Predict Chronic Kidney Disease(CKD)

Algorithms	Accuracy	Macro Avg_F1	Weighted Avg_F1	Roc Auc
SVM_CLASSIFICATION	**0.99**	**0.99**	**0.99**	**1**
LOGISTIC_REGRESSION_CLASSIFICATION	0.98	0.98	0.99	1
DECISION_TREE_CLASSIFICATION	0.98	0.98	0.99	0.9878
RANDOM_FOREST_CLASSIFICATION	0.97	0.97	0.97	0.9988
KNN_CLASSIFICATION	0.96	0.96	0.96	0.9695
BERNOULLI_NB	0.94	0.94	0.94	0.9967
COMPLEMENT_NB	0.82	0.82	0.82	0.9151
MULTINOMIAL_NB	0.82	0.82	0.82	0.9151
GAUSSIAN_NB	0.98	0.98	0.98	1

1) Problem Statement is to Predict Chronic Kidney Disease(CKD)
2) There are 399 Rows and 25 Columns in the dataset
3) There are 12 categorical columns(nominal) , Converted to Numerical
4) Final Model is SVM_CLASSIFICATION
5) SVM_CLASSIFICATION , because it score all the metrics 99% and ROC Score is 1

BeatModelEvaluation_CKD

SVM_CLASSIFICATION

In [14]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macroe Value for Best Parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macroe Value for Best Parameter {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}: 0.9924946382275899
```

In [15]:
```
print("The Confusion Matrix:\n",cm)
```

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

In [16]:
```
# SVC_Classification_CKD
print("The report:\n",clf_report)
# https://scikit-learn.org/stable/modules/model_evaluation
```

```
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 [17]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

Out[17]:
```
1.0
```

RANDOM_FOREST_CLASSIFICATION

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macroe Value for Best Parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macroe Value for Best Parameter {'max_depth': 3, 'max_features': 1, 'min_samples_leaf': 2, 'min_samples_split': 0.1, 'n_
estimators': 15}: 0.9699248120300752
```

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

```
The Confusion Matrix:
 [[49  2]
 [ 2 80]]
```

In [15]:
```
#RF_Classification-Gridcv_CKD
print("The report:\n",clf_report)
# https://scikit-learn.org/stable/modules/model_evaluation
```

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

           0       0.96      0.96      0.96        51
           1       0.98      0.98      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
```

In [16]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

Out[16]:
```
0.9988043998087041
```

DECISION_TREE_CLASSIFICATION

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macroe Value for Best Parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macroe Value for Best Parameter {'max_depth': 31, 'max_features': 3, 'min_samples_leaf': 1, 'min_samples_split': 2}: 0.9
850141736106648
```

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

```
The Confusion Matrix:
 [[51  0]
 [ 2 80]]
```

In [15]:
```
#DT_Classification-Gridcv_Gridcv_CKD
print("The report:\n",clf_report)
# https://scikit-learn.org/stable/modules/model_evaluation
```

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

           0       0.96      1.00      0.98        51
           1       1.00      0.98      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.99      0.98       133
weighted avg       0.99      0.98      0.99       133
```

In [16]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

Out[16]:
```
0.9878048780487805
```

LOGISTIC_REGRESSION_CLASSIFICATION

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macroe Value for Best Parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macroe Value for Best Parameter {'C': 1.5, 'fit_intercept': 'True', 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'l
iblinear'}: 0.9850141736106648
```

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

```
The Confusion Matrix:
 [[51  0]
 [ 2 80]]
```

In [15]:
```
#LogisticRegression_Classification_Gridcv_CKD
print("The report:\n",clf_report)
# https://scikit-learn.org/stable/modules/model_evaluation
```

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

           0       0.96      1.00      0.98        51
           1       1.00      0.98      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.99      0.98       133
weighted avg       0.99      0.98      0.99       133
```

In [16]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

Out[16]: 1.0

KNN_CLASSIFICATION

In [13]:
```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macroe Value for Best Parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macroe Value for Best Parameter {'n_neighbors': 1, 'p': 1}: 0.9626932787797391
```

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

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

In [15]:
```
#KNN_Classification-Gridcv_CKD
print("The report:\n",clf_report)
# https://scikit-learn.org/stable/modules/model_evaluation
```

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

Out[16]:
```
0.9695121951219512
```

NAÏVE_BAYES

```python
#BernoulliNB
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
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(clf_report)
print(cm)
```

```
              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

[[51  0]
 [ 8 74]]
```

```python
#ComplementNB
from sklearn.naive_bayes import ComplementNB
classifier = ComplementNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
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(clf_report)
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.68      0.98      0.81        51
           1       0.98      0.72      0.83        82

    accuracy                           0.82       133
   macro avg       0.83      0.85      0.82       133
weighted avg       0.87      0.82      0.82       133

[[50  1]
 [23 59]]
```

NAÏVE_BAYES

```python
#MultinomialNB
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
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(clf_report)
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.68      0.98      0.81        51
           1       0.98      0.72      0.83        82

    accuracy                           0.82       133
   macro avg       0.83      0.85      0.82       133
weighted avg       0.87      0.82      0.82       133

[[50  1]
 [23 59]]
```

```python
#GaussianNB
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
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(clf_report)
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        51
           1       1.00      0.96      0.98        82

    accuracy                           0.98       133
   macro avg       0.97      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133

[[51  0]
 [ 3 79]]
```