



**North South University**

**Department of Electrical and Computer Engineering**

**CSE445 – Machine Learning**

**Section: 6**

**Project Topic**

**Risk Factor Prediction of Chronic Kidney Disease**

**Group Members**

Md. Rifat Ahmed 1931725042	Faija Islam Oishe 1821720042	Sumiya Sultana 1931277642
-------------------------------	---------------------------------	------------------------------

# 1. Pre-processing

To Pre-process the Dataset, we've used Pandas on the Local system using the following Codes:

```
import pandas as pd

#Importing the Dataset
df = pd.read_csv('D:\Varsity Documents\Spring 23\CSE445\Works\Project\CKD
Prediction Dataset.csv')
```

```
# Viewing the first 5 rows
df.head()
```

```
# Since the first two rows doesn't have any values needed for our work
# We'll Drop the first two rows
df = df.drop([0, 1])

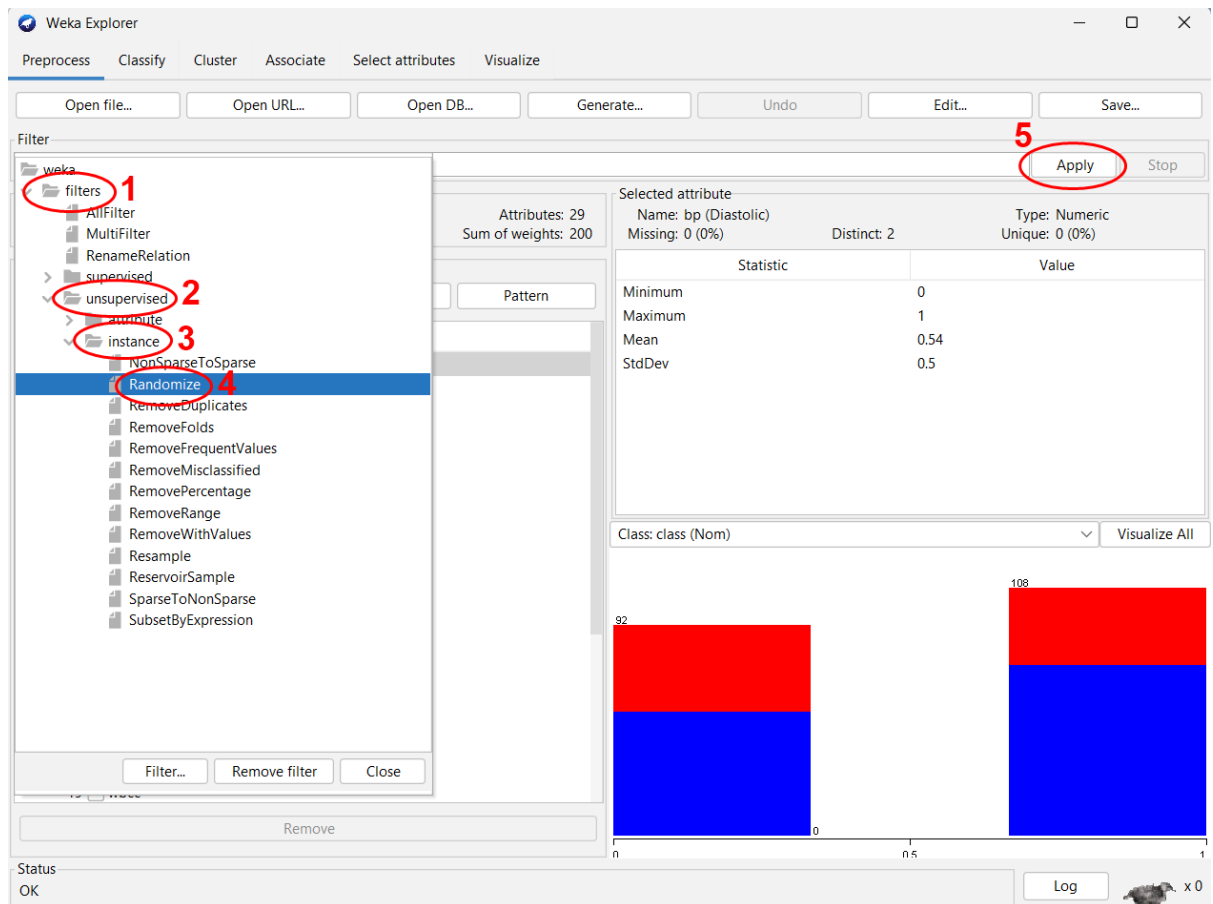
# Resetting the index of the DataFrame
df = df.reset_index(drop=True)

df.head()
```

```
# Exporting the newly updated Dataframe as a new Dataset
df.to_csv('D:\Varsity Documents\Spring 23\CSE445\Works\Project\CKD Prediction
Dataset (Pre-Processed).csv', index=False)
```

➤ Now, we'll move on to Weka for the Remaining Parts:

After opening the new CSV file on Weka we'll use the Randomize filter from Filter section to Randomly shuffle the Order of the instances in our Dataset:



Now, since we don't have any missing values in the dataset, we don't have to deal with any.

## 2. EDA & Visualization

Current relation  
Relation: CKD Prediction Dataset (Pre-Processed)-weka.filters.un...  
Instances: 200  
Attributes: 29  
Sum of weights: 200

Attributes

All None Invert Pattern

No.	Name
1	<input type="checkbox"/> bp (Diastolic)
2	<input type="checkbox"/> bp limit
3	<input type="checkbox"/> sg
4	<input type="checkbox"/> al
5	<input checked="" type="checkbox"/> class
6	<input type="checkbox"/> rbc
7	<input type="checkbox"/> su
8	<input type="checkbox"/> pc
9	<input type="checkbox"/> pcc
10	<input type="checkbox"/> ba
11	<input type="checkbox"/> bgr
12	<input type="checkbox"/> bu
13	<input type="checkbox"/> sod
14	<input type="checkbox"/> sc
15	<input type="checkbox"/> pot
16	<input type="checkbox"/> hemo
17	<input type="checkbox"/> pcv
18	<input type="checkbox"/> rbcc
19	<input type="checkbox"/> wbcc

Remove

Status  
OK

Selected attribute

Name: class  
Missing: 0 (0%)  
Distinct: 2  
Type: Nominal  
Unique: 0 (0%)

No.	Label	Count	Weight
1	ckd	128	128
2	notckd	72	72

Their Values

Class: class (Nom) Visualize All

Visualization

128

72

Log x 0

On the left side we can see all the attributes (total 29 in our dataset) and clicking on them we can see their values on the right side and their visualization at the bottom right corner.

For this instance, we can see that the attribute “class” holds the information “ckd” and “notckd” meaning either someone has ckd or they do not. And the visualization is shown based on the number of instances with these values.

### 3. Classification

We'll be classifying the dataset using two algorithms

- i) Logistic Regression
- ii) Random Forest

& we'll be using both cross-validation and dataset splitting technique to get the results to see which gives the best result.

And as for the Evaluation metrics we'll be keeping MAE, RMSE, TP Rate, FP Rate, Precision, Recall & F-score.

**Method 1:** We'll be using Logistic Regression using Cross-validation technique with 10 folds:

The screenshot shows the RStudio 'Classifier' window. The 'Choose' dropdown is set to 'Logistic -R 1.0E-8 -M -1 -num-decimal-places 4'. Under 'Test options', 'Cross-validation' is selected with 'Folds' set to 10. The 'Result list' on the left shows '14:37:36 - functions.Logistic' selected. The 'Classifier output' pane displays the following results:

```
age=35 - 43      78.2631
age=43 - 51      5.0057
age=51 - 59      0.0157
age=59 - 66      0.4272
age=66 - 74      0.0076
age=74 - 74      0.0002

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      195      97.5 %
Incorrectly Classified Instances    5       2.5 %
Kappa statistic                    0.9452
Mean absolute error                 0.0225
Root mean squared error             0.1435
Total Number of Instances          200

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall   F-Measure  Class
              0.992    0.056    0.969     0.992    0.981      ckd
              0.944    0.008    0.986     0.944    0.965      notckd
Weighted Avg.   0.975    0.038    0.975     0.975    0.975

=== Confusion Matrix ===

  a  b  <-- classified as
127  1 |  a = ckd
  4  68 |  b = notckd
```

We got an accuracy of 97.5%.

**Method 2:** Again, using Logistic Regression but this time we split the training dataset into 70%:

The screenshot shows the RStudio Classifier interface. The 'Test options' section has 'Percentage split' selected with a value of 70%. The 'Classifier output' section displays the following data:

Classifier output	
age=59 - 66	0.4272
age=66 - 74	0.0076
age=74 - 74	0.0002

Time taken to build model: 0.01 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances	59	98.3333 %
Incorrectly Classified Instances	1	1.6667 %
Kappa statistic	0.962	
Mean absolute error	0.0235	
Root mean squared error	0.1327	
Total Number of Instances	60	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
	0.976	0.000	1.000	0.976	0.988	ckd
	1.000	0.024	0.950	1.000	0.974	notckd
Weighted Avg.	0.983	0.008	0.984	0.983	0.983	

=== Confusion Matrix ===

```
a b <-- classified as
40 1 | a = ckd
0 19 | b = notckd
```

This time we got an accuracy of 98.3333%.

**Method 3:** Now, using Random Forest algorithm using Cross-validation technique with 10 folds:

Classifier: Choose **RandomForest** -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Test options:  
☐ Use training set  
☐ Supplied test set Set...  
☒ Cross-validation Folds **10**  
☐ Percentage split % 70  
 More options...

(Nom) class: **ckd**

Start Stop

Result list (right-click for options):  
 14:37:36 - functions.Logistic  
 14:38:36 - functions.Logistic  
**14:39:01 - trees.RandomForest**

Classifier output:  
 === Classifier model (full training set) ===  
 RandomForest  
 Bagging with 100 iterations and base learner  
 weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities  
 Time taken to build model: 0.03 seconds  
 === Stratified cross-validation ===  
 === Summary ===  

Correctly Classified Instances	200	100	%
Incorrectly Classified Instances	0	0	%
Kappa statistic	1		
Mean absolute error	0.0556		
Root mean squared error	0.0973		
Total Number of Instances	200		

 === Detailed Accuracy By Class ===  

	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
	1.000	0.000	1.000	1.000	1.000	ckd
	1.000	0.000	1.000	1.000	1.000	notckd
Weighted Avg.	1.000	0.000	1.000	1.000	1.000	

 === Confusion Matrix ===  
 a b <-- classified as  
 128 0 | a = ckd  
 0 72 | b = notckd

We got an accuracy of 100%.

**Method 4:** Again, using Random Forest but this time we split the training dataset into 70%:

Classifier: Choose **RandomForest** -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Test options:  
☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds 10  
☒ Percentage split % 70  
 More options...

(Nom) class: **ckd**

Start Stop

Result list (right-click for options):  
 14:37:36 - functions.Logistic  
 14:38:36 - functions.Logistic  
 14:39:01 - trees.RandomForest  
**14:41:25 - trees.RandomForest**

Classifier output:  
 Bagging with 100 iterations and base learner  
 weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities  
 Time taken to build model: 0.01 seconds  
 === Evaluation on test split ===  
 Time taken to test model on test split: 0 seconds  
 === Summary ===  

Correctly Classified Instances	60	100	%
Incorrectly Classified Instances	0	0	%
Kappa statistic	1		
Mean absolute error	0.086		
Root mean squared error	0.1484		
Total Number of Instances	60		

 === Detailed Accuracy By Class ===  

	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
	1.000	0.000	1.000	1.000	1.000	ckd
	1.000	0.000	1.000	1.000	1.000	notckd
Weighted Avg.	1.000	0.000	1.000	1.000	1.000	

 === Confusion Matrix ===  
 a b <-- classified as  
 41 0 | a = ckd  
 0 19 | b = notckd

And again, we've got an accuracy of 100%.

## 4. Result & Conclusion

Even though the Mean Absolute Error and the Root Mean Squared Error are bit varying. Using the 1<sup>st</sup> method, we get the lowest MAE while using the 3<sup>rd</sup> method we get the lowest RMSE.

But, to classify the dataset of Chronic Kidney Disease the best algorithm would be the Random Forest since we're getting a very small error in Logistic Regression and also it's giving an accuracy of 100% using both cross-validation and splitting technique.