are the plotted graphs of Classification Error for the Machine Learning classifiers in case of Full feature selection, CFS, Wrapper FS, LASSO FS, Full features with SMOTE, LASSO with SMOTE respectively.

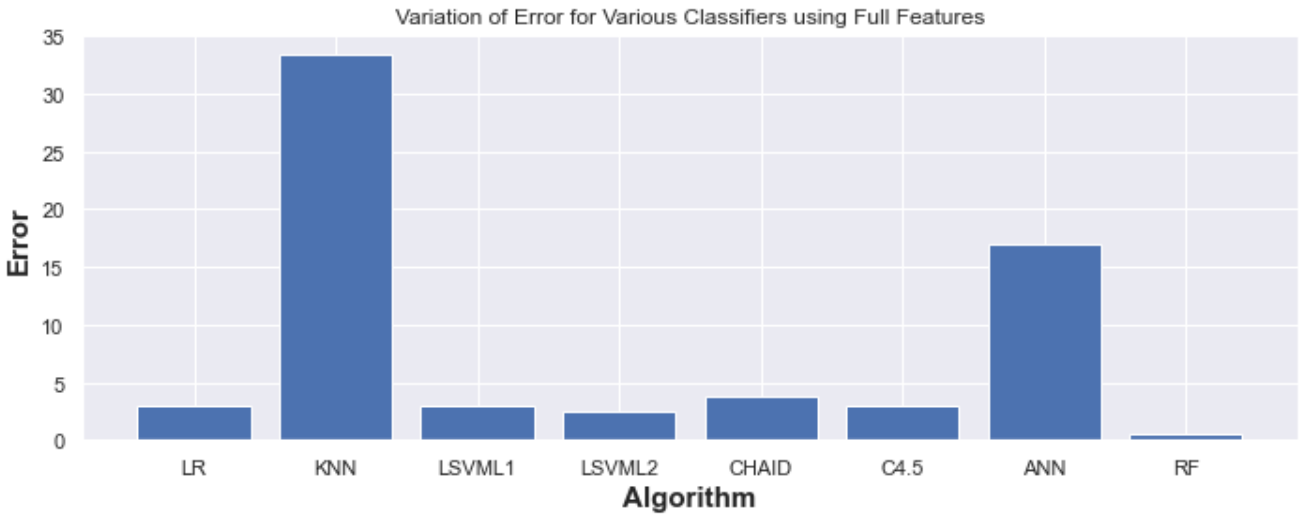


Fig. 4.14. Classification Error for ML classifiers in case of Full feature selection

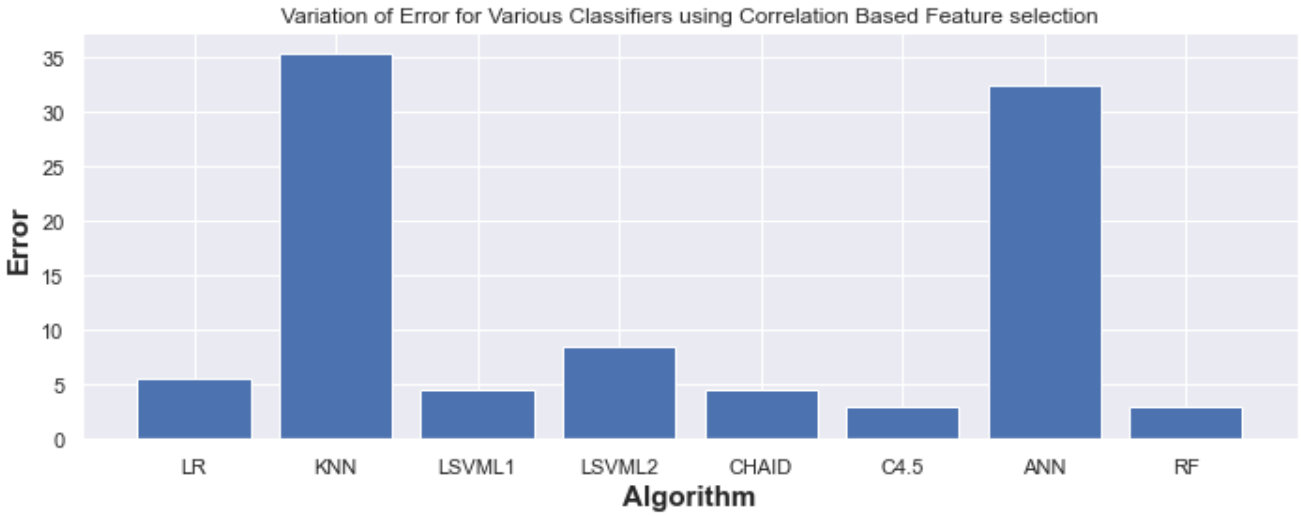


Fig. 4.15. Classification Error for ML classifiers in case of CFS

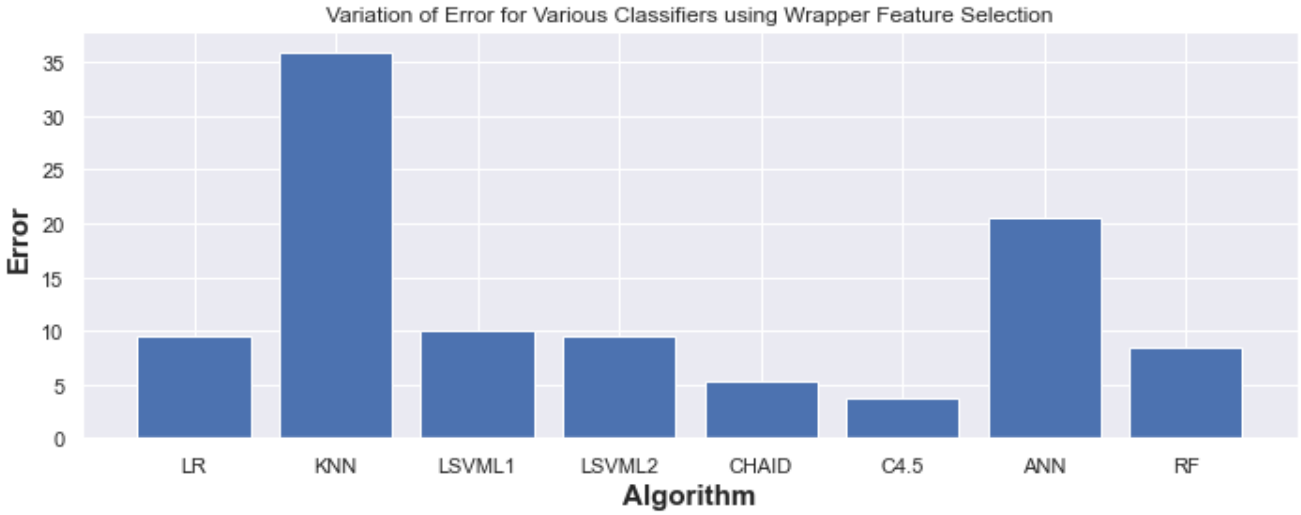


Fig. 4.16. Classification Error for ML classifiers in case of Wrapper FS

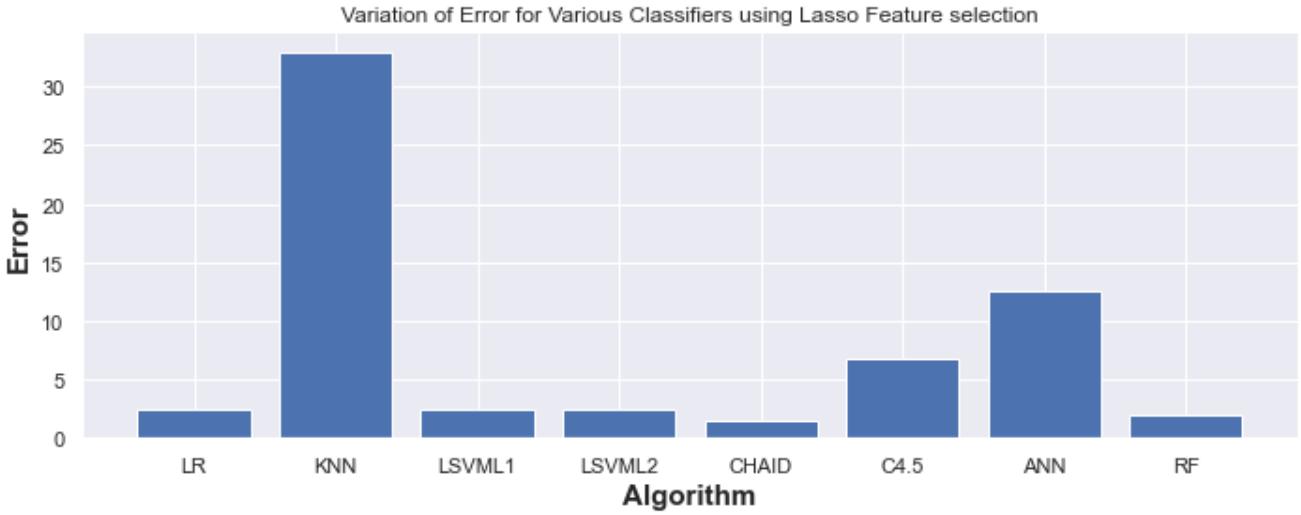


Fig. 4.17. Classification Error for ML classifiers in case of LASSO FS

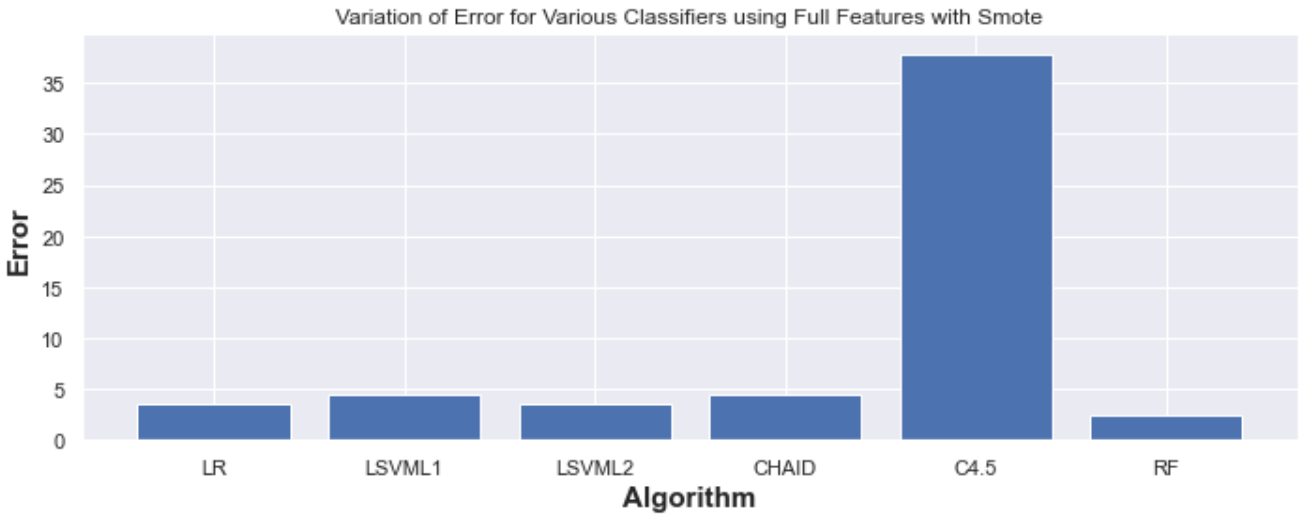


Fig. 4.18. Classification Error for ML classifiers in case of Full features with SMOTE

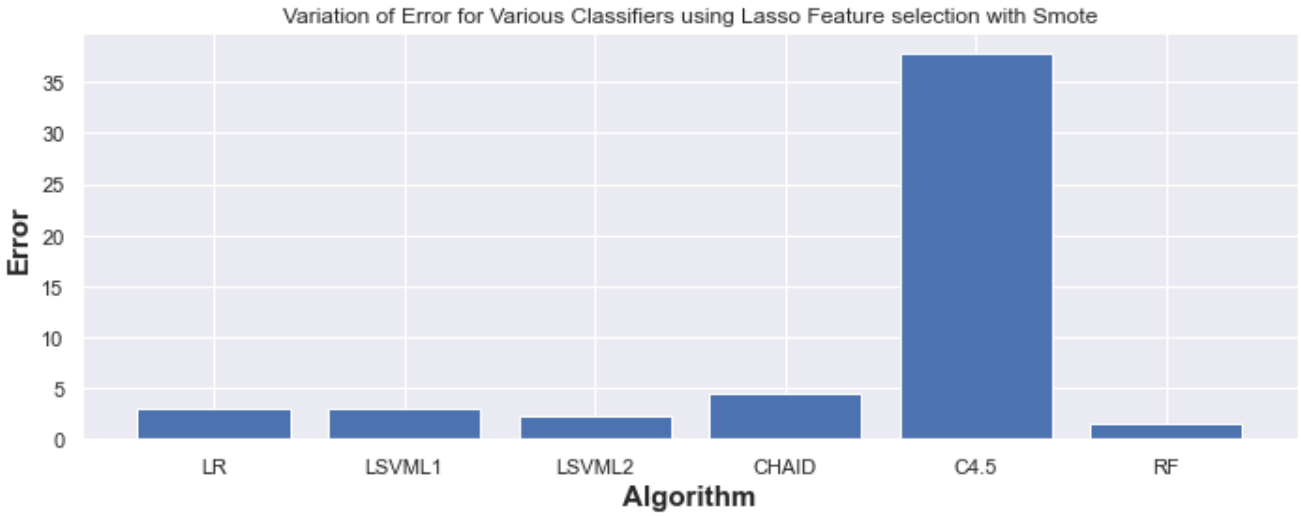


Fig. 4.19. Classification Error for ML classifiers in case of LASSO FS with SMOTE

F- Measure:

Here are the plotted graphs of F- Measure for the Machine Learning classifiers in case of Full feature selection, CFS, Wrapper FS, LASSO FS, Full features with SMOTE, LASSO with SMOTE respectively.

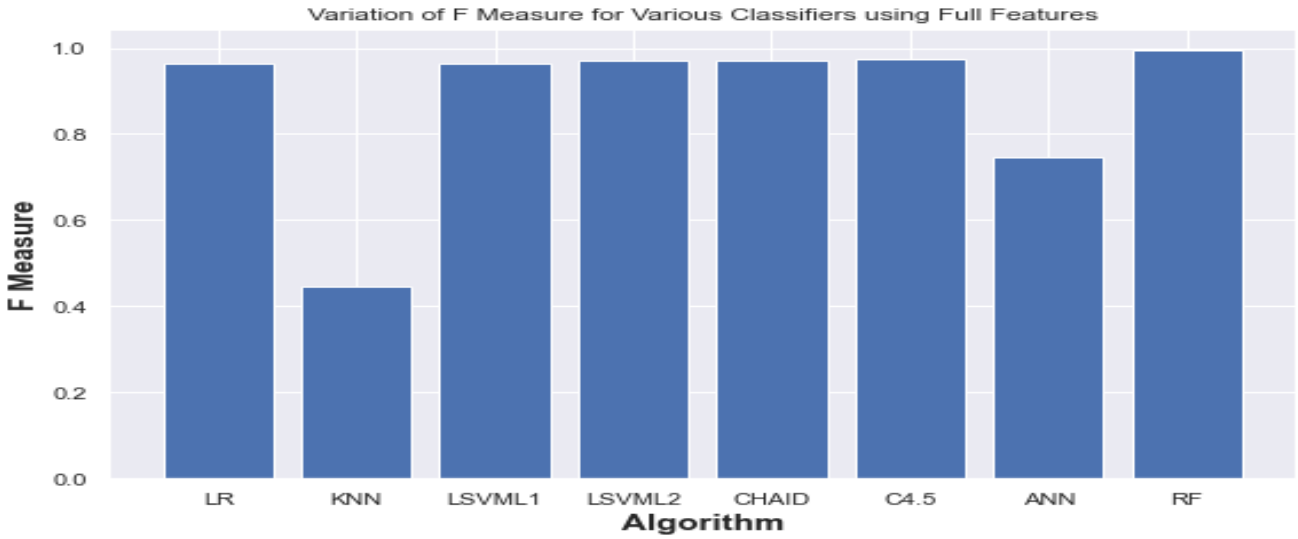


Fig. 4.20. F- Measure for ML classifiers in case of Full feature selection

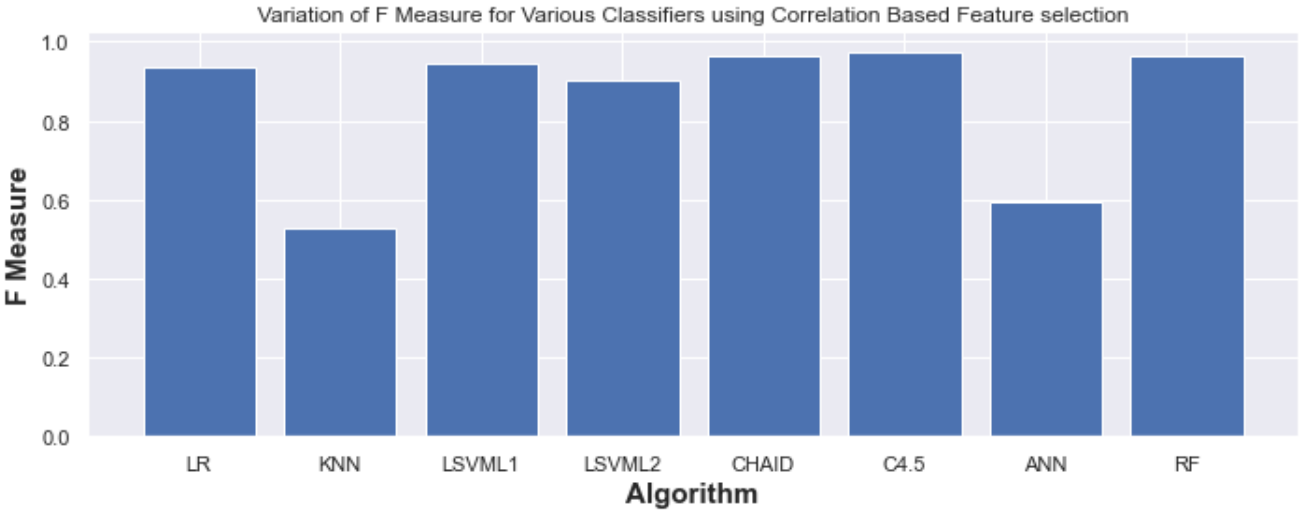


Fig. 4.21. F- Measure for ML classifiers in case of CFS

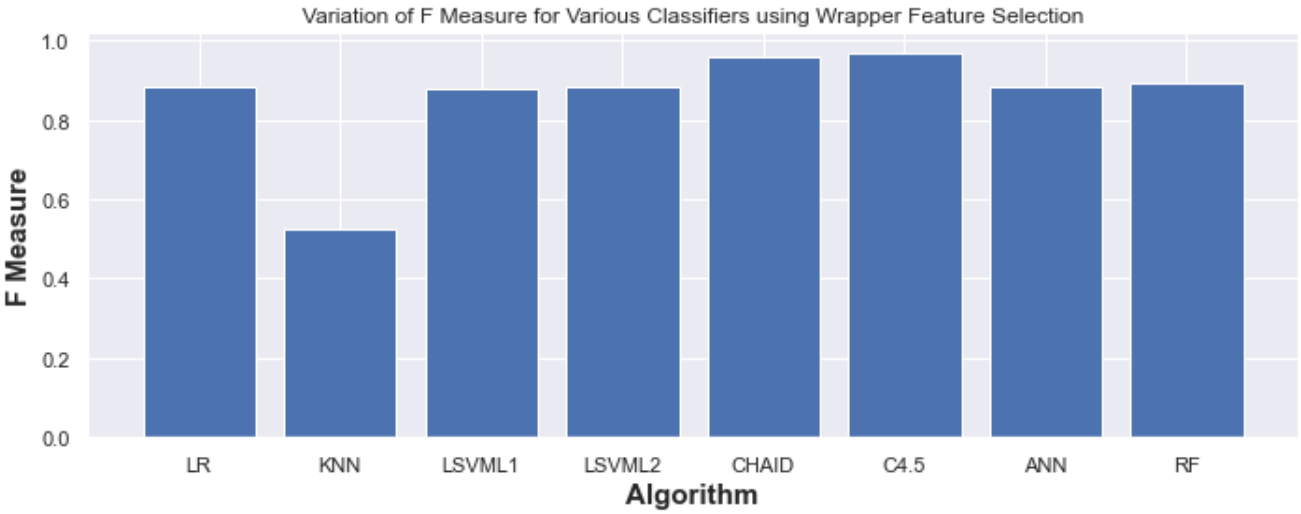


Fig. 4.22. F- Measure for ML classifiers in case of Wrapper FS

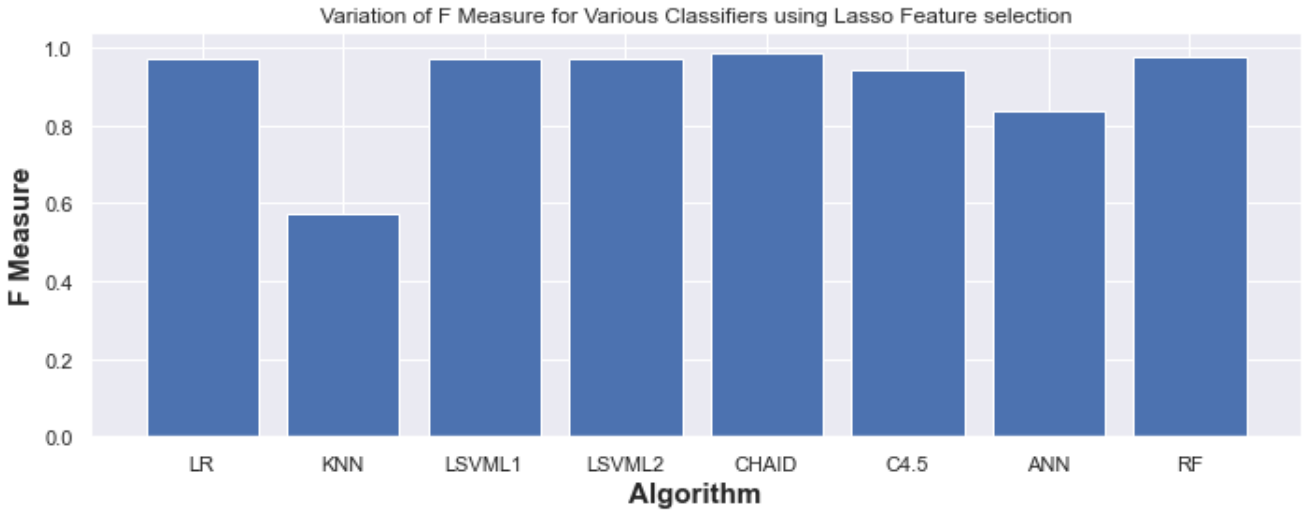


Fig. 4.23. F- Measure for ML classifiers in case of LASSO FS

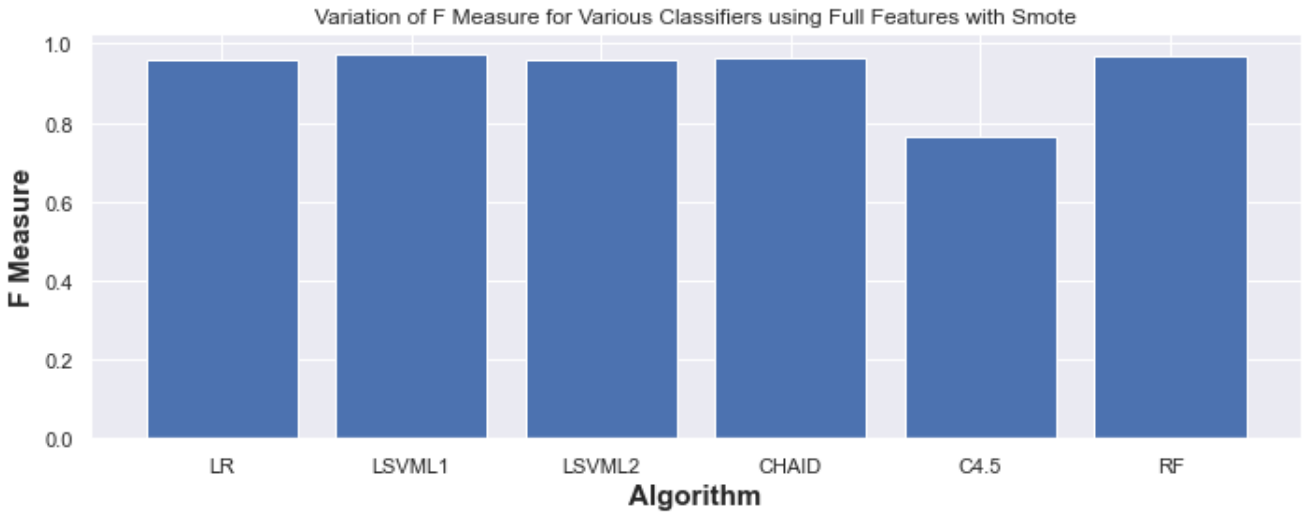


Fig. 4.24. F- Measure for ML classifiers in case of Full features with SMOTE

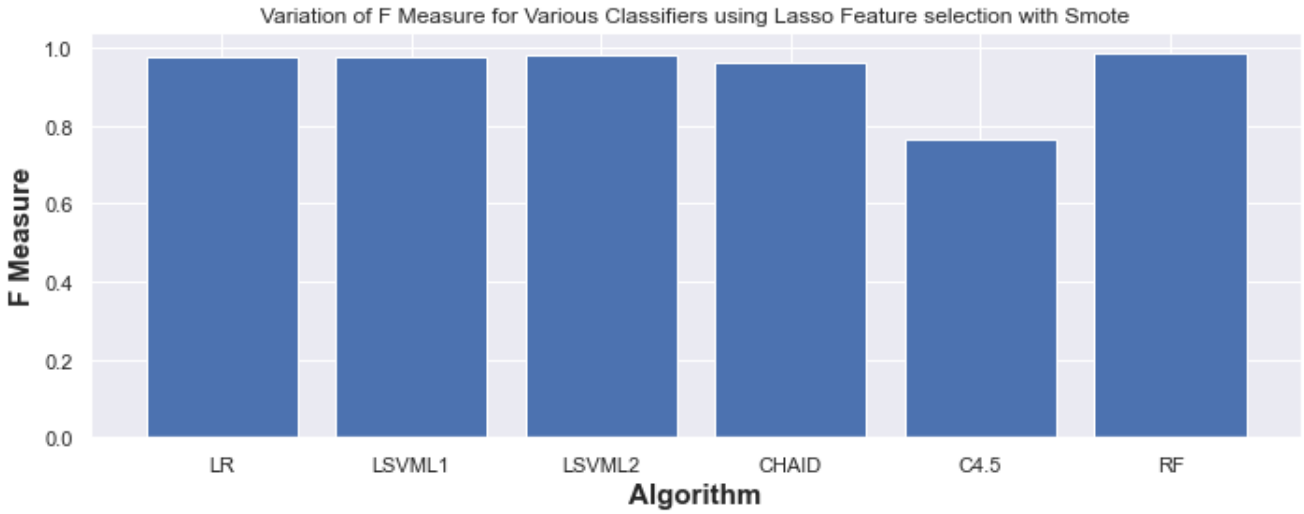


Fig. 4.25. F- Measure for ML classifiers in case of LASSO FS with SMOTE

AUC:

Here are the plotted graphs of AUC for the Machine Learning classifiers in case of Full feature selection, CFS, Wrapper FS, LASSO FS, Full features with SMOTE, LASSO with SMOTE respectively.

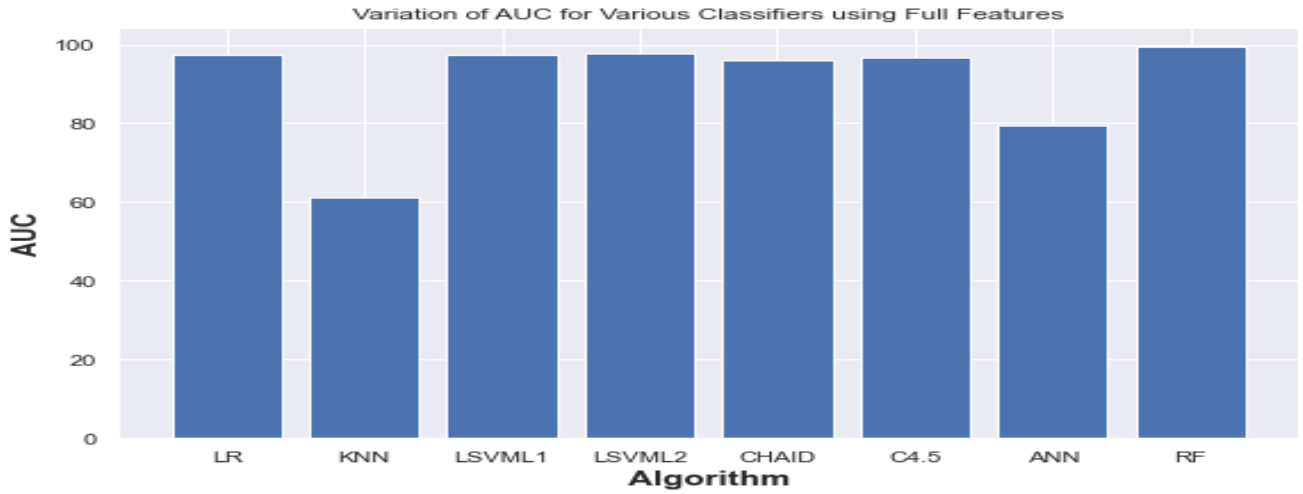


Fig. 4.26. AUC for ML classifiers in case of Full feature selection

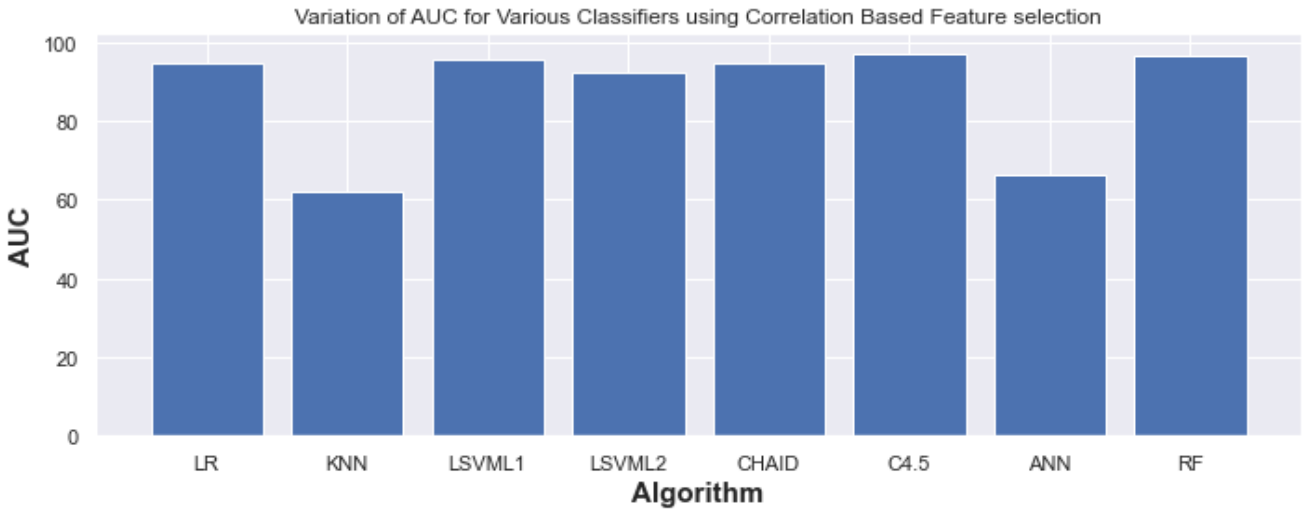


Fig. 4.27. AUC for ML classifiers in case of CFS

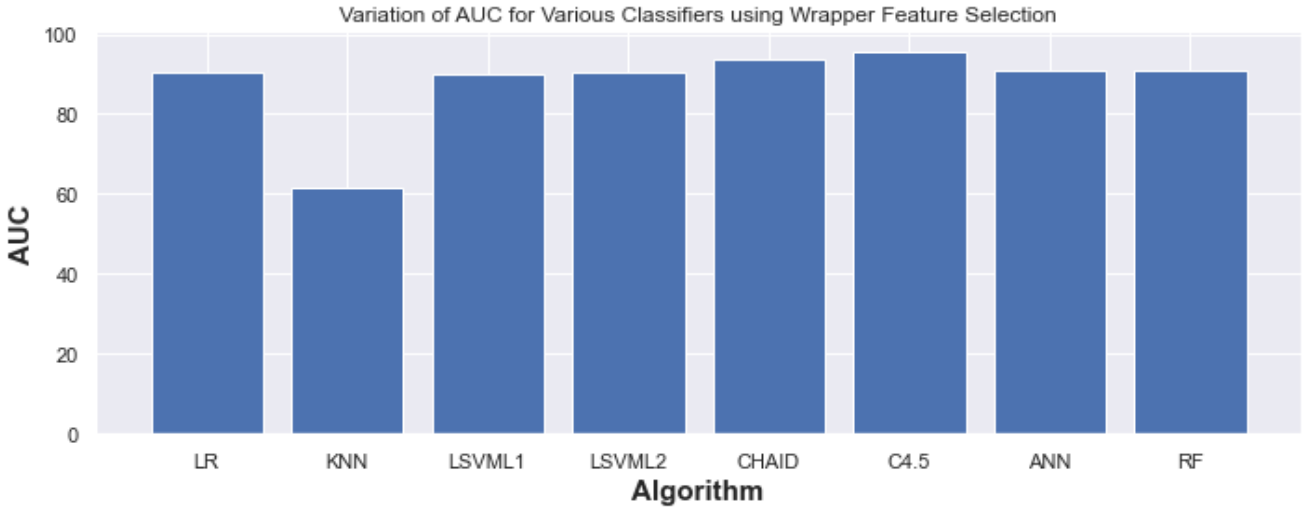


Fig. 4.28. AUC for ML classifiers in case of Wrapper FS

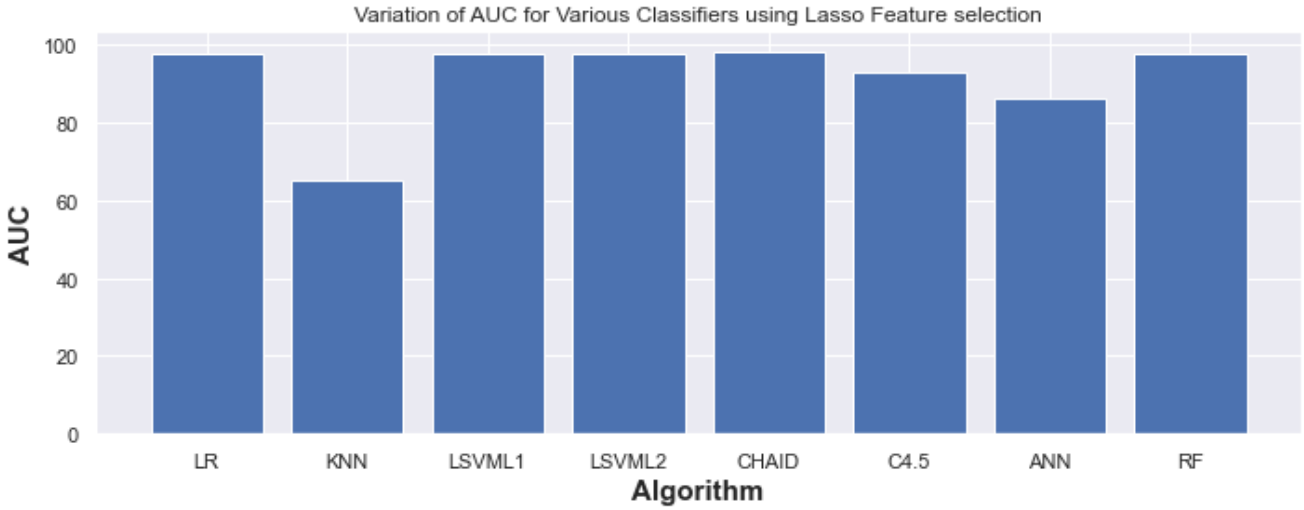


Fig. 4.29. AUC for ML classifiers in case of LASSO FS

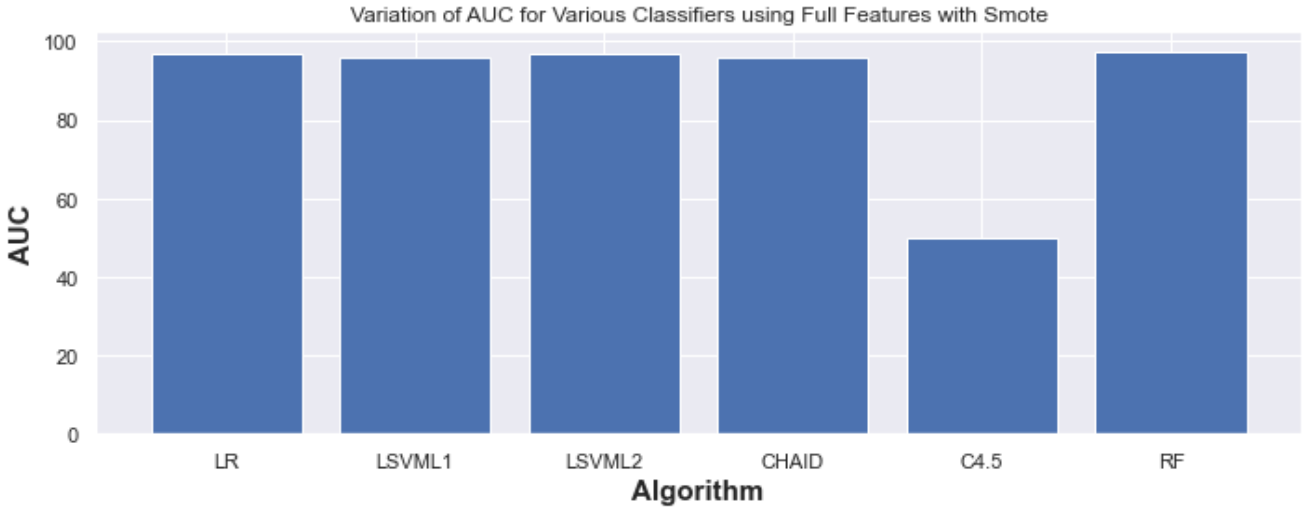


Fig. 4.30. AUC for ML classifiers in case of Full features with SMOTE

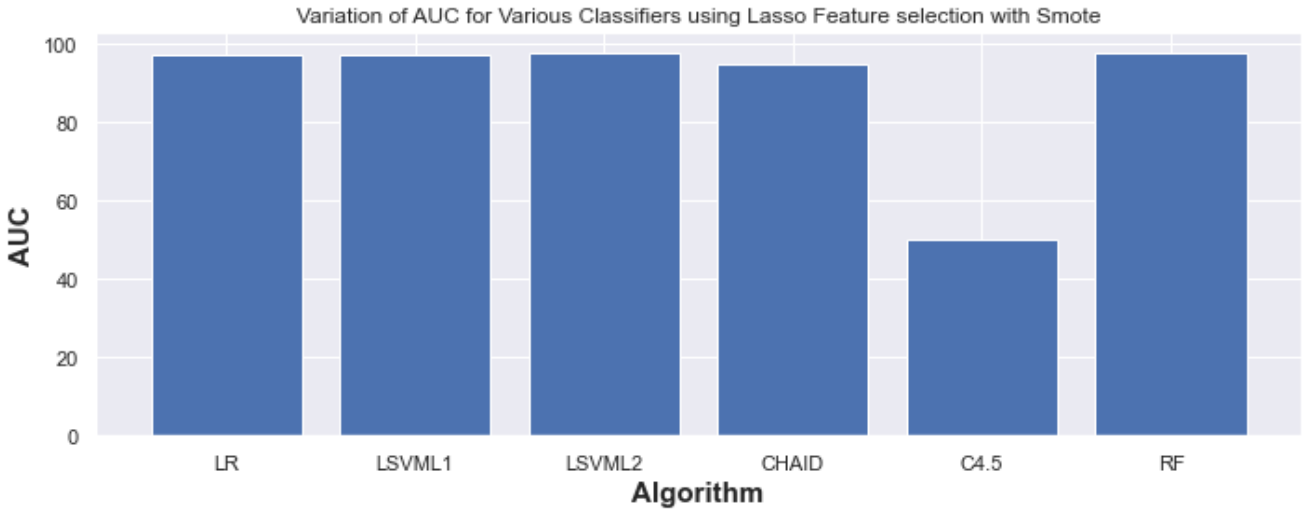


Fig. 4.31. AUC for ML classifiers in case of LASSO FS with SMOTE

GINI Index:

Here are the plotted graphs of GINI Index for the Machine Learning classifiers in case of Full feature selection, CFS, Wrapper FS, LASSO FS, Full features with SMOTE, LASSO with SMOTE respectively.

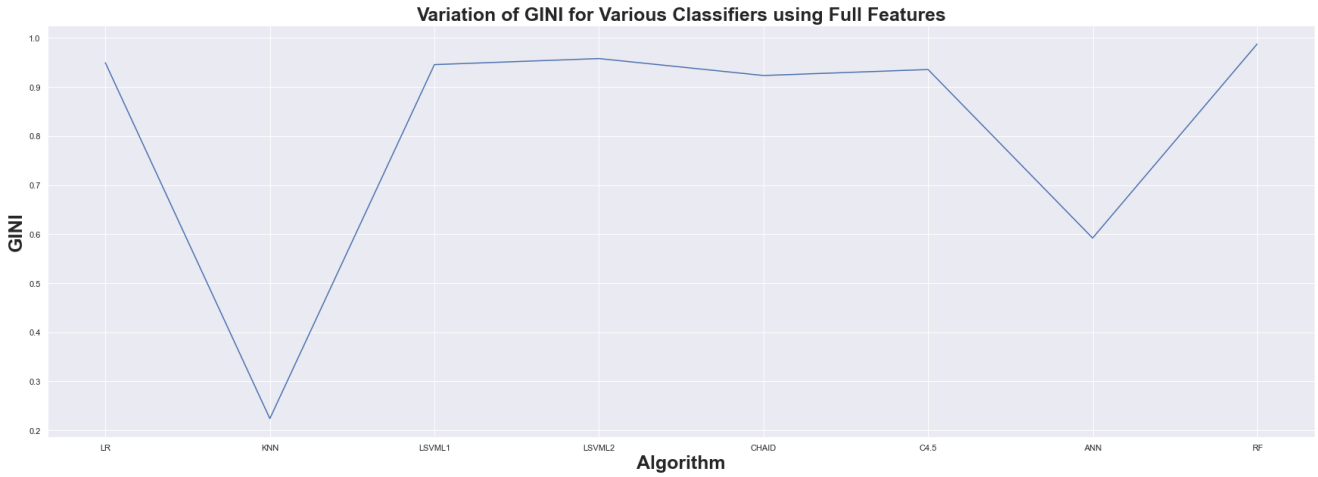


Fig. 4.32. GINI Index for ML classifiers in case of Full feature selection

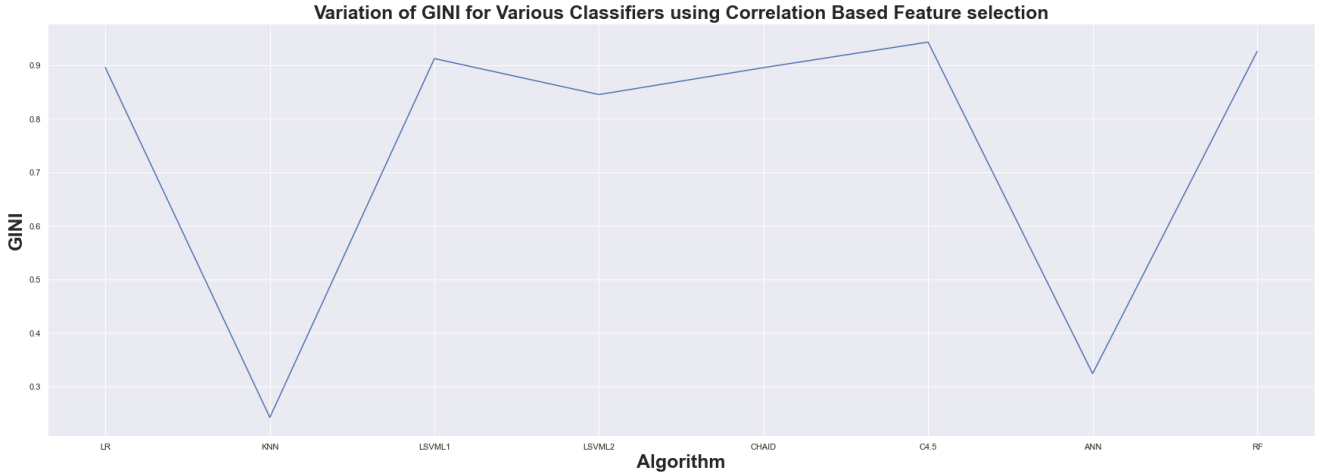


Fig. 4.33. GINI Index for ML classifiers in case of CFS

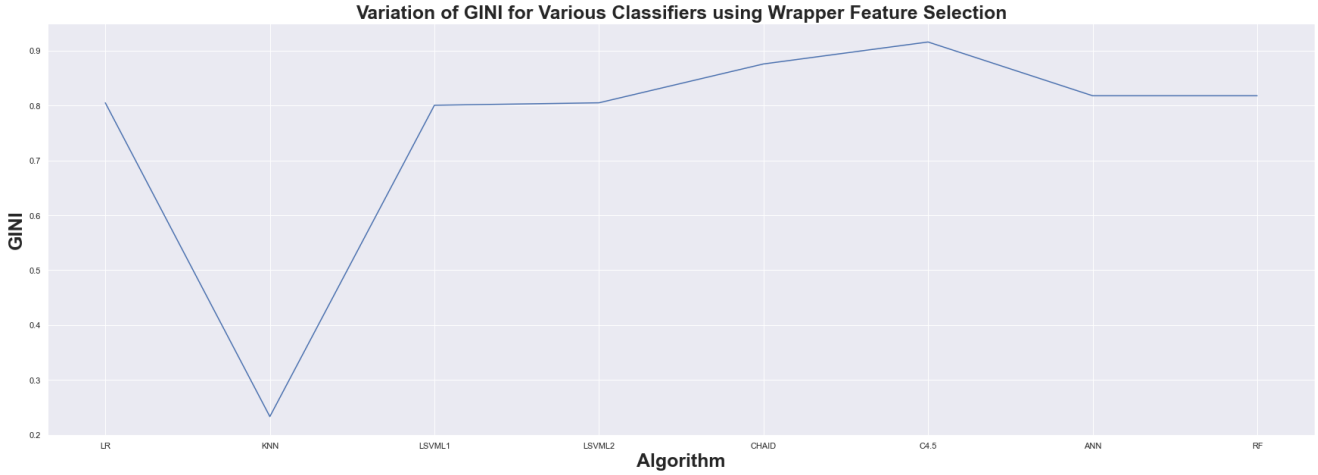


Fig. 4.34. GINI Index for ML classifiers in case of Wrapper FS

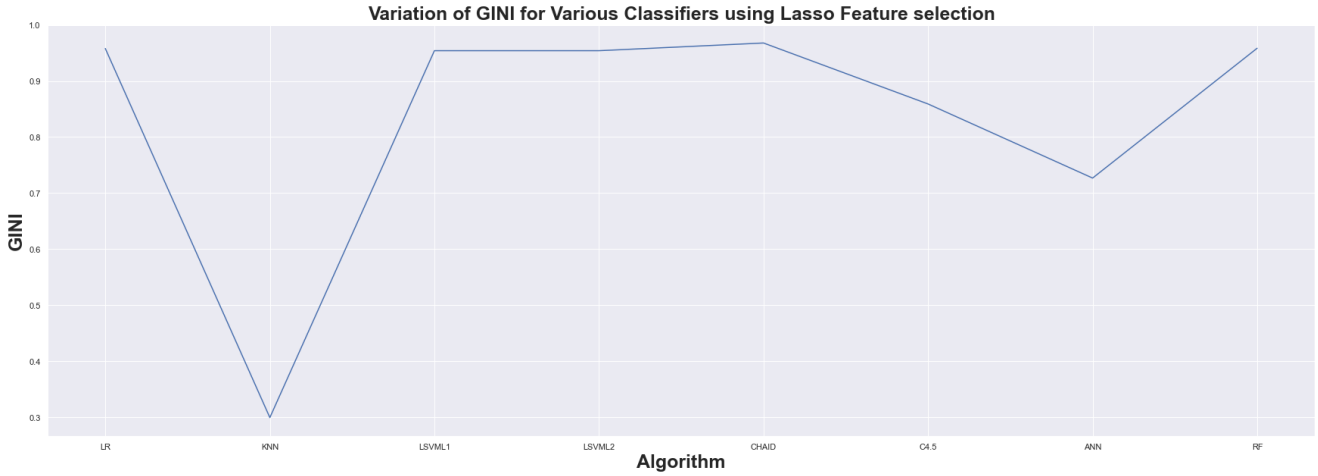


Fig. 4.35. GINI Index for ML classifiers in case of LASSO FS

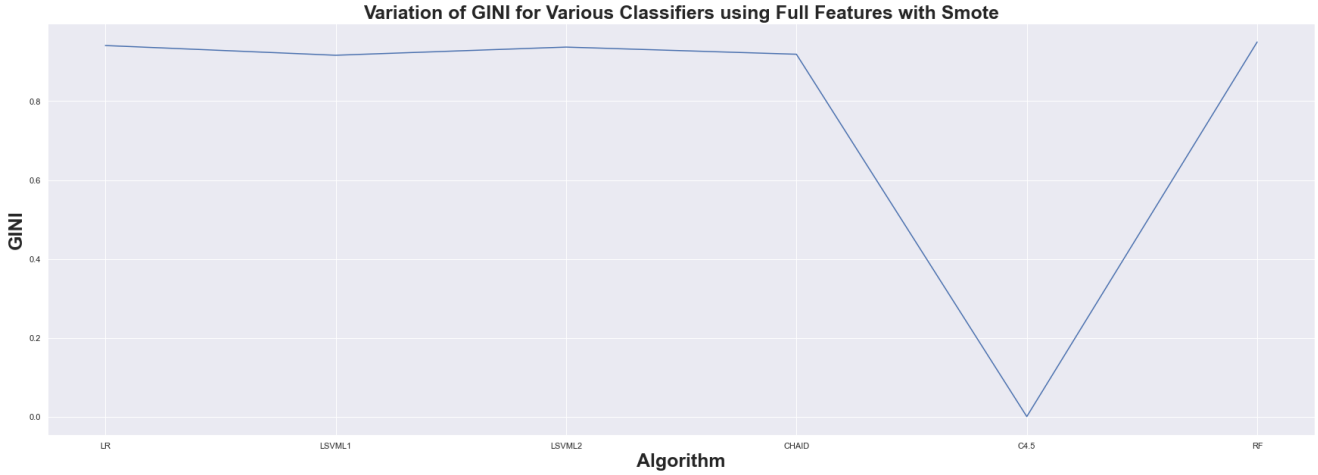


Fig. 4.36. GINI Index for ML classifiers in case of Full features with SMOTE

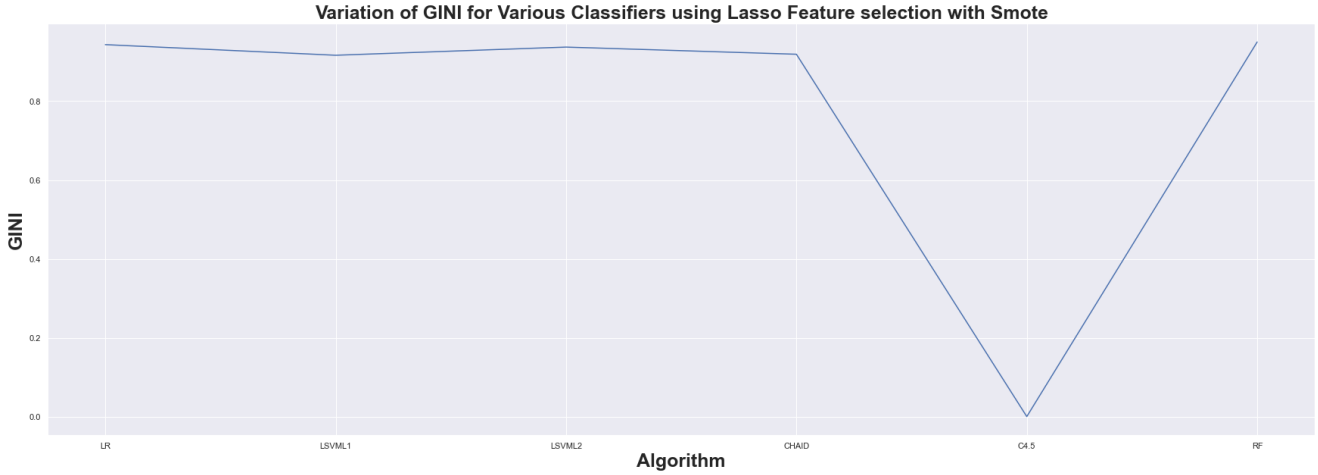


Fig. 4.37. GINI Index for ML classifiers in case of LASSO FS with SMOTE

CHAPTER 5

CONCLUSION AND FUTURE PLANS

5.1. Conclusion

The main topic of this research is the prediction of CKD using ML classifier techniques. We employed ML classifier methods as RF, CHAID, LSVM with penalty L1 and penalty L2, LR, KNN, LSVM with penalty L2, C 4.5, and ANN. For each classifier, the results for full feature, correlation-based, wrapper, and LASSO FS are computed.

LR, LSVM L1, LSVM L2, CHAID, C 4.5, and RF outperform Full Features and LASSO Feature Selection in terms of accuracy and other parameters.

SMOTE is used to calculate the outcomes of the classifiers and FS methods stated above. We did not employ the KNN and ANN algorithms for SMOTE since they did not yield results that were sufficient.

Random Forest outperforms LSVM penalty L2 with an accuracy of 98.49 percent and LASSO Feature Selection and SMOTE with an accuracy of 97.73 percent.

The findings led to the conclusion that Random Forest with LASSO Feature Selection with SMOTE yields better outcomes for predicting CKD.

5.2. Future Plans

We oversampled our data set with SMOTE because it had an average amount of instances. The same method can be used to perform undersampling in the future when a data collection with many instances is being considered.

CHAPTER 6

REFERENCES

[1] Pankaj Chittora, Sandeep Chaurasia, Prasun Chakrabarti, Gaurav Kumawat1, Tulika Chakrabarti, Zbigniew Leonowicz, Michał jasiński, Lukasz Jasiński, Radomir Gono, Elżbieta Jasińska, and Vadim Bolshev , (2021) "prediction of chronic kidney disease - a machine learning perspective", *ieee access.2021.3053763.*

[2] J. Qin, L. Chen, Y. Liu, C. Liu, C. Feng, and B. Chen, ''A machine learning methodology for diagnosing chronic kidney disease,'' *IEEE Access, vol. 8, pp. 20991–21002, 2020.*

[3] L. Kilvia De Almeida, L. Lessa, A. Peixoto, R. Gomes, and J. Celestino, ''Kidney failure detection using machine learning techniques,'' *in Proc. 8th Int. Workshop ADVANCEs ICT Infrastructures Services, 2020, pp. 1–8.*

[4] S. Shankar, S. Verma, S. Elavarthy, T. Kiran, and P. Ghuli, ''Analysis and prediction of chronic kidney disease,'' *Int. Res. J. Eng. Technol., vol. 7, no. 5, May 2020, pp. 4536–4541.*

[5] G. R. Vasquez-Morales, S. M. Martinez-Monterrubio, P. Moreno-Ger, and J. A. Recio-Garcia, ''Explainable prediction of chronic renal disease in the colombian population using neural networks and case-based reasoning,'' *IEEE Access, vol. 7, pp. 152900–152910, 2019.*

[6] M. Almasoud and T. E. Ward, ''Detection of chronic kidney disease using machine learning algorithms with least number of predictors,'' *Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 8, pp. 89–96, 2019.*

[7] Y. Amirgaliyev, S. Shamiluulu, and A. Serek, ''Analysis of chronic kidney disease dataset by applying machine learning methods,'' in Proc. *IEEE 12th Int. Conf. Appl. Inf. Commun. Technol. (AICT), Oct. 2018, pp. 1–4.*

[8] W. Gunarathne, K. D. M Perera, and K. A. D. C. P Kahandawaarachchi,''Performance evaluation on machine learning classification techniques for disease classification and forecasting through data analytics for chronic kidney disease (CKD),'' *in Proc. IEEE 17th Int. Conf. Bioinf. Bioeng. (BIBE), Oct.2017, pp.291–296.*

[9] K. R. A. Padmanaban and G. Parthiban, ''Applying machine learning techniques for predicting the risk of chronic kidney disease,'' *Indian J. Sci. Technol., vol. 9, no. 29, Aug. 2016.*

CHAPTER 6

APPENDIX -BASE PAPER

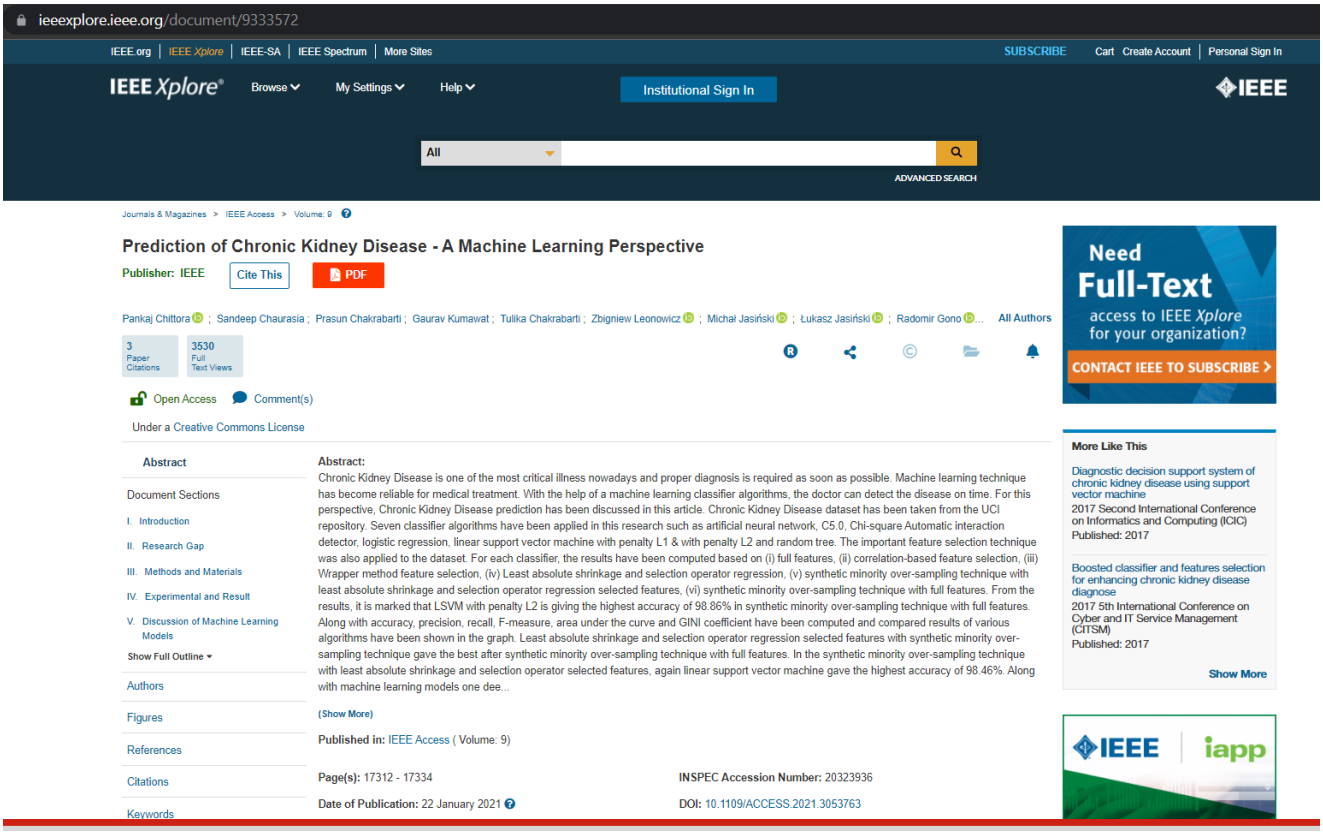


Fig. 6.1. Screenshot of Indexing

vvv

ORIGINALITY REPORT

10%	3%	9%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Pankaj Chittora, Sandeep Chaurasia, Prasun Chakrabarti, Gaurav Kumawat et al. "Prediction of Chronic Kidney Disease - A Machine Learning Perspective", IEEE Access, 2021 Publication	2%
2	Olayinka Ayodele Jongbo, Adebayo Olusola Adetunmbi, Roseline Bosede Ogunrinde, Bukola Badeji-Ajisafe. "Development of an Ensemble Approach to Chronic Kidney Disease Diagnosis", Scientific African, 2020 Publication	1%
3	Samrat Kumar Dey, Khandaker Mohammad Mohi Uddin, Hafiz Md. Hasan Babu, Md. Mahbubur Rahman, Arpita Howlader, K.M. Aslam Uddin. "Chi2-MI: A hybrid feature selection based machine learning approach in diagnosis of chronic kidney disease", Intelligent Systems with Applications, 2022 Publication	1%

4	"Decision Sciences for COVID-19", Springer Science and Business Media LLC, 2022 Publication	1%
5	Novia Arinda Pradisty, A. Aldrie Amir, Martin Zimmer. "Plant species- and stage-specific differences in microbial decay of mangrove leaf litter: the older the better?", Oecologia, 2021 Publication	1%
6	core.ac.uk Internet Source	1%
7	Wendy D. Fife. "Comparison Between Malignant and Nonmalignant Splenic Masses in Dogs using Contrast-Enhanced Computed Tomography", Veterinary Radiology & Ultrasound, 7/2004 Publication	<1%
8	www.biorxiv.org Internet Source	<1%
9	www.imedpub.com Internet Source	<1%
10	nceca.in Internet Source	<1%
11	ebin.pub Internet Source	<1%

12	W.H.S.D Gunarathne, K.D.M Perera, K.A.D.C.P Kahandawaarachchi. "Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease (CKD)", 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), 2017 Publication	<1%
13	Yan Chai Hum. "Segmentation of Hand Bone for Bone Age Assessment", Springer Science and Business Media LLC, 2013 Publication	<1%
14	conf.kln.ac.lk Internet Source	<1%
15	Muhammet Sinan Basarslan, Fatih Kayaalp. "Performance Analysis Of Fuzzy Rough Set-Based And Correlation-Based Attribute Selection Methods On Detection Of Chronic Kidney Disease With Various Classifiers", 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), 2019 Publication	<1%
16	d.researchbib.com Internet Source	<1%
17	vbn.aau.dk Internet Source	

<1%

18	Nikhila. "Chronic Kidney Disease Prediction using Machine Learning Ensemble Algorithm", 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		