CLASSIFICATION ASSIGNMENT

GRID

FEBRUARY 2, 2024

ABINAYA RAJALINGAM

1.)Identify your problem statement

- Dataset provided by hospital management to create a predictive model for predicting the disease Chronic Kidney Disease (CKD).
- Dataset contains patient health details, with those details we have to create a model and predict the patient who will be affected by CKD in future.

  Stage 1 – Machine Learning
  Stage 2 – Supervised Learning
  Stage 3 - Classification

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

- Dataset contains 399 rows and 25 columns
- 24 input columns and 1 output column

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

- Standard scaler is the method used as pre-processing
- Converted categorical data into nominal data, after converting dataset contains 399 rows and 28 columns, which means 27 input column and 1 output column

4.) Develop a good model with good evaluation metric. You can use anymachine learning algorithm; you can create many models. Finally, you have to come up with final model.

- SVM Classifier
- Decision Tree Classifier
- Random Forest Classifier
- Logistic Regression
- KNN Classifier

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

1.SVM classifier

```
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'} 0.9924946382275899
```

```
print(cm)
```

```
[[51  0]
 [ 1 81]]
```

```
print(clf_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
```

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

```
1.0
```

2.Random Forest Classifier

```
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': 100} 0.9849624060150376
```

```
print(cm)
```

```
[[50  1]
 [ 1 81]]
```

```
print(clf_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
```

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

```
0.9997608799617408
```

3.Decision Tree Classifier

```
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': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'} 0.9478851104269762
```

```
print(cm)
```

```
[[51  0]
 [ 7 75]]
```

```
print(clf_report)
```

```
              precision    recall  f1-score   support

       False       0.88      1.00      0.94        51
        True       1.00      0.91      0.96        82

    accuracy                           0.95       133
   macro avg       0.94      0.96      0.95       133
weighted avg       0.95      0.95      0.95       133
```

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

```
0.9573170731707317
```

4.Logistic Regression

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,y_pred,average='weighted')
print("The f1_macro value for the best parameter:{}".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for the best parameter:{'penalty': 'l2', 'solver': 'newton-cg'} 0.9916844900066377
```

```
print(cm)
```

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

```
print(clf_report)
```

```
              precision    recall  f1-score   support

       False       0.98      1.00      0.99        45
        True       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
```

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

```
1.0
```

5.KNN Classifier

```
print(cm)
```

```
[[45  0]
 [ 6 69]]
```

```
print(clf_report)
```

```
              precision    recall  f1-score   support

       False       0.88      1.00      0.94        45
        True       1.00      0.92      0.96        75

    accuracy                           0.95       120
   macro avg       0.94      0.96      0.95       120
weighted avg       0.96      0.95      0.95       120
```

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,y_pred,average='weighted')
print("The f1_macro value for the best parameter:{}".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for the best parameter:{'metric': 'minkowski', 'n_neighbors': 5, 'p': 1, 'weights': 'uniform'} 0.95052083333
33334
```

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

```
0.9995555555555555
```

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

- I have chosen the best model as SVM Classifier because its f1_score is 0.99249 and roc_auc_score is 1.0 which values are best and higher than the other models.
- Type -1 error is also not present and Type-2 error is least than other models.
- Logistic Regression is also best model with roc score but I have chosen SVM Classifier as the best when compared with f1_score.