Hyperparameter Tuning for kNN for Predicting Heart Disease

```
from google.colab import files
upload = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving heart (2).csv to heart (2) (1).csv
```

```
import pandas as pd
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV #cross validation to select the optimal params for knn
# k distance will be considered under a given set of values.
```

```
df = pd.read_csv('heart (2).csv')
```

```
print(df.info())
print(df.shape)
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
None
(303, 14)
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0   
1   37    1   2       130   250    0        1      187      0      3.5      0   
2   41    0   1       130   204    0        0      172      0      1.4      2   
3   56    1   1       120   236    0        1      178      0      0.8      2   
4   57    0   0       120   354    0        1      163      1      0.6      2   

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

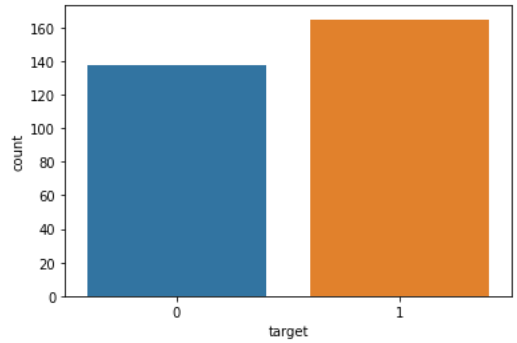
```
df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
```

```
ca          0
thal        0
target      0
dtype: int64
```

```
sns.countplot(df['target'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f029334b670>
```




```
x=df.drop(columns=['target'])
y=df['target']
knn=KNeighborsClassifier()
train_X,test_X,train_Y,test_Y=train_test_split(x,y,train_size=0.2, random_state=100)
```

train a new knn model using gscv

```
knn.fit(train_X,train_Y)
```

```
KNeighborsClassifier()
```

```
y_pred=knn.predict(test_X)
```

```
print(classification_report(test_Y,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.53      0.51      0.52       108
               1       0.62      0.64      0.63       135

        accuracy                           0.58       243
       macro avg       0.57      0.57      0.57       243
    weighted avg       0.58      0.58      0.58       243
```

```
roc_auc_score(test_Y,y_pred)
```

```
0.5731481481481482
```

```
#set params using gscv/ set leaf size to 1 neigh to 10 and dist metric to 1,2 when 1 manhattan dis 2 euclidian dis. gscv to search the optima
leaf_size=list(range(1,15))
n_neighbors=list(range(1,10))
p=[1,2]
hyperparameters=dict(leaf_size=leaf_size,n_neighbors=n_neighbors,p=p)
```

```
knn_2 = KNeighborsClassifier()
clf = GridSearchCV(knn_2, hyperparameters, cv=10, scoring = 'roc_auc')
best_model = clf.fit(x,y)
```

```
print(hyperparameters)
```

```
{'leaf_size': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], 'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9], 'p': [1, 2]}
```

```
print(best_model.best_estimator_.get_params())
```

```
{'algorithm': 'auto', 'leaf_size': 9, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 7, 'p': 1, 'weights':
```

```
print('Best leaf_size:', best_model.best_estimator_.get_params()['leaf_size'])
print('Best p:', best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
print('Best Score:', best_model.best_score_)
```

```
Best leaf_size: 9
Best p: 1
Best n_neighbors: 7
Best Score: 0.7483536683904332
```

```
x=df.drop(columns=['target'])
y=df['target']
knn=KNeighborsClassifier()
train_X,test_X,train_Y,test_Y=train_test_split(x,y,train_size=0.2, random_state=4)
```

```
knn.fit(train_X,train_Y)
```

```
KNeighborsClassifier()
```

```
y_pred=knn.predict(test_X)
```

```
print(classification_report(test_Y,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.53      0.66      0.59       106
               1       0.67      0.54      0.60       137

        accuracy                           0.59       243
       macro avg       0.60      0.60      0.59       243
    weighted avg       0.61      0.59      0.59       243
```

```
roc_auc_score(test_Y,y_pred)
```

```
0.600261671946013
```