PRIYANKA

Hope Artificial Intelligence

Classification Assignment

Problem Statement or Requirement:

A requirement from the Hospital, Management asked us to create a predictive model which will predict the Chronic Kidney Disease (CKD) based on the several parameters. The Client has provided the dataset of the same.

1.) Identify your problem statement

- Domain Selection - Machine Learning
- Learning Selection- Supervised Learning
- Classification/Regression - Classification

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

Rows – 399
Columns - 25

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

I performed one-hot encoding on the nominal data using pd.get_dummies to convert categorical variables into multiple binary columns.

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

Decision Tree

```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test, grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        75

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

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

DECISION TREE

```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test, grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        75

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

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

```
1.0
```

RANDOM FOREST

[13]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test, grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       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
```

[14]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions, average='weighted')
print("The f1_macro  value for best parameters {}:" .format(grid.best_params_), f1_macro)
```

```
The f1_macro  value for best parameters {'criterion': 'entropy', 'n_estimators': 50}: 0.9916844900066377
```

[15]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, grid.predict_proba(x_test)[:,1])
```

[15]:
```
0.9997037037037038
```

SVM

[24]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test, grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        45
           1       1.00      0.96      0.98        75

    accuracy                           0.97       120
   macro avg       0.97      0.98      0.97       120
weighted avg       0.98      0.97      0.98       120
```

[25]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions, average='weighted')
print("The f1_macro  value for best parameters {}:" .format(grid.best_params_), f1_macro)
```

```
The f1_macro  value for best parameters {'gamma': 'scale', 'kernel': 'linear'}: 0.9751481237656352
```

[26]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, grid.predict_proba(x_test)[:,1])
```

[26]:
```
0.9994074074074074
```

LOGISTIC REGRESSION

[17]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test, grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       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
```

[18]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions, average='weighted')
print("The f1_macro  value for best parameters {}:" .format(grid.best_params_), f1_macro)
```

```
The f1_macro  value for best parameters {'penalty': 'l2', 'solver': 'lbfgs'}: 0.9916844900066377
```

[23]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, grid.predict_proba(x_test)[:,1])
```

[23]:
```
1.0
```

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

DECISION TREE – Almost all of them are good models.. The Decision Tree is a good model because the ROC score, recall, precision, and F1 score all indicate a model performance of 1. Therefore, it is considered a good model.


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

```
1.0
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test, grid_predictions)
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        75

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```