CKD Assignment in ML Classification

1. **Dataset** has 399 rows × 25 columns

Actual Dataset has 25 Columns	'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hrmo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane', 'classification'
Numerical Columns	Age, Bp, Al, Su, Bgr, Bu, Sc, Sod, Pot, Hrmo, Pcv, Wc, Rc
String Columns	Sg, Rbc, Pc, Pcc, Ba
Yes/No/True/False Columns	Htn, Dm, Cad, Appet, Pe, Ane, Classification
After get_dummies implemented, dataset has 28 Numerical columns	'age', 'bp', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hrmo', 'pcv', 'wc', 'rc', 'sg_b', 'sg_c', 'sg_d', 'sg_e', 'rbc_normal', 'pc_normal', 'pcc_present', 'ba_present', 'htn_yes', 'dm_yes', 'cad_yes', 'appet_yes', 'pe_yes', 'ane_yes', 'classification_yes'

2. Afer get_dummies the Dataset

```
dataset = pd.get_dummies(dataset,drop_first=True,dtype=int)
```

```
dataset
```

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

399 rows × 28 columns

```
dataset = pd.get_dummies(dataset,drop_first=True,dtype=int)
```

dataset

bu	sc	sod	pot	hrmo	...	pc_normal	pcc_present	ba_present	htn_yes	dm_yes	cad_yes	appet_yes	pe_yes	ane_yes	classification_yes
82105	3.077356	137.528754	4.627244	12.518156	...	0	0	0	0	0	0	1	1	0	1
00000	0.700000	137.528754	4.627244	10.700000	...	1	0	0	0	0	0	1	0	0	1
00000	0.600000	138.000000	4.400000	12.000000	...	1	0	0	0	0	0	1	0	0	1
00000	0.700000	138.000000	3.200000	8.100000	...	1	0	0	0	0	0	1	0	1	1
00000	0.600000	137.528754	4.627244	11.800000	...	1	0	0	0	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
00000	1.300000	139.000000	3.700000	12.500000	...	1	0	0	0	0	0	1	0	0	1
00000	2.800000	137.528754	4.627244	8.700000	...	1	0	0	1	1	0	1	0	1	1
00000	6.000000	134.000000	2.700000	9.100000	...	1	0	0	1	1	0	0	0	0	1
00000	6.800000	142.000000	5.500000	8.500000	...	1	0	0	1	1	0	1	0	1	1
00000	1.000000	140.000000	5.000000	16.300000	...	1	0	0	0	0	0	1	0	0	0

3. Independent Columns:

   independent =
   dataset[["age","bp","al","su","bgr","bu","sc","sod","pot","hrmo","pcv","wc","rc","sg_b","sg_b","sg_c","sg_d","sg_d","sg_e","rbc_normal","pc_normal","pcc_present","ba_present","htn_yes","dm_yes","cad_yes","appet_yes","pe_yes","ane_yes"]]

4. Dependent Columns:
   dependent = dataset[["classification_yes"]]
5. F1, Confusion Matrix, Clf_Report and roc_auc reports

```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
print("The f1_macro value for the best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for the best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}: 0.9924946382275899
```

```
print("The Confusion Matrix:\n",cm)
```

```
The Confusion Matrix:
 [[51  0]
 [ 1 81]]
```

```
print("The report:\n",clf_report)
```

```
The report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        51
           1       1.00      0.99      0.99        82

    accuracy                           0.99       133
   macro avg       0.99      0.99      0.99       133
weighted avg       0.99      0.99      0.99       133
```

```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, grid.predict_proba(x_test)[:,1])
```

```
1.0
```

6. User Inputs to generate Future Prediction

```
age_input = float(input("Age:"))
bp_input = float(input("bp:"))
sg_b_input = int(input("sg_b 0 or 1:"))
sg_c_input = int(input("sg_c 0 or 1:"))
sg_d_input = int(input("sg_d 0 or 1:"))
sg_e_input = int(input("sg_e 0 or 1:"))
al_input = int(input("al (0 to 5):"))
su_input = int(input("su (0 to 5):"))
rbc_normal_input = int(input("rbc_normal 0 or 1:"))
pc_normal_input = int(input("pc_normal 0 or 1:"))
pcc_present_input = int(input("pcc_present 0 or 1:"))
ba_present_input = int(input("ba_present 0 or 1:"))
bgr_input = float(input("bgr:"))
bu_input = float(input("bu:"))
sc_input = float(input("sc:"))
sod_input = float(input("sod:"))
pot_input = float(input("pot:"))
hrmo_input = float(input("hrmo:"))
pcv_input = float(input("pcv:"))
wc_input = float(input("wc:"))
rc_input = float(input("rc:"))
htn_yes_input = int(input("htn_yes 0 or 1:"))
dm_yes_input = int(input("dm_yes 0 or 1:"))
cad_yes_input = int(input("cad_yes 0 or 1:"))
appet_yes_input = int(input("appet_yes 0 or 1:"))
pe_yes_input = int(input("pe_yes 0 or 1:"))
ane_yes_input = int(input("ane_yes 0 or 1:"))
Future_Prediction = grid.predict([[age_input,bp_input,sg_b_input,sg_c_input,sg_d_input,sg_e_input,al_input,su_input,rbc_normal_input,pc_normal_input,
wc_input,rc_input,htn_yes_input,dm_yes_input,cad_yes_input,appet_yes_input,pe_yes_input,ane_yes_input]])
print("Future_Prediction={}", format(Future_Prediction))
```

7. Future Prediction Output

```
Age: 43
bp: 130
sg_b 0 or 1: 0
sg_c 0 or 1: 1
sg_d 0 or 1: 0
sg_e 0 or 1: 1
al (0 to 5): 3
su (0 to 5): 4
rbc_normal 0 or 1: 1
pc_normal 0 or 1: 0
pcc_present 0 or 1: 0
ba_present 0 or 1: 1
bgr: 432.33
bu: 69.93
sc: 877.34
sod: 123.34
pot: 464.23
hrmo: 13.49
pcv: 236.89
wc: 89.400
rc: 14.460
htn_yes 0 or 1: 0
dm_yes 0 or 1: 1
cad_yes 0 or 1: 0
appet_yes 0 or 1: 1
pe_yes 0 or 1: 1
ane_yes 0 or 1: 0
Future_Prediction={} [1]
```