Assignment - Classification Algorithm

1. The problem statement for this task is to develop a predictive model that can accurately estimate the **chronic kidney disease** (CKD) based on several input parameters.

2. The total number of rows of the dataset is 399 and the total number of columns is 25. For the model, Classification ais taken as a predictive value or output parameter and the remaining attributes are the input parameters.

3. I'm using One Hot Encoding method to convert the categorical (nominal) data, sg, rbc, pc, pcc, ba, htn, dm, cad, appet, pe, ane, and classification into numerical values.

4. Finalized Model 1:

   Logistic Regression - Phase 01 - Model Creation
   https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Final%20Model.ipynb

   Logistic Regression - Phase 02 - Deployment
   https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Deployment-Final%20Model.ipynb

Finalized Model 1:

SVM Classification - Phase 01 - Model Creation
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Final%20Model.ipynb

SVM Classification - Phase 02 - Deployment
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Deployment-Final%20Model.ipynb

5.

Logistic Regression

In [13]:
```
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
```

In [14]:
```
print("The confusion Matrix:\n", cm)
```

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

In [15]:
```
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
```

In [16]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[16]:
```
1.0
```

SVM Classification

In [13]:
```
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
```

In [14]:
```
print("The confusion Matrix:\n", cm)
```

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

In [15]:
```
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
```

In [16]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[16]:
```
1.0
```

Decision Tree Classification

In [13]:
```
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 {'class_weight': 'balanced', 'criterion': 'gini', 'max_features': 'sqrt', 'splitter': 'ra
ndom'}: 0.955283779067923
```

In [14]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 6 76]]
```

In [15]:
```
print("The report:\n", clf_report)
```

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

           0       0.89      1.00      0.94        51
           1       1.00      0.93      0.96        82

    accuracy                           0.95       133
   macro avg       0.95      0.96      0.95       133
weighted avg       0.96      0.95      0.96       133
```

In [16]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[16]:
```
0.9634146341463414
```

Random Forest Classification

In [13]:
```
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 {'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': 'log2', 'n_estimator
s': 10}: 0.9924946382275899
```

In [14]:
```
print("The confusion Matrix:\n", cm)
```

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

In [15]:
```
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
```

In [16]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[16]:
```
0.9998804399808703
```

K – Nearest Neighbor Classification

In [13]:
```
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', 'metric': 'minkowski', 'metric_params': None, 'n_neighbors': 3, 'we
ights': 'uniform'}: 0.9626932787797391
```

In [14]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 5 77]]
```

In [15]:
```
print("The report:\n", clf_report)
```

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

           0       0.91      1.00      0.95        51
           1       1.00      0.94      0.97        82

    accuracy                           0.96       133
   macro avg       0.96      0.97      0.96       133
weighted avg       0.97      0.96      0.96       133
```

In [16]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[16]: 0.998804399808704

Navies' Bayes – Gaussian Classification

In [11]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[149   1]
 [  4 245]]
```

In [12]:
```
print("The report:\n", clf_report)
```

```
The 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
```

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(dependent, grid_predictions, average = 'weighted')
print("The f1_macro value", f1_macro)
```

```
The f1_macro value 0.9874927342358296
```

In [14]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[14]: 0.996532797858099

Navies' Bayes – Multinomial Classification

In [11]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[145   5]
 [ 39 210]]
```

In [12]:
```
print("The report:\n", clf_report)
```

```
The 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
```

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(dependent, grid_predictions, average = 'weighted')
print("The f1_macro value", f1_macro)
```

```
The f1_macro value 0.8912968721618211
```

In [14]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[14]:
```
0.9623025435073628
```

Navies' Bayes – Complement Classification

In [11]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[145   5]
 [ 39 210]]
```

In [12]:
```
print("The report:\n", clf_report)
```

```
The 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
```

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(dependent, grid_predictions, average = 'weighted')
print("The f1_macro value", f1_macro)
```

```
The f1_macro value 0.8912968721618211
```

In [14]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[14]:
```
0.9623025435073628
```

Navies' Bayes – Bernoulli Classification

In [11]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[149   1]
 [  7 242]]
```

In [12]:
```
print("The report:\n", clf_report)
```

```
The 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
```

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(dependent, grid_predictions, average = 'weighted')
print("The f1_macro value", f1_macro)
```

```
The f1_macro value 0.9800241876810629
```

In [14]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[14]:
```
0.994016064257028
```

Navies' Bayes – Categorical Classification

In [20]:
```
print("The confusion Matrix:\n", cm)
```

```
The confusion Matrix:
 [[150   0]
 [  1 248]]
```

In [21]:
```
print("The report:\n", clf_report)
```

```
The 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
```

In [22]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(dependent, grid_predictions, average = 'weighted')
print("The f1_macro value", f1_macro)
```

```
The f1_macro value 0.9974953761738116
```

In [23]:
```
from sklearn.metrics import roc_auc_score
#Receiver Operating Characteristic_Area Under Curve
#if the curve is L shape then the classification is 100% correct

roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
# (: = rows) & (1 = Column)
#grid.predict_proba(X_test)->inbuilt function(probability compulsory)
```

Out[23]:
```
1.0
```

6. Final Model

```
In [13]: 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

In [14]: print("The confusion Matrix:\n", cm)

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

In [15]: 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

In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 1.0
```

Model 1: Logistic Regression

Logistic Regression - Phase 01 - Model Creation
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Final%20Model.ipynb

Logistic Regression - Phase 02 - Deployment

https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Deployment-Final%20Model.ipynb

```
In [13]: 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

In [14]: print("The confusion Matrix:\n", cm)

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

In [15]: 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

In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 1.0
```

Model 2: SVM Classification

SVM Classification - Phase 01 - Model Creation
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Final%20Model.ipynb

SVM Classification - Phase 02 - Deployment

https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Deployment-Final%20Model.ipynb

These two models are the final models, because both models have the same and highest **Precision** value, **Recall** value, **F1-score** value, **Accuracy** value, **Macro average** value and **Weighted average** value. Especially both models have the highest Receiver Operating Characteristic Area Under Curve Score (**roc_auc_score**).