dt-classification-1

April 5, 2024

0.1 Predicting Diabetes using Decision Tree

[1]:

```
## importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[2]:

```
## Loading dataset
data = pd.read_csv(r'C:\Users\ntpc\Desktop\diabetes.csv')
data.head()
```

[2]:

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[3]:

```
data.dtypes
```

[3]:

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

[4]:

```
data.isnull().sum()
```

[4]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[5]:

```
# Splitting the data into features (X) and target variable (y)
X = data.drop('Outcome', axis=1)
y = data['Outcome']
```

[6]:

```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[7]:

```
dt_classifier = DecisionTreeClassifier()
```

[8]:

```
# Defining hyperparameter grid for grid search
param_grid = {
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

[9]:

```
grid_search = GridSearchCV(dt_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

[9]:

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7, 10],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10]})
```

[10]:

```
best_params = grid_search.best_params_
```

[11]:

```
best_dt_classifier = DecisionTreeClassifier(**best_params)
best_dt_classifier.fit(X_train, y_train)
```

[11]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=4)
```

[12]:

```
# Making predictions on the testing data
y_pred = best_dt_classifier.predict(X_test)
```

[13]:

```python
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Best Hyperparameters: {best_params}')
print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(report)
```

```
Best Hyperparameters: {'max_depth': 3, 'min_samples_leaf': 4,
'min_samples_split': 2}
Accuracy: 0.7597402597402597
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.84      0.82        99
           1       0.68      0.62      0.65        55

    accuracy                           0.76       154
   macro avg       0.74      0.73      0.73       154
weighted avg       0.76      0.76      0.76       154
```