Requirement:

A requirement from the Hospital, Management asked us to create a predictive model which will predict the Chronic Kidney Disease (CKD) based on the several parameters. The Client has provided the dataset of the same.

1. Identify your problem statement

Disease prediction

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

Rows: 399
Columns : 28

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

Dataset has nominal dataset, using of get_dummies method, have changed to string to integer.

4. 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.

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

RandomForest

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

```
[[45  0]
 [ 1 74]]
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.99      0.99        75

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

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

```
0.9916844900066377
The f1_score value for best parameter{'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100}: 0.9916844900066377
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test) [:,1])
```

```
0.9997037037037038
```

SVM:

Not performance

DT:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

```
[[43  2]
 [ 4 71]]
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.91      0.96      0.93        45
           1       0.97      0.95      0.96        75

    accuracy                           0.95       120
   macro avg       0.94      0.95      0.95       120
weighted avg       0.95      0.95      0.95       120
```

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

```
0.9502056404230317
The f1_score value for best parameter{'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'}: 0.9502056404230317
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test) [:,1])
```

```
0.9511111111111112
```

RF:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

```
[[45  0]
 [ 1 74]]
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.99      0.99        75

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

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

```
0.9916844900066377
The f1_score value for best parameter{'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100}: 0.9916844900066377
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test) [:,1])
```

```
0.9997037037037038
```

KNN:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

```
[[42  3]
 [27 48]]
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.61      0.93      0.74        45
           1       0.94      0.64      0.76        75

    accuracy                           0.75       120
   macro avg       0.77      0.79      0.75       120
weighted avg       0.82      0.75      0.75       120
```

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

```
0.7525062656641605
The f1_score value for best parameter{'metric': 'minkowski', 'n_neighbors': 10, 'p': 1}: 0.7525062656641605
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test) [:,1])
```

```
0.8241481481481482
```

Logistic:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

```
[[43  2]
 [ 0 75]]
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        45
           1       0.97      1.00      0.99        75

    accuracy                           0.98       120
   macro avg       0.99      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

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

```
0.9832535885167464
The f1_score value for best parameter{'penalty': 'l2', 'solver': 'liblinear'}: 0.9832535885167464
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test) [:,1])
```

```
0.9973333333333334
```

NB:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

```
[[45  0]
 [ 2 73]]
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.97      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.99      0.98       120
weighted avg       0.98      0.98      0.98       120
```

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

```
0.9834018801410106
The f1_score value for best parameter{'var_smoothing': 1e-09}: 0.9834018801410106
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test) [:,1])
```

```
1.0
```

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

F1_score and ROC_AUC_Core got near to 1 values in RandomForestClassifier.