PREDICTING CHRONIC KIDNEY DISEASE

Q1. Identify your problem statement

- Here data are **number formats**, so it comes under ***Machine Learning***.
- Requirement is clear and input and output data are present, so it comes under ***Supervised Learning***.
- Going to predict the chronic kidney disease are **present or not**, so it meant the ***Classification***.

Q2. Tell basic info about the dataset (Total number of rows, columns)

- Initially the data is 399 rows and 25 columns. Among 25 columns, “classification” feature represents the output/target variable.

```
[2]: dataset = pd.read_csv("CKD.csv") #read the csv file using pandas library

     dataset
```

[2]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane
0	2.000000	76.459948	c	3.0	0.0	normal	abnormal	notpresent	notpresent	148.112676	...	38.868902	8408.191126	4.705597	no	no	no	yes	yes	no
1	3.000000	76.459948	c	2.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	34.000000	12300.000000	4.705597	no	no	no	yes	poor	no
2	4.000000	76.459948	a	1.0	0.0	normal	normal	notpresent	notpresent	99.000000	...	34.000000	8408.191126	4.705597	no	no	no	yes	poor	no
3	5.000000	76.459948	d	1.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	38.868902	8408.191126	4.705597	no	no	no	yes	poor	yes
4	5.000000	50.000000	c	0.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	36.000000	12400.000000	4.705597	no	no	no	yes	poor	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
394	51.492308	70.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	219.000000	...	37.000000	9800.000000	4.400000	no	no	no	yes	poor	no
395	51.492308	70.000000	c	0.0	2.0	normal	normal	notpresent	notpresent	220.000000	...	27.000000	8408.191126	4.705597	yes	yes	no	yes	poor	yes
396	51.492308	70.000000	c	3.0	0.0	normal	normal	notpresent	notpresent	110.000000	...	26.000000	9200.000000	3.400000	yes	yes	no	poor	poor	no
397	51.492308	90.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	207.000000	...	38.868902	8408.191126	4.705597	yes	yes	no	yes	poor	yes
398	51.492308	80.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	100.000000	...	53.000000	8500.000000	4.900000	no	no	no	yes	poor	no

399 rows × 25 columns

Q3. Mention the pre-processing method if you’re doing any (like converting string to number – nominal data)

- **One hot encoder** is done in the original dataset using get_dummies. Because few features are in string format ex: sg, pa, etc. After preprocessing, it has 399 rows and 28 columns

[5]:
```
#data preprocessing - because categorial data can't be handle so we modify the data into numerical data using one hot encoding(Nominal)

dataset = pd.get_dummies(dataset, drop_first=True) #"drop_first=True" removes 1st category of each categorical column

dataset
```

[5]:

	age	bp	al	su	bgr	bu	sc	sod	pot	hrmo	...	pc_normal	pcc_present	ba_present	htn_yes	dm_yes	cad_ye
0	2.000000	76.459948	3.0	0.0	148.112676	57.482105	3.077356	137.528754	4.627244	12.518156	...	False	False	False	False	False	Fals
1	3.000000	76.459948	2.0	0.0	148.112676	22.000000	0.700000	137.528754	4.627244	10.700000	...	True	False	False	False	False	Fals
2	4.000000	76.459948	1.0	0.0	99.000000	23.000000	0.600000	138.000000	4.400000	12.000000	...	True	False	False	False	False	Fals
3	5.000000	76.459948	1.0	0.0	148.112676	16.000000	0.700000	138.000000	3.200000	8.100000	...	True	False	False	False	False	Fals
4	5.000000	50.000000	0.0	0.0	148.112676	25.000000	0.600000	137.528754	4.627244	11.800000	...	True	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
394	51.492308	70.000000	0.0	0.0	219.000000	36.000000	1.300000	139.000000	3.700000	12.500000	...	True	False	False	False	False	Fals
395	51.492308	70.000000	0.0	2.0	220.000000	68.000000	2.800000	137.528754	4.627244	8.700000	...	True	False	False	True	True	Fals
396	51.492308	70.000000	3.0	0.0	110.000000	115.000000	6.000000	134.000000	2.700000	9.100000	...	True	False	False	True	True	Fals
397	51.492308	90.000000	0.0	0.0	207.000000	80.000000	6.800000	142.000000	5.500000	8.500000	...	True	False	False	True	True	Fals
398	51.492308	80.000000	0.0	0.0	100.000000	49.000000	1.000000	140.000000	5.000000	16.300000	...	True	False	False	False	False	Fals

399 rows × 28 columns

Q4. Develop a good model with good evaluation metric. You can use any machine learning algorithm; you can create many models. Finally, you have to come up with final model.

- I have used following algorithms and evaluated with precision, recall, f1_score, accuracy and ROC AUC to get the best model
  - Support vector classifier
  - Decision Tree
  - Random Forest
  - Logistic Regression
  - KNN and
  - Naïve Bayes – Gaussian, Multinomial, Complement, Bernoulli
- So, **Naïve Bayes – Gaussian and Random Forest** are **best model** in accuracy and roc_auc curve.

Q5. All the research values of each algorithm should be documented. (You can make tabulation or screenshot of the results.)

Algorithm	*Precision_No*	*Precision_Yes*	*Recall_No*	*Recall_Yes*	*F1-score_No*	*F1-score_Yes*	*Accuracy*	*ROC_AUC*
SVC	0.68	0.90	0.87	0.76	0.76	0.83	0.80	0.868
Decision Tree	0.95	0.96	0.93	0.97	0.94	0.97	0.96	0.987
Random Forest	0.98	0.99	0.98	0.99	0.98	0.99	0.98	0.999
Logistic Regression	0.96	0.99	0.98	0.97	0.97	0.98	0.97	0.998
KNN	0.66	0.95	0.93	0.71	0.77	0.81	0.79	0.858
Naïve Bayes – Gaussian	0.96	1.00	1.00	0.97	0.98	0.99	0.98	1.000
Naïve Bayes – Multinomial	0.67	0.98	0.98	0.71	0.79	0.82	0.81	0.909

Naïve Bayes – Complement	0.67	0.98	0.98	0.71	0.79	0.82	0.81	0.909
Naïve Bayes – Bernoulli	0.85	1.00	1.00	0.89	0.92	0.94	0.93	0.996

Q6. Mention your final model, justify why u have chosen the same.

- Below models are best, because accuracy is 98% for Random Forest and Naïve Bayes- Gaussian.
- **Naïve Bayes – Gaussian** ROC_AUC is performed well, which means it showed the **prefect classification probability score.**

Algorithm	***Precision_No***	***Precision_Yes***	***Recall_No***	***Recall_Yes***	***F1-score_No***	***F1-score_Yes***	***Accuracy***	***ROC_AUC***
Random Forest	0.98	0.99	0.98	0.99	0.98	0.99	0.98	0.999
Naïve Bayes – Gaussian	0.96	1.00	1.00	0.97	0.98	0.99	0.98	1.000