

## Problem Statement or 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

**We need to predict the Chronic Kidney Disease based on the given parameters in the dataset**

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

**Total number of columns:25**

**Total number of rows: 249**

**Independent/Input: 24 columns**

**Dependent/Output: 1 column**

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

**I have used `pd.get_dummies` which converted strings into int and standard scaler to optimize the difference between data**

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.

**I have created classification models**

**1.SVM**

**2.Decision Tree**

**3.Random Forest**

**4.Logistic Regression**

**5.KNN**

**6.Multinomial Naïve Bayes**

**7.Categorical Naïve Bayes**

**8.Complement Naïve Bayes**

**9.Gaussian Naïve Bayes**

**10.Bernoulli Naïve Bayes**

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

**Please refer below**

```
[11]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(dep, grid_predictions)
```

```
[12]: print(cm)
```

```
[[150  0]
 [ 0 249]]
```

```
[13]: from sklearn.metrics import classification_report
      clf_report = classification_report(dep, grid_predictions)
```

```
[14]: print(clf_report)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	150
1	1.00	1.00	1.00	249
accuracy			1.00	399
macro avg	1.00	1.00	1.00	399
weighted avg	1.00	1.00	1.00	399

```
[15]: from sklearn.metrics import roc_auc_score

      roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[15]: 1.0
```

```
[10]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(dep, grid_predictions)
```

```
[11]: print(cm)

[[150  0]
 [ 0 249]]
```

```
[12]: from sklearn.metrics import classification_report
      clf_report = classification_report(dep, grid_predictions)
```

```
[13]: print(clf_report)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	150
1	1.00	1.00	1.00	249
accuracy			1.00	399
macro avg	1.00	1.00	1.00	399
weighted avg	1.00	1.00	1.00	399

```
[14]: from sklearn.metrics import roc_auc_score

      roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[14]: 1.0
```

```
[11]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(dep, grid_predictions)
```

```
[12]: print(cm)
```

```
[[150  0]
 [ 0 249]]
```

```
[13]: from sklearn.metrics import classification_report
      clf_report = classification_report(dep, grid_predictions)
```

```
[14]: print(clf_report)
```

```

              precision    recall  f1-score   support

      0       1.00      1.00      1.00        150
      1       1.00      1.00      1.00        249

   accuracy          1.00
  macro avg          1.00
 weighted avg          1.00
```

```
[15]: from sklearn.metrics import roc_auc_score

      roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[15]: 1.0
```

```
[10]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(dep, grid_predictions)
```

```
[11]: print(cm)
```

```
[[150  0]
 [ 0 249]]
```

```
[12]: from sklearn.metrics import classification_report
      clf_report = classification_report(dep, grid_predictions)
```

```
[13]: print(clf_report)
```

```

              precision    recall  f1-score   support

         0       1.00      1.00      1.00        150
         1       1.00      1.00      1.00        249

 accuracy          1.00
 macro avg          1.00
 weighted avg       1.00
```

```
[14]: from sklearn.metrics import roc_auc_score

      roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[14]: 1.0
```

```
[10]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(dep, grid_predictions)
```

```
[11]: print(cm)

[[150  0]
 [  8 241]]
```

```
[12]: from sklearn.metrics import classification_report
      clf_report = classification_report(dep, grid_predictions)
```

```
[13]: print(clf_report)
```

	precision	recall	f1-score	support
0	0.95	1.00	0.97	150
1	1.00	0.97	0.98	249
accuracy			0.98	399
macro avg	0.97	0.98	0.98	399
weighted avg	0.98	0.98	0.98	399

```
[14]: from sklearn.metrics import roc_auc_score

      roc_auc_score(dep, grid.predict_proba(indep)[: ,1])
```

```
[14]: 0.9998527443105756
```

```
[9]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions)
```

```
[10]: print(cm)

[[145  5]
 [ 39 210]]
```

```
[11]: from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions)
```

```
[12]: print(clf_report)
```

	precision	recall	f1-score	support
0	0.79	0.97	0.87	150
1	0.98	0.84	0.91	249
accuracy			0.89	399
macro avg	0.88	0.91	0.89	399
weighted avg	0.91	0.89	0.89	399

```
[13]: from sklearn.metrics import roc_auc_score

roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[13]: 0.9623025435073628
```

```
[9]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions)
```

```
[10]: print(cm)

[[150  0]
 [ 1 248]]
```

```
[11]: from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions)
```

```
[12]: print(clf_report)
```

	precision	recall	f1-score	support
0	0.99	1.00	1.00	150
1	1.00	1.00	1.00	249
accuracy			1.00	399
macro avg	1.00	1.00	1.00	399
weighted avg	1.00	1.00	1.00	399

```
[13]: from sklearn.metrics import roc_auc_score

roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[13]: 1.0
```



File Edit View Run Kernel Settings Help

⏏ + ✂ 📄 📄 ▶ ■ ↺ ⏩ Code ▾

```
[9]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions)
```

```
[10]: print(cm)
```

```
[[145  5]
 [ 39 210]]
```

```
[11]: from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions)
```

```
[12]: print(clf_report)
```

	precision	recall	f1-score	support
0	0.79	0.97	0.87	150
1	0.98	0.84	0.91	249
accuracy			0.89	399
macro avg	0.88	0.91	0.89	399
weighted avg	0.91	0.89	0.89	399

```
[13]: from sklearn.metrics import roc_auc_score

roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[13]: 0.9623025435073628
```

File Edit View Run Kernel Settings Help

+

```
[9]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions)
```

```
[10]: print(cm)
```

```
[[149  1]
 [  4 245]]
```

```
[11]: from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions)
```

```
[12]: print(clf_report)
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	150
1	1.00	0.98	0.99	249
accuracy			0.99	399
macro avg	0.98	0.99	0.99	399
weighted avg	0.99	0.99	0.99	399

```
[13]: from sklearn.metrics import roc_auc_score

roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[13]: 0.996532797858099
```

```
[9]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions)
```

```
[10]: print(cm)

[[149  1]
 [  7 242]]
```

```
[11]: from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions)
```

```
[12]: print(clf_report)
```

	precision	recall	f1-score	support
0	0.96	0.99	0.97	150
1	1.00	0.97	0.98	249
accuracy			0.98	399
macro avg	0.98	0.98	0.98	399
weighted avg	0.98	0.98	0.98	399

```
[13]: from sklearn.metrics import roc_auc_score

roc_auc_score(dep,grid.predict_proba(indep)[:,:1])
```

```
[13]: 0.994016064257028
```

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

All the below algorithm has given high accuracy so used Random Forest

1.SVM

2.Decision Tree

3.Random Forest

4.Logistic Regression