Classification Assignment – CKD

Results:

1. Based on the data shared this falls under Domain – Machine Learning – Classification
2. Total – 399 rows and 25 columns of data present in CKD.csv
3. Converting the columns – sg , rbc, pc, pcc, ba,pe from string to numeric
4. Predicted the results using the below Classification Algorithms
    a. Decision Tree Classification Algorithm -99% Accuracy
    b. Logistic Regression Algorithm – 98% Accuracy
    c. Naive Bayes – 94% Accuracy
    d. KNN Algorithm- 94% Accuracy
    e. Random Forest Classification Algorithm -98% Accuracy

Choosing Decision Tree Classification Algorithm *as the best one for this Data set for CKD.csv , as this provides a prediction with 99% Accuracy*

- **Prediction Results with Decision Tree Classification Algorithm**
    - ✓ Overall Accuracy is 99% is predicted using **Decision Tree Classification Algorithm**

[12]:
```
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',
```

[13]:
```
print("The confusion Matrix:\n",cm)
```

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

[14]:
```
print("The report:\n",clf_report)
```

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

       False       0.98      1.00      0.99        51
        True       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
```

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

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

[15]:
```
np.float64(0.9939024390243902)
```

- **Prediction Results with Logistic Regression Algorithm**
  - ✓ Overall Accuracy is 98% is predicted using **Logistic Regression Algorithm**

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

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

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

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

       False       0.96      1.00      0.98        51
        True       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
```

- **Prediction Results with Naive Bayes**
  - ✓ Overall Accuracy is 94% is predicted using BernoulliNB
  - ✓ Out of all Classification, Truly Classified displays as 1.00
  - ✓ Out of all Classification, Correctly classified as true is 90%

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

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

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

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

       False       0.86      1.00      0.93        51
        True       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
```

- **Prediction Results with KNN Algorithm**

- ✓ Overall Accuracy is 94% is predicted using **KNN Algorithm**

```
3]: from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    cm
```

```
3]: array([[51,  0],
           [ 8, 74]])
```

```
4]: from sklearn.metrics import classification_report
    clf_report = classification_report(y_test, y_pred)
    print(clf_report)
```

```
              precision    recall  f1-score   support

       False       0.86      1.00      0.93        51
        True       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
```

- **Prediction Results with Random Forest Algorithm**

  - ✓ Overall Accuracy is 98% is predicted using **Random Forest Algorithm**

[12]:
```
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': 'gini', 'max_features': 'log2', 'n_est
```

[13]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[50  1]
 [ 1 81]]
```

[14]:
```
print("The report:\n",clf_report)
```

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

       False       0.98      0.98      0.98        51
        True       0.99      0.99      0.99        82

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