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.
Machine Learning > Supervised Learning > Classification

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 col

Dependent/Output: 1 col

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

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.

Created classification models

1.SVM

2.Decision Tree

3. Logistic Regression

4. Random Forest

5.KNN

6. Naïve Bayes - Multinomial Naïve Bayes ,Categorical Naïve Bayes ,Complement Naïve Bayes ,Gaussian Naïve Bayes, Bernoulli Naïve Bayes.

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

The result screen shot below for ref,

jupyter 1.SVM Classification Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help

Code

[27]:
```
# print classification report
from sklearn.metrics import classification_report
clf_report = classification_report(y_test, grid_predictions)
```

[29]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

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

[31]:
```
print("The confusion Matrix:\n",cm)
```

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

[33]:
```
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
```

[35]:
```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[35]:
```
1.0
```

jupyter 2.Decision Tree Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code JupyterLab

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

[31]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'best'}: 0.9700283472213296
```

[33]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[50  1]
 [ 3 79]]
```

[35]:
```
print("The report:\n",clf_report)
```

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

           0       0.94      0.98      0.96        51
           1       0.99      0.96      0.98        82

    accuracy                           0.97       133
   macro avg       0.97      0.97      0.97       133
weighted avg       0.97      0.97      0.97       133
```

[37]:
```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[37]:
```
0.9719033955045432
```

jupyter **3.Logistic Classification** Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code

[27]:
```
# print classification report
from sklearn.metrics import classification_report
clf_report = classification_report(y_test, grid_predictions)
```

[29]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 0.9924946382275899
```

[31]:
```
print("The confusion Matrix:\n",cm)
```

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

[33]:
```
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
```

[35]:
```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[35]:
```
1.0
```

jupyter **4.RF Classification** Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code JupyterLab

```
f1_macro=f1_score(dep,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 100}: 1.0
```

[37]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[150   0]
 [  0 249]]
```

[39]:
```
print("The report:\n",clf_report)
```

```
The 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
```

[41]:
```
from sklearn.metrics import roc_auc_score

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

[41]:
```
1.0
```

jupyter 5.KNN Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code

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

[35]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'algorithm': 'auto', 'n_neighbors': 5, 'weights': 'distance'}: 0.940494593126172
```

[37]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 8 74]]
```

[39]:
```
print("The report:\n",clf_report)
```

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

           0       0.86      1.00      0.93        51
           1       1.00      0.90      0.95        82

    accuracy                           0.94       133
   macro avg       0.93      0.95      0.94       133
weighted avg       0.95      0.94      0.94       133
```

[41]:
```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[41]:
```
1.0
```

jupyter 6.Naive baye's Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code

```
X_test = sc.transform(X_test)
```

[25]:
```
from sklearn.model_selection import GridSearchCV
```

[27]:
```
from sklearn.naive_bayes import MultinomialNB
param_grid = {'alpha':[0.1,1,10,100 ]}
grid = GridSearchCV(MultinomialNB(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

#fitting the model for Grid search
grid.fit(indep, dep)
grid_predictions  = grid.predict(indep)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions )
from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions )
print(clf_report)
print(cm)
from sklearn.metrics import roc_auc_score
roc_auc_score(dep, grid.predict_proba(indep)[:,1])
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
              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

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

[27]:
```
0.9623025435073628
```

jupyter 6.Naive baye's Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code

[29]:
```
from sklearn.naive_bayes import BernoulliNB

param_grid = {'alpha':[0.1,1,10,100 ]}

grid = GridSearchCV(BernoulliNB(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

#fitting the model for Grid search
grid.fit(indep, dep)
grid_predictions  = grid.predict(indep)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions )
from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions )
print(clf_report)
print(cm)
from sklearn.metrics import roc_auc_score
roc_auc_score(dep, grid.predict_proba(indep)[:,1])
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
              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

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

[29]:
```
0.994016064257028
```

jupyter 6.Naive baye's Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code

[31]:
```
from sklearn.naive_bayes import CategoricalNB

param_grid = {'alpha':[0.1,1,10,100 ]}

grid = GridSearchCV(CategoricalNB(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')


#fitting the model for Grid search
grid.fit(indep, dep)
grid_predictions  = grid.predict(indep)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions )
from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions )
print(clf_report)
print(cm)
from sklearn.metrics import roc_auc_score
roc_auc_score(dep, grid.predict_proba(indep)[:,1])
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
              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

[[150   0]
 [  1 248]]
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:1051: UserWarning: One or more of th
nan nan]
  warnings.warn(
```

[31]:
```
1.0
```

jupyter 6.Naive baye's Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code JupyterLab

[31]:
```
1.0
```

[33]:
```
from sklearn.naive_bayes import ComplementNB

param_grid = {'alpha':[0.1,1,10,100 ]}

grid = GridSearchCV(ComplementNB(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

#fitting the model for Grid search
grid.fit(indep, dep)
grid_predictions  = grid.predict(indep)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions )
from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions )
print(clf_report)
print(cm)
from sklearn.metrics import roc_auc_score
roc_auc_score(dep, grid.predict_proba(indep)[:,1])
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
              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

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

[33]:
```
0.9623025435073628
```

jupyter 6.Naive baye's Classification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Code

[35]:
```
from sklearn.naive_bayes import GaussianNB

param_grid = {'priors': [None,  # None lets the algorithm assume uniform priors
                         [0.2, 0.8],  # Priors for two classes: class 0 - 0.2, class 1 - 0.8
                         [0.5, 0.5],  # Equal priors for both classes
                         [0.7, 0.3]]} # Priors for two classes: class 0 - 0.7, class 1 - 0.3

grid = GridSearchCV(GaussianNB(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

#fitting the model for Grid search
grid.fit(indep, dep)
grid_predictions  = grid.predict(indep)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(dep, grid_predictions )
from sklearn.metrics import classification_report
clf_report = classification_report(dep, grid_predictions )
print(clf_report)
print(cm)
from sklearn.metrics import roc_auc_score
roc_auc_score(dep, grid.predict_proba(indep)[:,1])
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
              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

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

[35]:
```
0.996532797858099
```

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

Finally we used to select **Random Forest is best model** with below settings

The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 1
00}: 1.0