Chronic Kidney Disease Prediction

1) Data available – CKD.csv file having 399 rows and 25 columns
2) Few columns converted from string to Boolean (nominal data conversion) with file having 399 rows and 28 columns
3) Best model with Grid Search Cross Validation for Support vector classifier

[9]:
```
#best parameters after tuning
print('The best parameters for SVC',grid.best_params_)
grid_predictions=grid.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,grid_predictions)
print('The confusion matrix:\n',cm)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,grid_predictions)
print('The classification report:\n',clf_report)
```

```
The best parameters for SVC {'C': 1000, 'gamma': 'auto', 'kernel': 'linear'}
The confusion matrix:
 [[49  2]
 [ 1 81]]
The classification report:
              precision    recall  f1-score   support

       False       0.98      0.96      0.97        51
        True       0.98      0.99      0.98        82

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

[10]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[10]:
```
np.float64(0.9990435198469632)
```

4) Best model with Grid Search Cross Validation for Decision Tree classifier

[9]:
```
#best parameters after tuning
print('The best parameters for DTC',grid.best_params_)
grid_predictions=grid.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,grid_predictions)
print('The confusion matrix:\n',cm)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,grid_predictions)
print('The classification report:\n',clf_report)
```

```
The best parameters for DTC {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'}
The confusion matrix:
 [[51  0]
 [ 6 76]]
The classification report:
              precision    recall  f1-score   support

       False       0.89      1.00      0.94        51
        True       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
```

[10]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[10]:
```
np.float64(0.9634146341463414)
```

5) Best model with Grid Search Cross Validation for Random Forest classifier

[9]:
```
#best parameters after tuning
print('The best parameters for RFC',grid.best_params_)
grid_predictions=grid.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,grid_predictions)
print('The confusion matrix:\n',cm)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,grid_predictions)
print('The classification report:\n',clf_report)
```

```
The best parameters for RFC {'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 100}
The confusion matrix:
 [[51  0]
 [ 1 81]]
The classification 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
```

[10]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[10]:
```
np.float64(0.9997608799617408)
```

6) Best model with Grid Search Cross Validation for Logistic Regression classifier

[9]:
```
#best parameters after tuning
print('The best parameters for LRC',grid.best_params_)
grid_predictions=grid.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,grid_predictions)
print('The confusion matrix:\n',cm)
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,grid_predictions)
print('The classification report:\n',clf_report)
```

```
The best parameters for LRC {'penalty': 'l2', 'solver': 'liblinear'}
The confusion matrix:
 [[49  2]
 [ 0 82]]
The classification report:
              precision    recall  f1-score   support

       False       1.00      0.96      0.98        51
        True       0.98      1.00      0.99        82

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

[10]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
```

[10]:
```
np.float64(0.9990435198469632)
```

7) The overall best model that can be deployed is with Grid Search Cross validation for Random Forest classifier with 0.99 f1_score weighted average