Hope Artificial Intelligence

Classification Assignment

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

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

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

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

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

Note: Mentioned points are necessary, kindly mail your document as well as .ipynb (code file) with respective name.

 Sub file name also should be properly named for Example (SVM_Ramisha_Assi-5.ipynb)

Communication is important (How you are representing the document.).

Kindly uploaded in the Github and Share it with us.

1.) Identify your problem statement

- Given Input and output in Numerical value. So, it comes under the **Machine Learning Process.**

- Given input and output data is very clear. So, it comes under the **Supervised learning Method.**

- Given output is Categorical Data. So, It is comes under the **Classification Algorithm.**

  The client is asking us to create a model for Chronic Kidney Disease (CKD) prediction by using the dataset provided.

  This AI Model can be named as "**CKD AI Care**"

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

- Rows : 399 & Column : 25

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

- **Nominal Data – One Hot Encoding** method is used for converting string to number.

4.) Develop a good model with good evaluation metric. You can use any machine learning algorithm; you can create many models.

Here I have used the classification algorithms. The given below Classification Algorithms used for creating a model for CKD Prediction.

1. SVM Classification.
2. Decision Tree Classification.
3. Random Forest Classification.
4. Logistic Regression.
5. K-Nearest Neighbours (KNN) Classification.
6. Naive Bayes Classification.

1. SVM CLASSIFICATION

In [16]:
```
from sklearn.metrics import f1_score

f1_macro=f1_score(y_test,y_pred,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'}: 0.9834018801410106
```

In [17]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 2 73]]
```

In [18]:
```
print("The report:\n",clf_report)
```

```
The 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
```

In [19]:
```
#AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
from sklearn.metrics import roc_auc_score

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

Out[19]: 0.9997037037037036

2. DECISION TREE CLASSIFICATION

In [16]:
```
from sklearn.metrics import f1_score

f1_macro=f1_score(y_test,y_pred,average='weighted')

print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'gini', 'max_features': 'log2', 'splitter': 'random'}: 0.9751481237656352
```

In [17]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 3 72]]
```

In [18]:
```
print("The report:\n",clf_report)
```

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

           0       0.94      1.00      0.97        45
           1       1.00      0.96      0.98        75

    accuracy                           0.97       120
   macro avg       0.97      0.98      0.97       120
weighted avg       0.98      0.97      0.98       120
```

In [19]:
```
#AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
from sklearn.metrics import roc_auc_score

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

Out[19]: 0.98

3. RANDOM FOREST CLASSIFICATION

In [17]:
```
from sklearn.metrics import f1_score

f1_macro=f1_score(y_test,y_pred,average='weighted')

print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 10}: 0.9833333333333335
```

In [18]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[44  1]
 [ 1 74]]
```

In [19]:
```
print("The report:\n",clf_report)
```

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

           0       0.98      0.98      0.98        45
           1       0.99      0.99      0.99        75

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

In [20]:
```
#AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
from sklearn.metrics import roc_auc_score

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

Out[20]: 0.9995555555555555

4. LOGISTIC REGRESSION

In [16]:
```
from sklearn.metrics import f1_score

f1_macro=f1_score(y_test,y_pred,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': 'lbfgs'}: 0.9916844900066377
```

In [17]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 1 74]]
```

In [18]:
```
print("The report:\n",clf_report)
```

```
The 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
```

In [19]:
```
#AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
from sklearn.metrics import roc_auc_score

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

Out[19]: 1.0

5. K-NEAREST NEIGHBOURS (KNN) CLASSIFICATION

In [16]:
```
from sklearn.metrics import f1_score

f1_macro=f1_score(y_test,y_pred,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', 'metric': 'minkowski', 'n_neighbors': 1, 'p': 2, 'weights': 'unifor
m'}: 0.9834018801410106
```

In [17]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 2 73]]
```

In [18]:
```
print("The report:\n",clf_report)
```

```
The 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
```

In [19]:
```
#AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
from sklearn.metrics import roc_auc_score

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

Out[19]:
```
0.9866666666666667
```

6. NAIVE BAYES CLASSIFICATION

In [27]:
```
from sklearn.naive_bayes import GaussianNB
Classifier=GaussianNB()
Classifier=Classifier.fit(x_train,y_train)
Classifier
y_pred=Classifier.predict(x_test)
y_pred
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,y_pred)
print("GAUSSIAN NAIVE BAYES : ")
print("The confusion Matrix:\n",cm)
print("The clf report:\n",clf_report)
```

```
GAUSSIAN NAIVE BAYES :
The confusion Matrix:
 [[45  0]
 [ 2 73]]
The 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
```

```
C:\Users\Go\Anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [28]:
```
from sklearn.naive_bayes import MultinomialNB
Classifier=MultinomialNB()
Classifier=Classifier.fit(x_train,y_train)
Classifier
y_pred=Classifier.predict(x_test)
y_pred
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,y_pred)
print("MULTINOMIAL NAIVE BAYES : ")
print("The confusion Matrix:\n",cm)
print("The clf report:\n",clf_report)
```

```
MULTINOMIAL NAIVE BAYES :
The confusion Matrix:
 [[44  1]
 [22 53]]
The clf report:
              precision    recall  f1-score   support

           0       0.67      0.98      0.79        45
           1       0.98      0.71      0.82        75

    accuracy                           0.81       120
   macro avg       0.82      0.84      0.81       120
weighted avg       0.86      0.81      0.81       120
```

```
C:\Users\Go\Anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [29]:
```
from sklearn.naive_bayes import ComplementNB
Classifier=ComplementNB()
Classifier=Classifier.fit(x_train,y_train)
Classifier
y_pred=Classifier.predict(x_test)
y_pred
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,y_pred)
print("COMPLEMENT NAIVE BAYES : ")
print("The confusion Matrix:\n",cm)
print("The clf report:\n",clf_report)
```

```
COMPLEMENT NAIVE BAYES :
The confusion Matrix:
 [[44  1]
 [22 53]]
The clf report:
              precision    recall  f1-score   support

           0       0.67      0.98      0.79        45
           1       0.98      0.71      0.82        75

    accuracy                           0.81       120
   macro avg       0.82      0.84      0.81       120
weighted avg       0.86      0.81      0.81       120
```

```
C:\Users\Go\Anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [30]:
```
from sklearn.naive_bayes import BernoulliNB
Classifier=BernoulliNB()
Classifier=Classifier.fit(x_train,y_train)
Classifier
y_pred=Classifier.predict(x_test)
y_pred
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,y_pred)
print("BERNOULLI NAIVE BAYES : ")
print("The confusion Matrix:\n",cm)
print("The clf report:\n",clf_report)
```

```
BERNOULLI NAIVE BAYES :
The confusion Matrix:
 [[45  0]
 [ 8 67]]
The clf report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        45
           1       1.00      0.89      0.94        75

    accuracy                           0.93       120
   macro avg       0.92      0.95      0.93       120
weighted avg       0.94      0.93      0.93       120
```

```
C:\Users\Go\Anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

5.) FINAL RESULT:

The **Logistic Regression** Classification Algorithm gives the best result among all the algorithms used for this given dataset.

1. **The confusion Matrix:**

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

2. **The classification 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
```

3. **AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve Score:**

```
roc_auc_score: 1.0
```

4. **The f1 macro value for best parameter {'penalty': 'l2', 'solver': 'lbfgs'}:**

```
f1_macro value: 0.9916844900066377
```