random-forest-and-tuning

October 16, 2023

[1]:
```
# loading libraries
import pandas as pd
import seaborn as sns
import numpy as np
from matplotlib import pyplot as plt
```

[2]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

[2]:
```
# loading dataset
df = pd.read_csv('Heart Disease.csv')
```

[3]:
```
df.head()
```

[3]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth	SleepTime	Asthma	KidneyDisease	SkinCancer
0	No	16.60	Yes	No	No	3.0	30.0	No	Female	55-59	White	Yes	Yes	Very good	5.0	Yes	No	Yes
1	No	20.34	No	No	Yes	0.0	0.0	No	Female	80 or older	White	No	Yes	Very good	7.0	No	No	No
2	No	26.58	Yes	No	No	20.0	30.0	No	Male	65-69	White	Yes	Yes	Fair	8.0	Yes	No	No
3	No	24.21	No	No	No	0.0	0.0	No	Female	75-79	White	No	No	Good	6.0	No	No	Yes
4	No	23.71	No	No	No	28.0	0.0	Yes	Female	40-44	White	No	Yes	Very good	8.0	No	No	No

[4]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[5]:
```
from pandas.core.dtypes.common import is_numeric_dtype
for column in df.columns:
    if is_numeric_dtype(df[column]):
        continue
    df[column] = le.fit_transform(df[column])
```

[6]:
```
df.head()
```

[6]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	\
0	0	16.60	1	0	0	3.0	
1	0	20.34	0	0	1	0.0	
2	0	26.58	1	0	0	20.0	
3	0	24.21	0	0	0	0.0	
4	0	23.71	0	0	0	28.0	

	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	\
0	30.0	0	0	7	5	2	
1	0.0	0	0	12	5	0	
2	30.0	0	1	9	5	2	
3	0.0	0	0	11	5	0	
4	0.0	1	0	4	5	0	

	PhysicalActivity	GenHealth	SleepTime	Asthma	KidneyDisease	SkinCancer
0	1	4	5.0	1	0	1
1	1	4	7.0	0	0	0
2	1	1	8.0	1	0	0
3	0	2	6.0	0	0	1
4	1	4	8.0	0	0	0

[7]:
```
# separate values
x = df.drop('HeartDisease',axis=1)
y = df.HeartDisease
```

[8]:
```
x.head()
```

[8]:

	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	\
0	16.60	1	0	0	3.0	30.0	
1	20.34	0	0	1	0.0	0.0	
2	26.58	1	0	0	20.0	30.0	
3	24.21	0	0	0	0.0	0.0	
4	23.71	0	0	0	28.0	0.0	

	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth	\
0	0	0	7	5	2	1	4	
1	0	0	12	5	0	1	4	
2	0	1	9	5	2	1	1	
3	0	0	11	5	0	0	2	

```
4            1    0            4     5         0                  1          4
```

```
   SleepTime  Asthma  KidneyDisease  SkinCancer
0        5.0       1              0           1
1        7.0       0              0           0
2        8.0       1              0           0
3        6.0       0              0           1
4        8.0       0              0           0
```

[9]:
```
y.head()
```

[9]:
```
0    0
1    0
2    0
3    0
4    0
Name: HeartDisease, dtype: int32
```

[10]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=.3,random_state=42)
```

[11]:
```
xtrain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 223856 entries, 303145 to 121958
Data columns (total 17 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   BMI               223856 non-null  float64
 1   Smoking           223856 non-null  int32
 2   AlcoholDrinking   223856 non-null  int32
 3   Stroke            223856 non-null  int32
 4   PhysicalHealth    223856 non-null  float64
 5   MentalHealth      223856 non-null  float64
 6   DiffWalking       223856 non-null  int32
 7   Sex               223856 non-null  int32
 8   AgeCategory       223856 non-null  int32
 9   Race              223856 non-null  int32
 10  Diabetic          223856 non-null  int32
 11  PhysicalActivity  223856 non-null  int32
 12  GenHealth         223856 non-null  int32
 13  SleepTime         223856 non-null  float64
 14  Asthma            223856 non-null  int32
 15  KidneyDisease     223856 non-null  int32
 16  SkinCancer        223856 non-null  int32
dtypes: float64(4), int32(13)
memory usage: 19.6 MB
```

[12]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
```

[14]:
```
dtc.fit(xtrain,ytrain)
```

[14]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

[15]:
```
pred = dtc.predict(xtest)
pred
```

[15]:
```
array([0, 0, 0, …, 0, 0, 0])
```

[16]:
```
ytest.head()
```

[16]:
```
271884    0
270361    0
219060    0
24010     0
181930    0
Name: HeartDisease, dtype: int32
```

[17]:
```
# testing accuracy
accu_dt = dtc.score(xtest,ytest)
accu_dt
```

[17]:
```
0.8638405653592387
```

0.0.1 Random Forest

[13]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
```

[19]:
```
rfc.fit(xtrain,ytrain)
```

[19]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[20]:
```
# testing accuracy
accu_rf = rfc.score(xtest,ytest)
accu_rf
```

[20]:
```
0.9047519778192393
```

0.1 Tuning

0.1.1 Randomized Search CV

[21]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[15]:
```
#Assign Parameters
n_estimators = [int(x) for x in np.linspace(start=15,stop=300,num=40)]
criterion = ['gini', 'entropy']
max_features = ['sqrt', 'log2']
max_depth = np.random.randint(1,10,20)
min_samples_split = np.random.randint(2,10,15)
min_samples_leaf = np.random.randint(1,10,15)
parameters = {
    'n_estimators'  : n_estimators,
    'criterion' : criterion,
    'max_features': max_features,
    'max_depth' : max_depth,
    'min_samples_split' : min_samples_split,
    'min_samples_leaf': min_samples_leaf
}
```

[23]:
```
parameters
```

[23]:
```
{'n_estimators': [15,
  22,
  29,
  36,
  44,
  51,
  58,
  66,
  73,
  80,
  88,
  95,
  102,
  110,
  117,
  124,
  131,
```

```
 139,
 146,
 153,
 161,
 168,
 175,
 183,
 190,
 197,
 205,
 212,
 219,
 226,
 234,
 241,
 248,
 256,
 263,
 270,
 278,
 285,
 292,
 300],
 'criterion': ['gini', 'entropy'],
 'max_features': ['sqrt', 'log2'],
 'max_depth': array([4, 2, 4, 7, 5, 2, 3, 7, 2, 6, 1, 7, 9, 1, 5, 8, 4, 7, 8,
5]),
 'min_samples_split': array([2, 5, 5, 6, 3, 6, 5, 4, 6, 9, 4, 9, 9, 7, 8]),
 'min_samples_leaf': array([1, 6, 5, 1, 5, 2, 8, 3, 2, 2, 5, 5, 4, 5, 5])}
```

[24]:
```
random_forest = RandomForestClassifier()
```

[25]:
```
rs = RandomizedSearchCV(random_forest, parameters, n_iter = 300,cv = 3)
 ↪ #cv = cross validation
```

[26]:
```
rs.fit(xtrain,ytrain)
```

[26]:
```
RandomizedSearchCV(cv=3, error_score=nan,
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    max_samples=None,
                                                    min_impurity_decrease=0.0,
```

```
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
      min_weight_fraction_leaf=0.0,
                                                    n_estimators=100,
                                                    n_jobs…
                                             'max_features': ['sqrt', 'log2'],
                                             'min_samples_leaf': array([1, 6, 5, 1,
      5, 2, 8, 3, 2, 2, 5, 5, 4, 5, 5]),
                                             'min_samples_split': array([2, 5, 5, 6,
      3, 6, 5, 4, 6, 9, 4, 9, 9, 7, 8]),
                                             'n_estimators': [15, 22, 29, 36, 44, 51,
                                                              58, 66, 73, 80, 88, 95,
                                                              102, 110, 117, 124,
                                                              131, 139, 146, 153,
                                                              161, 168, 175, 183,
                                                              190, 197, 205, 212,
                                                              219, 226, …]},
                          pre_dispatch='2*n_jobs', random_state=None, refit=True,
                          return_train_score=False, scoring=None, verbose=0)
```

[27]:
```
pd.DataFrame(rs.cv_results_)
```

[27]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	\
0	5.430365	0.052867	0.691399	0.001661	
1	10.061610	0.056102	0.853971	0.009705	
2	1.378290	0.014977	0.190531	0.004594	
3	7.822788	0.043505	0.629860	0.006899	
4	17.967162	0.690604	1.392113	0.044654	
..	…	…	…	…	
295	11.151847	0.080100	0.829919	0.001306	
296	8.531872	0.025751	0.730784	0.013566	
297	19.071294	0.057096	1.487293	0.000017	
298	2.274742	0.030877	0.261399	0.008162	
299	25.763117	0.168773	1.967093	0.081768	

	param_n_estimators	param_min_samples_split	param_min_samples_leaf	\
0	146	3	8	
1	131	8	2	
2	36	9	5	
3	88	4	5	
4	183	8	8	
..	…	…	…	
295	95	9	5	
296	110	3	5	
297	205	9	2	
298	44	5	4	

```
299               183                      5                      3

    param_max_features param_max_depth param_criterion  \
0                 sqrt               1            gini
1                 log2               4         entropy
2                 sqrt               1            gini
3                 sqrt               5         entropy
4                 log2               5         entropy
..                 …               …              …
295               log2               7            gini
296               sqrt               4            gini
297               log2               5         entropy
298               sqrt               2         entropy
299               log2               9            gini

                                                 params  split0_test_score  \
0    {'n_estimators': 146, 'min_samples_split': 3, …           0.914754
1    {'n_estimators': 131, 'min_samples_split': 8, …           0.914754
2    {'n_estimators': 36, 'min_samples_split': 9, '…           0.914754
3    {'n_estimators': 88, 'min_samples_split': 4, '…           0.915022
4    {'n_estimators': 183, 'min_samples_split': 8, …           0.915035
..                                                 …                …
295  {'n_estimators': 95, 'min_samples_split': 9, '…           0.915853
296  {'n_estimators': 110, 'min_samples_split': 3, …           0.914847
297  {'n_estimators': 205, 'min_samples_split': 9, …           0.915048
298  {'n_estimators': 44, 'min_samples_split': 5, '…           0.914754
299  {'n_estimators': 183, 'min_samples_split': 5, …           0.916000

     split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0             0.914754           0.914752         0.914753    5.385501e-07
1             0.914794           0.914793         0.914780    1.868913e-05
2             0.914754           0.914752         0.914753    5.385501e-07
3             0.915115           0.914994         0.915044    5.207322e-05
4             0.915129           0.914900         0.915021    9.398096e-05
..                 …                  …                …               …
295           0.915692           0.915490         0.915678    1.485021e-04
296           0.914914           0.914779         0.914847    5.517795e-05
297           0.915062           0.914967         0.915026    4.195908e-05
298           0.914754           0.914752         0.914753    5.385501e-07
299           0.915866           0.915610         0.915825    1.616904e-04

     rank_test_score
0                222
1                208
2                222
3                154
4                165
```

```
..                ...
295                58
296               181
297               161
298               222
299                18

[300 rows x 17 columns]
```

[28]:
```
# Best parameters
rs.best_params_
```

[28]:
```
{'n_estimators': 248,
 'min_samples_split': 7,
 'min_samples_leaf': 1,
 'max_features': 'log2',
 'max_depth': 9,
 'criterion': 'entropy'}
```

[29]:
```
# Best accuracy score
rs.best_score_
```

[29]:
```
0.9159638329719062
```

[30]:
```
nrs = rs.best_estimator_
```

[31]:
```
nrs.fit(xtrain,ytrain)
```

[31]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=9, max_features='log2',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=7,
                       min_weight_fraction_leaf=0.0, n_estimators=248,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[32]:
```
# testing accuracy
accu_rscv = nrs.score(xtest,ytest)
accu_rscv
```

[32]:
```
0.9148729922138025
```

Grid Search CV

[14]:
```
from sklearn.model_selection import GridSearchCV
```

[16]:
```
parameters
```

```
[16]: {'n_estimators': [15,
        22,
        29,
        36,
        44,
        51,
        58,
        66,
        73,
        80,
        88,
        95,
        102,
        110,
        117,
        124,
        131,
        139,
        146,
        153,
        161,
        168,
        175,
        183,
        190,
        197,
        205,
        212,
        219,
        226,
        234,
        241,
        248,
        256,
        263,
        270,
        278,
        285,
        292,
        300],
       'criterion': ['gini', 'entropy'],
       'max_features': ['sqrt', 'log2'],
       'max_depth': array([4, 2, 5, 7, 4, 9, 3, 4, 8, 2, 5, 3, 9, 8, 7, 6, 2, 9, 7,
      5]),
       'min_samples_split': array([6, 6, 5, 8, 2, 3, 3, 2, 2, 3, 5, 6, 7, 6, 5]),
       'min_samples_leaf': array([3, 4, 6, 3, 8, 4, 2, 8, 1, 4, 6, 9, 4, 8, 7])}
```

[17]:
```
clf = RandomForestClassifier()
```

[18]:
```
gs = GridSearchCV(clf,parameters,cv=3,n_jobs=-1)
```

[ ]:
```
gs.fit(xtrain,ytrain)
```

[ ]:
```
gs.best_score_
```

[ ]:
```
ngs = gs.best_estimator_
ngs
```

[ ]:
```
ngs.fit(xtrain,ytrain)
```

[ ]:
```
accu_gscv = ngs.score(xtest,ytest)
```

Comparison

[ ]:
```
df_2 = pd.DataFrame([('Decision Tree',accu_dt),
                     ('Random Forest',accu_rf),
                     ('Randomized Search CV',accu_rscv),
                     ('Grid Search␣
 ↪CV',accu_gscv)],columns=['Methods','Accuracy'])
```

[ ]:
```
sns.barplot(x="Accuracy", y="Methods", data=df_2,
                palette="Blues_d")
```