14th April Assignment

April 29, 2023

1 Assignment 69

Build a random forest classifier to predict the risk of heart disease based on a dataset of patient information. The dataset contains 303 instances with 14 features, including age, sex, chest pain type, resting blood pressure, serum cholesterol, and maximum heart rate achieved.

Dataset link: https://drive.google.com/file/d/1bGoIE4Z2kG5nyh-fGZAJ7LH0ki3UfmSJ/view? usp=share_link

Q1. Preprocess the dataset by handling missing values, encoding categorical variables, and scaling the numerical features if necessary.

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.simplefilter('ignore')
```

[2]:
```
# read the data
df = pd.read_csv('dataset.csv')
df.head()
```

[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

[3]:
```
# information of data
df.info()
```

```
<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
```

[4]:
```
df.describe()
```

[4]:

	age	sex	cp	trestbps	chol	fbs	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	
std	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	
75%	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	

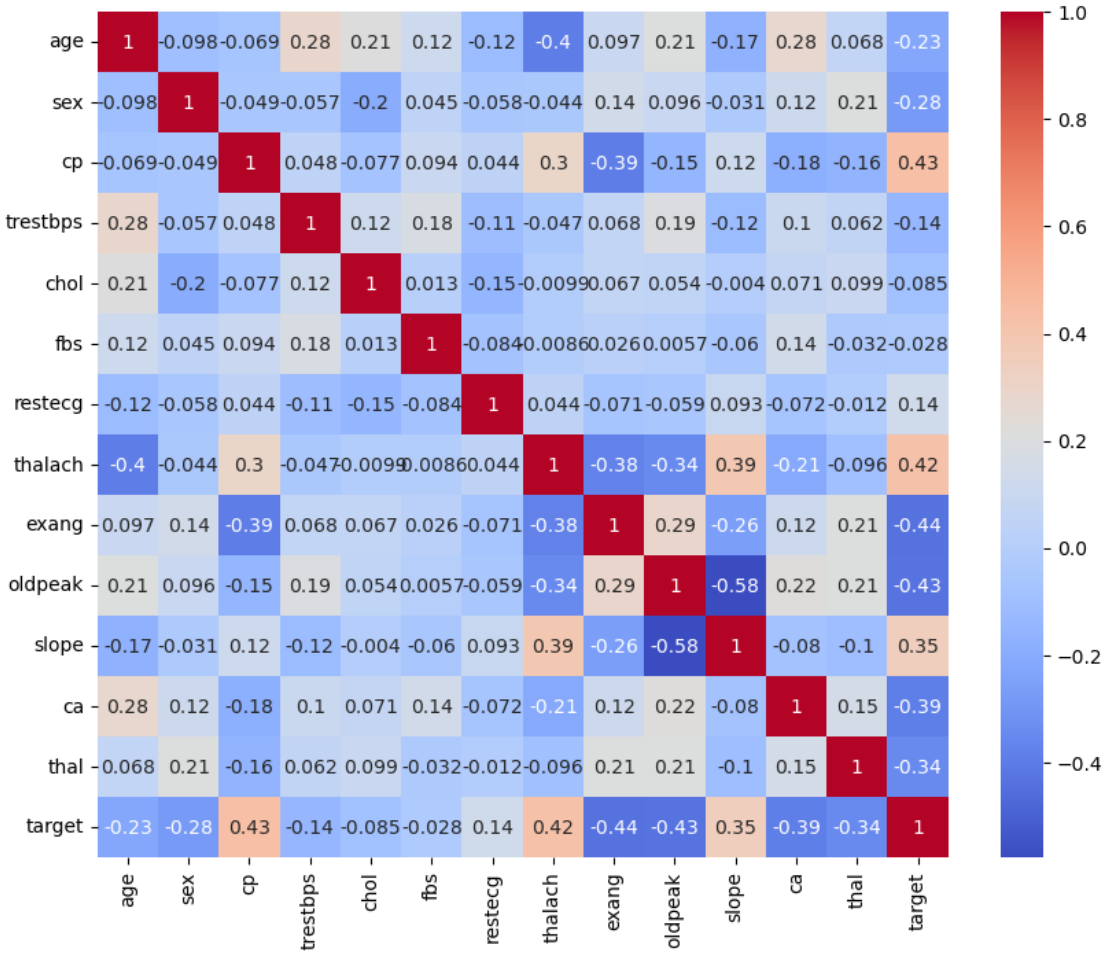
```
              thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[6]:
```
df.isnull().sum()
```

[6]:
```
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
```

There is no missing value

[7]:
```
# heatmap corr
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),cmap='coolwarm',annot=True)
plt.show()
```



Q2. Split the dataset into a training set (70%) and a test set (30%).

[8]:
```
#segregate independent and dependent feature
X=df.iloc[:,:-1]
y=df['target']
```

[9]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪30,random_state=42)
```

Q3. Train a random forest classifier on the training set using 100 trees and a maximum depth of 10 for each tree. Use the default values for other hyperparameters.

[10]:
```
from sklearn.ensemble import RandomForestClassifier
```

[11]:
```
classifier=RandomForestClassifier(n_estimators=100,max_depth=10)
```

[12]:
```
classifier.fit(X_train,y_train)
```

[12]:
```
RandomForestClassifier(max_depth=10)
```

[13]:
```
y_pred=classifier.predict(X_test)
```

Q4. Evaluate the performance of the model on the test set using accuracy, precision, recall, and F1 score.

[14]:
```
from sklearn.metrics import accuracy_score,classification_report
```

[15]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.8131868131868132
```

[16]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.78      0.79        41
           1       0.82      0.84      0.83        50

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.81      0.81      0.81        91
```

Q6. Tune the hyperparameters of the random forest classifier using grid search or random search. Try different values of the number of trees, maximum depth, minimum samples split, and minimum samples leaf. Use 5-fold cross-validation to evaluate the performance of each set of hyperparameters.

[17]:
```
# let's build model with hyperparameter tunning
parameters = {
    'max_depth':[3,5,10,None],
    'n_estimators':[100,200,300],
    'criterion':['gini','entropy']
}
```

[18]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[19]:
```
cv =␣
 ↪RandomizedSearchCV(RandomForestClassifier(),param_distributions=parameters,cv=5,scoring='ac
```

[20]:
```
cv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END criterion=gini, max_depth=3, n_estimators=100;, score=0.860 total
```

```
time=   0.2s
[CV 2/5] END criterion=gini, max_depth=3, n_estimators=100;, score=0.860 total
time=   0.2s
[CV 3/5] END criterion=gini, max_depth=3, n_estimators=100;, score=0.738 total
time=   0.2s
[CV 4/5] END criterion=gini, max_depth=3, n_estimators=100;, score=0.881 total
time=   0.2s
[CV 5/5] END criterion=gini, max_depth=3, n_estimators=100;, score=0.738 total
time=   0.2s
[CV 1/5] END criterion=entropy, max_depth=10, n_estimators=200;, score=0.837
total time=   0.4s
[CV 2/5] END criterion=entropy, max_depth=10, n_estimators=200;, score=0.860
total time=   0.4s
[CV 3/5] END criterion=entropy, max_depth=10, n_estimators=200;, score=0.714
total time=   0.4s
[CV 4/5] END criterion=entropy, max_depth=10, n_estimators=200;, score=0.857
total time=   0.4s
[CV 5/5] END criterion=entropy, max_depth=10, n_estimators=200;, score=0.762
total time=   0.4s
[CV 1/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.884
total time=   0.4s
[CV 2/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.837
total time=   0.4s
[CV 3/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.738
total time=   0.4s
[CV 4/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.881
total time=   0.4s
[CV 5/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.738
total time=   0.4s
[CV 1/5] END criterion=gini, max_depth=5, n_estimators=300;, score=0.860 total
time=   0.5s
[CV 2/5] END criterion=gini, max_depth=5, n_estimators=300;, score=0.860 total
time=   0.5s
[CV 3/5] END criterion=gini, max_depth=5, n_estimators=300;, score=0.690 total
time=   0.5s
[CV 4/5] END criterion=gini, max_depth=5, n_estimators=300;, score=0.881 total
time=   0.5s
[CV 5/5] END criterion=gini, max_depth=5, n_estimators=300;, score=0.738 total
time=   0.5s
[CV 1/5] END criterion=entropy, max_depth=3, n_estimators=300;, score=0.860
total time=   0.5s
[CV 2/5] END criterion=entropy, max_depth=3, n_estimators=300;, score=0.860
total time=   0.5s
[CV 3/5] END criterion=entropy, max_depth=3, n_estimators=300;, score=0.762
total time=   0.5s
[CV 4/5] END criterion=entropy, max_depth=3, n_estimators=300;, score=0.905
total time=   0.5s
[CV 5/5] END criterion=entropy, max_depth=3, n_estimators=300;, score=0.762
```

```
total time=   0.5s
[CV 1/5] END criterion=entropy, max_depth=None, n_estimators=100;, score=0.860
total time=   0.2s
[CV 2/5] END criterion=entropy, max_depth=None, n_estimators=100;, score=0.814
total time=   0.2s
[CV 3/5] END criterion=entropy, max_depth=None, n_estimators=100;, score=0.714
total time=   0.2s
[CV 4/5] END criterion=entropy, max_depth=None, n_estimators=100;, score=0.905
total time=   0.2s
[CV 5/5] END criterion=entropy, max_depth=None, n_estimators=100;, score=0.786
total time=   0.2s
[CV 1/5] END criterion=gini, max_depth=5, n_estimators=200;, score=0.860 total
time=   0.4s
[CV 2/5] END criterion=gini, max_depth=5, n_estimators=200;, score=0.860 total
time=   0.4s
[CV 3/5] END criterion=gini, max_depth=5, n_estimators=200;, score=0.714 total
time=   0.4s
[CV 4/5] END criterion=gini, max_depth=5, n_estimators=200;, score=0.905 total
time=   0.4s
[CV 5/5] END criterion=gini, max_depth=5, n_estimators=200;, score=0.762 total
time=   0.4s
[CV 1/5] END criterion=entropy, max_depth=5, n_estimators=300;, score=0.884
total time=   0.5s
[CV 2/5] END criterion=entropy, max_depth=5, n_estimators=300;, score=0.837
total time=   0.5s
[CV 3/5] END criterion=entropy, max_depth=5, n_estimators=300;, score=0.690
total time=   0.5s
[CV 4/5] END criterion=entropy, max_depth=5, n_estimators=300;, score=0.905
total time=   0.5s
[CV 5/5] END criterion=entropy, max_depth=5, n_estimators=300;, score=0.738
total time=   0.5s
[CV 1/5] END criterion=entropy, max_depth=10, n_estimators=300;, score=0.860
total time=   0.5s
[CV 2/5] END criterion=entropy, max_depth=10, n_estimators=300;, score=0.860
total time=   0.5s
[CV 3/5] END criterion=entropy, max_depth=10, n_estimators=300;, score=0.690
total time=   0.5s
[CV 4/5] END criterion=entropy, max_depth=10, n_estimators=300;, score=0.857
total time=   0.5s
[CV 5/5] END criterion=entropy, max_depth=10, n_estimators=300;, score=0.786
total time=   0.5s
[CV 1/5] END criterion=entropy, max_depth=5, n_estimators=100;, score=0.884
total time=   0.2s
[CV 2/5] END criterion=entropy, max_depth=5, n_estimators=100;, score=0.814
total time=   0.2s
[CV 3/5] END criterion=entropy, max_depth=5, n_estimators=100;, score=0.690
total time=   0.2s
[CV 4/5] END criterion=entropy, max_depth=5, n_estimators=100;, score=0.905
```

```
total time=   0.2s
[CV 5/5] END criterion=entropy, max_depth=5, n_estimators=100;, score=0.762
total time=   0.2s
```

[20]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(),
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [3, 5, 10, None],
                                        'n_estimators': [100, 200, 300]},
                   scoring='accuracy', verbose=3)
```

Q7. Report the best set of hyperparameters found by the search and the corresponding performance metrics. Compare the performance of the tuned model with the default model.

[21]:
```
cv.best_params_
```

[21]:
```
{'n_estimators': 300, 'max_depth': 3, 'criterion': 'entropy'}
```

[22]:
```
cv.best_score_
```

[22]:
```
0.8299003322259135
```

Q8. Interpret the model by analysing the decision boundaries of the random forest classifier. Plot the decision boundaries on a scatter plot of two of the most important features. Discuss the insights and limitations of the model for predicting heart disease risk.