In [1]:

```
import csv
from ast import literal_eval
import numpy as np
#import in library
import pandas as pd
df = pd.read_csv('covid_hospitalization_sample.csv',encoding="ISO-8859-1")
df.head()
from sklearn.model_selection import train_test_split
```

Task 1: Preparation of data (2 marks)

1. What is the proportion of patients who are classified as COVID-19 positive? Would you require a sampling technique on this data?

The perportion of tested positive is 558/(672+558)=45.37%

In [2]:

```
tested_cases=df[['SARS-Cov-2 exam result','Patient ID']].copy()
tested_cases.groupby('SARS-Cov-2 exam result').count()
```

Out[2]:

SARS-Cov-2 exam result	Patient ID
negative	672
positive	558

Sampling the data

3.What variables did you include in the analysis? justify the choice of input selection(input/target).

The document states the hospital only interested in the impact of "regular blood test" and "influenza rapid test" group variables.

In [3]:

```
#Create a subset of data that contain only the useful columns
sample_fixed=df[['Patient ID','Patient age quantile','Patient addmited to regular ward','Patient addmited to semi-intensive unit','P
atient addmited to intensive care unit', 'Proteina C reativa', 'Neutrophils', 'Mean platelet volume',
'Monocytes', 'Red blood cell distribution width', 'Red blood Cells', 'Platelets',
'Eosinophils', 'Basophils', 'Leukocytes', 'Mean corpuscular hemoglobin', 'Mean corpuscular volume', 'Mean corpuscular hemoglobin con
centration',
'Lymphocytes', 'Hemoglobin', 'Hematocrit','Influenza B rapid test', 'Influenza A rapid test']]
pd.options.display.max_columns=None
sample_fixed
```

Out[3]:

	Patient ID	Patient age quantile	Patient addmited to regular ward	Patient addmited to semi-intensive unit	Patient addmited to intensive care unit	Proteina C reativa	Neutrophils	Mean platelet volume	Monocytes	Red blood cell distribution width	Red blood Cells	Platelets	E
0	9abc76405794c6d	9	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	01d324f278f3101	16	0	0	0	-0.316791	-0.356851	0.010677	1.250496	-0.182790	0.525133	0.135801	
2	b2fb9312efbadc1	9	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	eb5ef46a892698f	0	0	0	0	-0.342622	NaN	-0.438097	-1.270772	0.613318	0.472242	1.065375	
4	a713345aef928fa	10	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1225	0716648a7fa58a6	3	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1226	0db34a7c845e57a	13	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1227	0ecac021e9c4511	17	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1228	25ab118504a09df	2	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1229	713f1426f544d25	17	0	0	0	-0.511518	-0.143785	-0.662483	-0.351560	-0.359703	1.794518	0.123239	

1230 rows × 23 columns

2. Did you have to fix any data quality problems? Detail them.

convert the value into binary 0/1 variable,also covert dtype, one hot encoding and imputation as following

In [4]:

```
#create the target dataframe
target_map = {'negative':0, 'positive': 1}
tested_cases['SARS-Cov-2 exam result'] = tested_cases['SARS-Cov-2 exam result'].map(target_map)
target=tested_cases['SARS-Cov-2 exam result'].copy()
target.head()
```

Out[4]:

```
0    0
1    0
2    0
3    0
4    0
Name: SARS-Cov-2 exam result, dtype: int64
```

In [5]:

```
#convert the dtype
sample_fixed['Patient ID']=sample_fixed['Patient ID'].astype(str)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [6]:

```
# One hot encoding
demo_sample_fixed = sample_fixed[['Influenza B rapid test','Influenza A rapid test']]
demo_sample_fixed=pd.get_dummies(demo_sample_fixed)
#merge the two data frame
sample_fixed=pd.concat([sample_fixed,demo_sample_fixed],axis=1)

sample_fixed
```

Out[6]:

	Patient ID	Patient age quantile	Patient addmited to regular ward	Patient addmited to semi-intensive unit	Patient addmited to intensive care unit	Proteina C reativa	Neutrophils	Mean platelet volume	Monocytes	Red blood cell distribution width	Red blood Cells	Platelets	E
0	9abc76405794c6d	9	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	01d324f278f3101	16	0	0	0	-0.316791	-0.356851	0.010677	1.250496	-0.182790	0.525133	0.135801	
2	b2fb9312efbadc1	9	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	eb5ef46a892698f	0	0	0	0	-0.342622	NaN	-0.438097	-1.270772	0.613318	0.472242	1.065375	
4	a713345aef928fa	10	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1225	0716648a7fa58a6	3	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1226	0db34a7c845e57a	13	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1227	0ecac021e9c4511	17	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1228	25ab118504a09df	2	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1229	713f1426f544d25	17	0	0	0	-0.511518	-0.143785	-0.662483	-0.351560	-0.359703	1.794518	0.123239	

1230 rows × 27 columns

In [7]:

```
#drop the unused columns
sample_fixed.drop(['Influenza B rapid test','Influenza A rapid test','Patient ID','Patient age quantile','Patient addmited to regular ward','Patient addmited to semi-intensive unit','Patient addmited to intensive care unit'],axis=1,inplace=True)
sample_fixed
```

Out[7]:

	Proteina C reativa	Neutrophils	Mean platelet volume	Monocytes	Red blood cell distribution width	Red blood Cells	Platelets	Eosinophils	Basophils	Leukocytes	Mean corpuscular hemoglobin	Mean corpuscular volume
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	-0.316791	-0.356851	0.010677	1.250496	-0.182790	0.525133	0.135801	-0.624811	0.081693	-0.653951	-0.501356	0.086074
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	-0.342622	NaN	-0.438097	-1.270772	0.613318	0.472242	1.065375	-0.835508	-1.140144	-0.080696	-1.651331	-1.255906
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	..
1225	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1226	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1227	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1228	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1229	-0.511518	-0.143785	-0.662483	-0.351560	-0.359703	1.794518	0.123239	0.175837	0.692611	-0.709607	-0.710443	-0.815256

1230 rows × 20 columns

In [8]:

```
#impute missing data with its mean value using fillna() and mean()
sample_fixed['Proteina C reativa'].fillna(sample_fixed['Proteina C reativa'].mean(), inplace=True)
sample_fixed['Neutrophils'].fillna(sample_fixed['Neutrophils'].mean(), inplace=True)
sample_fixed['Mean platelet volume'].fillna(sample_fixed['Mean platelet volume'].mean(), inplace=True)
sample_fixed['Monocytes'].fillna(sample_fixed['Monocytes'].mean(), inplace=True)
sample_fixed['Red blood cell distribution width'].fillna(sample_fixed['Red blood cell distribution width'].mean(), inplace=True)
sample_fixed['Red blood Cells'].fillna(sample_fixed['Red blood Cells'].mean(), inplace=True)
sample_fixed['Platelets'].fillna(sample_fixed['Platelets'].mean(), inplace=True)
sample_fixed['Eosinophils'].fillna(sample_fixed['Eosinophils'].mean(), inplace=True)
sample_fixed['Basophils'].fillna(sample_fixed['Basophils'].mean(), inplace=True)
sample_fixed['Leukocytes'].fillna(sample_fixed['Leukocytes'].mean(), inplace=True)
sample_fixed['Mean corpuscular hemoglobin'].fillna(sample_fixed['Mean corpuscular hemoglobin'].mean(), inplace=True)
sample_fixed['Mean corpuscular volume'].fillna(sample_fixed['Mean corpuscular volume'].mean(), inplace=True)
sample_fixed['Mean corpuscular hemoglobin concentration'].fillna(sample_fixed['Mean corpuscular hemoglobin concentration'].mean(), i
nplace=True)
sample_fixed['Lymphocytes'].fillna(sample_fixed['Lymphocytes'].mean(), inplace=True)
sample_fixed['Hemoglobin'].fillna(sample_fixed['Hemoglobin'].mean(), inplace=True)
sample_fixed['Hematocrit'].fillna(sample_fixed['Hemoglobin'].mean(), inplace=True)

sample_fixed['Proteina C reativa']
```

Out[8]:

```
0       0.119012
1      -0.316791
2       0.119012
3      -0.342622
4       0.119012
          ...
1225    0.119012
1226    0.119012
1227    0.119012
1228    0.119012
1229   -0.511518
Name: Proteina C reativa, Length: 1230, dtype: float64
```

In [9]:

```
# setting random state
rs=10
#split the train and test data to 70% and 30%
sample_fixed_mat=sample_fixed.to_numpy()
sample_fixed_train,sample_fixed_test,target_train,target_test=train_test_split(sample_fixed_mat,target,test_size=0.3,stratify=target
,random_state=rs)
target_train
```

Out[9]:

```
24      0
1119    0
1109    0
434     1
257     1
       ..
465     1
464     1
737     0
241     1
45      0
Name: SARS-Cov-2 exam result, Length: 861, dtype: int64
```

Task 2: Predictive modeling using Decision Tree (4 marks)

1. Build a decision tree using the default setting. Answer the followings:

In [10]:

```
#building the decision tree by defalut
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

model=DecisionTreeClassifier(random_state=rs)
model.fit(sample_fixed_train,target_train)
```

Out[10]:

```
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=10, splitter='best')
```

a. What is the classification accuracy of training and test datasets?

In [11]:

```
print("Train accuracy: ", model.score(sample_fixed_train,target_train))
```

```
Train accuracy:  0.6353077816492451
```

In [12]:

```
print("Test accuracy: ", model.score(sample_fixed_test,target_test))
```

```
Test accuracy:  0.5880758807588076
```

In [13]:

```
#classification_report
target_pred=model.predict(sample_fixed_test)
print(classification_report(target_test,target_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.34      0.47       202
           1       0.53      0.89      0.66       167

    accuracy                           0.59       369
   macro avg       0.66      0.61      0.57       369
weighted avg       0.67      0.59      0.56       369
```

In [14]:

```
#feature importances
importances=model.feature_importances_
feature_names=sample_fixed.columns

#sort them out in descending order
indices=np.argsort(importances)
indices=np.flip(indices,axis=0)

#limit to 20 features
indices=indices[:20]

for i in indices:
    print(feature_names[i],':', importances[i])
```

```
Influenza A rapid test_positive : 0.18511076230510762
Influenza B rapid test_positive : 0.16624385997424365
Eosinophils : 0.14706316837111683
Leukocytes : 0.11963781746353686
Neutrophils : 0.09816332291709781
Mean corpuscular volume : 0.09487771473943589
Basophils : 0.05431427537007332
Monocytes : 0.036623865302481746
Influenza B rapid test_negative : 0.03339059444788342
Red blood Cells : 0.021864963927081288
Proteina C reativa : 0.012752674204376436
Platelets : 0.010717598142159973
Mean platelet volume : 0.009624317449780618
Hematocrit : 0.009615065385624606
Hemoglobin : 0.0
Lymphocytes : 0.0
Mean corpuscular hemoglobin concentration : 0.0
Mean corpuscular hemoglobin : 0.0
Influenza A rapid test_negative : 0.0
Red blood cell distribution width : 0.0
```

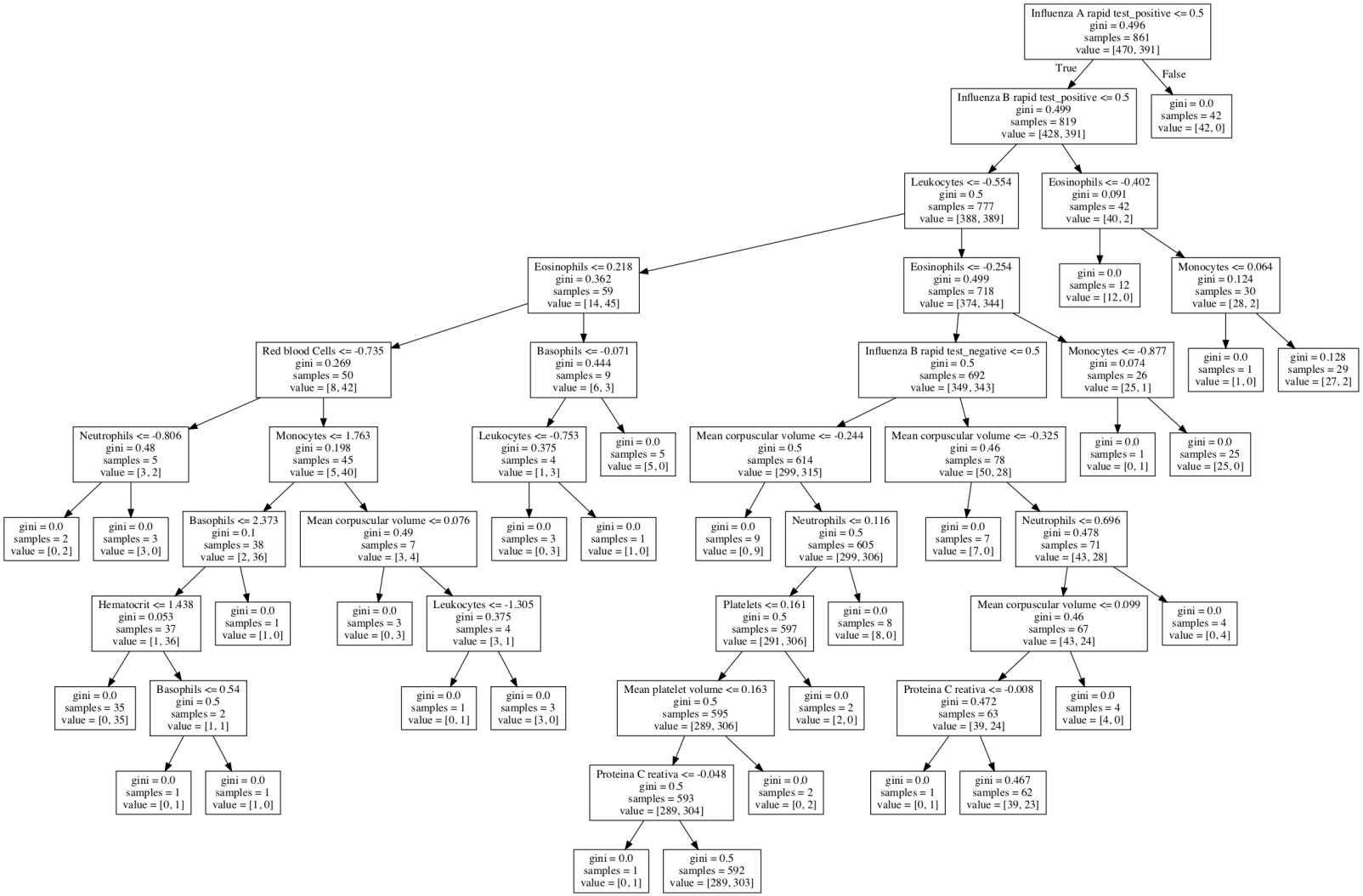
In [15]:

```
#Visualising decision tree
import pydot
from io import StringIO
from sklearn.tree import export_graphviz

dotfile=StringIO()
export_graphviz(model,out_file=dotfile,feature_names=sample_fixed.columns)
graph=pydot.graph_from_dot_data(dotfile.getvalue())
graph[0].write_png("dt_viz.png")
```

In [16]:

```
#display the image
from IPython.display import Image
Image(filename="dt_viz.png",width=1000,height=1000)
```

Out[16]:



In [17]:

```
#retrain with a small max_depth limit
model_small=DecisionTreeClassifier(max_depth=3,random_state=rs)
model_small.fit(sample_fixed_train,target_train)

print("Train accuracy: ",model_small.score(sample_fixed_train,target_train))
print("Test accuracy: ",model_small.score(sample_fixed_test,target_test))

target_pred=model_small.predict(sample_fixed_test)
print(classification_report(target_test,target_pred))
```

```
Train accuracy:  0.5818815331010453
Test accuracy:  0.5663956639566395
              precision    recall  f1-score   support

           0       0.56      0.95      0.71       202
           1       0.63      0.10      0.18       167

    accuracy                           0.57       369
   macro avg       0.60      0.53      0.44       369
weighted avg       0.59      0.57      0.47       369
```

In [18]:

```
importances = model_small.feature_importances_
feature_names = sample_fixed.columns
# sort them out in descending order
indices = np.argsort(importances)
indices = np.flip(indices, axis=0)
# limit to 20 features, you can leave this out to print out everything
indices = indices[:20]
for i in indices:
 print(feature_names[i], ':', importances[i])
```

```
Influenza A rapid test_positive : 0.41949224397557183
Influenza B rapid test_positive : 0.3767366575521471
Leukocytes : 0.20201609138018473
Eosinophils : 0.0017550070920963063
Influenza B rapid test_negative : 0.0
Neutrophils : 0.0
Mean platelet volume : 0.0
Monocytes : 0.0
Red blood cell distribution width : 0.0
Red blood Cells : 0.0
Platelets : 0.0
Basophils : 0.0
Hematocrit : 0.0
Influenza A rapid test_negative : 0.0
Mean corpuscular hemoglobin : 0.0
Mean corpuscular volume : 0.0
Mean corpuscular hemoglobin concentration : 0.0
Lymphocytes : 0.0
Hemoglobin : 0.0
Proteina C reativa : 0.0
```

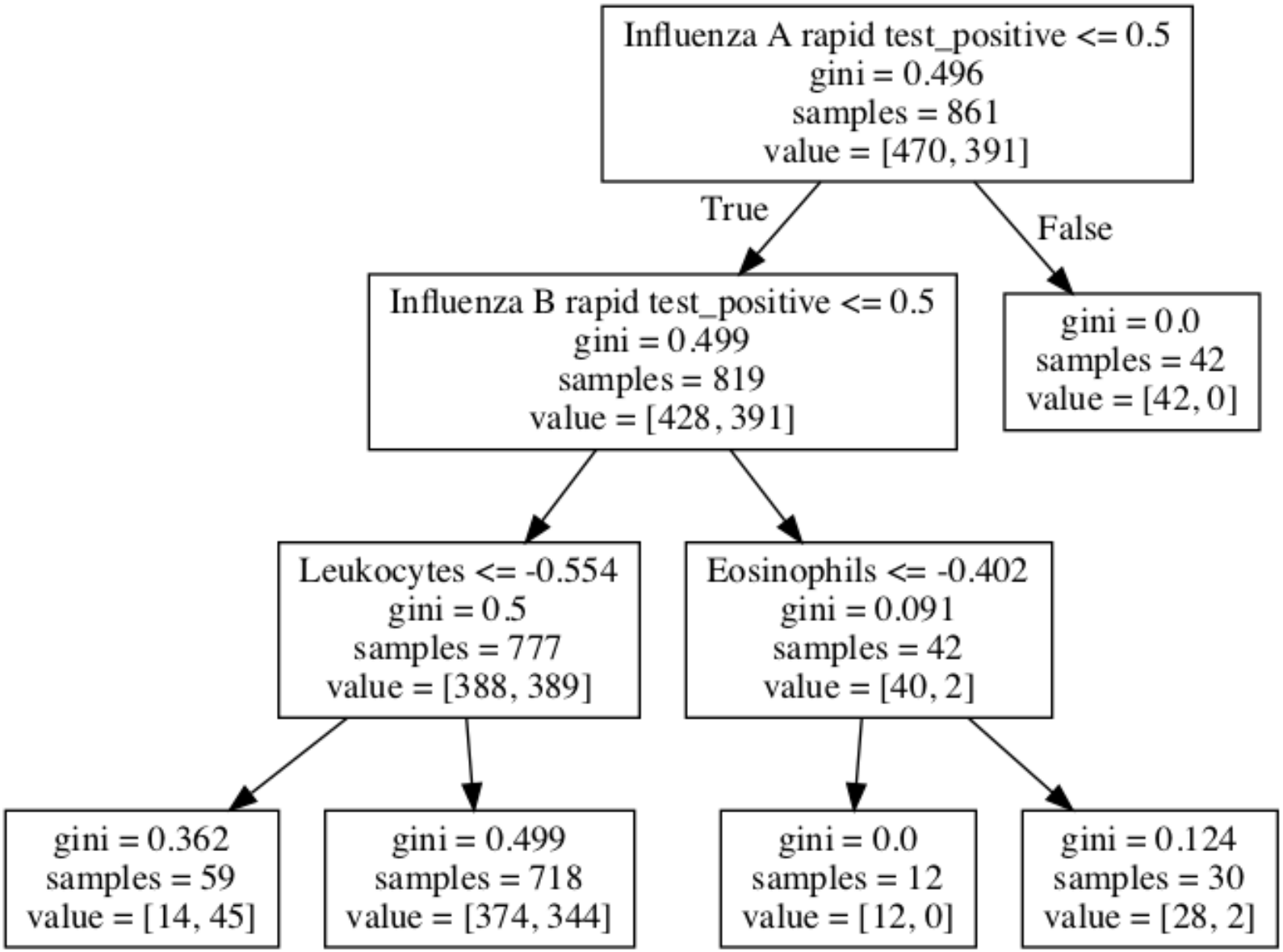
In [19]:

```
#visualize
dotfile=StringIO()
export_graphviz(model_small,out_file=dotfile,feature_names=sample_fixed.columns)
graph=pydot.graph_from_dot_data(dotfile.getvalue())
graph[0].write_png("dt_viz_small.png")
```

In [20]:

```
#display the image
from IPython.display import Image
Image(filename="dt_viz_small.png",width=1000,height=1000)
```

Out[20]:


Influenza A rapid test_positive <= 0.5
gini = 0.496
samples = 861
value = [470, 391]
True
False
Influenza B rapid test_positive <= 0.5
gini = 0.499
samples = 819
value = [428, 391]
gini = 0.0
samples = 42
value = [42, 0]
Leukocytes <= -0.554
gini = 0.5
samples = 777
value = [388, 389]
Eosinophils <= -0.402
gini = 0.091
samples = 42
value = [40, 2]
gini = 0.362
samples = 59
value = [14, 45]
gini = 0.499
samples = 718
value = [374, 344]
gini = 0.0
samples = 12
value = [12, 0]
gini = 0.124
samples = 30
value = [28, 2]

In [21]:

```
from sklearn.model_selection import GridSearchCV
# Grid search CV
params={'criterion':['gini','entropy'],
        'max_depth':range(1,16),
        'min_samples_leaf': range(0, 25, 5)[1:]}

cv_1=GridSearchCV(param_grid=params, estimator=DecisionTreeClassifier(random_state=rs), return_train_score=True,cv=10)
cv_1.fit(sample_fixed_train,target_train)
```

Out[21]:

```
GridSearchCV(cv=10, error_score=nan,
             estimator=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=10,
                                              splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(1, 16),
                         'min_samples_leaf': range(5, 25, 5)},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring=None, verbose=0)
```

In [22]:

```
result_set=cv_1.cv_results_
print(result_set)
```

```
{'mean_fit_time': array([0.00288048, 0.00137806, 0.00137529, 0.00137908, 0.00160768,
       0.00214579, 0.00194793, 0.00154951, 0.00190053, 0.00177329,
       0.00183871, 0.00178664, 0.00251987, 0.00213327, 0.00205157,
       0.00202727, 0.00224538, 0.00218761, 0.00239401, 0.00201383,
       0.00231888, 0.00223763, 0.00225012, 0.00210006, 0.00237715,
       0.00246701, 0.00222807, 0.00214117, 0.00261526, 0.00236974,
       0.00234694, 0.00218139, 0.00249879, 0.00311351, 0.00320373,
       0.00296023, 0.00299954, 0.00281878, 0.00272558, 0.00270267,
       0.0045552 , 0.0036119 , 0.00293086, 0.00279841, 0.00324583,
       0.00308468, 0.00278411, 0.00335491, 0.00325186, 0.00299821,
       0.00303185, 0.00248342, 0.00252934, 0.00239415, 0.0022579 ,
       0.00215898, 0.00255737, 0.00235965, 0.00220253, 0.00213203,
       0.00144684, 0.00139742, 0.00152543, 0.00141165, 0.00172107,
       0.00174737, 0.00169034, 0.00165191, 0.00203187, 0.00214796,
       0.00191414, 0.00200651, 0.0023206 , 0.0022974 , 0.00217907,
       0.00211358, 0.00260174, 0.00251379, 0.00241179, 0.00217628,
       0.00267646, 0.00249667, 0.00237997, 0.00232248, 0.00275581,
       0.0032294 , 0.0024873 , 0.00231326, 0.00288556, 0.00265155,
       0.00252514, 0.00280683, 0.00294247, 0.00260894, 0.00249491,
       0.0023392 , 0.00306044, 0.0026756 , 0.00293052, 0.0024435 ,
       0.00298617, 0.0028713 , 0.00251961, 0.00233405, 0.00295768,
       0.00262721, 0.00245061, 0.00231638, 0.00292351, 0.00264959,
       0.00244715, 0.00230792, 0.00294702, 0.00266664, 0.00299792,
       0.00361416, 0.0031065 , 0.00264628, 0.00255661, 0.0023068 ]), 'std_fit_time': array([1.51238185e-03, 6.86712560e-
05, 8.40324673e-05, 1.15005474e-04,
       1.17139998e-04, 3.69077189e-04, 5.86845699e-04, 3.57614279e-05,
       2.00447000e-04, 1.90490477e-05, 1.80449207e-04, 6.42288368e-05,
       8.53233086e-04, 2.54592446e-04, 1.40101559e-04, 1.30964651e-04,
       1.36332647e-04, 1.00814825e-04, 3.79523196e-04, 3.18934557e-05,
       6.28338991e-05, 5.69148695e-05, 1.13958846e-04, 5.02250170e-05,
       6.64687096e-05, 1.74036947e-04, 3.66770855e-05, 7.65937673e-05,
       1.72472989e-04, 6.32933929e-05, 8.16610468e-05, 9.57179898e-05,
       1.04413404e-04, 6.16897935e-04, 7.71982646e-04, 3.09960373e-04,
       2.26742111e-04, 2.99785322e-04, 1.81844414e-04, 1.40464718e-04,
       2.15675834e-03, 7.20043697e-04, 4.70868188e-04, 3.09415664e-04,
       1.47949439e-04, 3.67150592e-04, 1.64044499e-04, 5.32163709e-04,
       2.27536256e-04, 1.93324780e-04, 2.28786281e-04, 3.56375007e-04,
       8.81981705e-05, 1.28739048e-04, 1.19648223e-04, 6.21216708e-05,
       1.11285212e-04, 1.39176810e-04, 3.70814785e-05, 5.30039625e-05,
       6.00033064e-05, 1.86971857e-05, 1.55904366e-04, 8.78544125e-05,
       3.74899930e-05, 9.93442471e-05, 8.80758892e-05, 4.37817535e-05,
       7.53808554e-05, 2.91242436e-04, 3.60722268e-05, 3.52051888e-04,
       1.04794231e-04, 1.54334444e-04, 6.58011998e-05, 1.03956242e-04,
       2.71108308e-04, 1.29942115e-04, 1.81825136e-04, 4.81431996e-05,
       1.06783178e-04, 3.67192168e-05, 7.13076490e-05, 1.58158116e-04,
       1.14106397e-04, 7.12599331e-04, 1.20479174e-04, 5.55202341e-05,
       8.85194245e-05, 8.30871611e-05, 8.34610654e-05, 4.65516889e-04,
       1.20828466e-04, 6.50873158e-05, 8.56025764e-05, 7.01831609e-05,
       1.39655998e-04, 5.90660605e-05, 3.60536371e-04, 2.51626336e-04,
       1.01236776e-04, 1.88553509e-04, 1.46252572e-04, 1.00300293e-04,
       1.01436288e-04, 8.24281027e-05, 1.05716809e-04, 6.39787894e-05,
       1.19846569e-04, 9.07767461e-05, 4.51901584e-05, 3.81109619e-05,
```

```
        1.25774740e-04, 1.26356091e-04, 5.65750407e-04, 7.41938327e-04,
        5.42894320e-04, 1.09826325e-04, 1.83011996e-04, 6.30187887e-05]), 'mean_score_time': array([0.000948  , 0.0004580
7, 0.00045724, 0.00046208, 0.00046215,
        0.00054562, 0.00056865, 0.00044982, 0.00047629, 0.00044999,
        0.0004638 , 0.00045626, 0.00053837, 0.00054471, 0.00047119,
        0.00046713, 0.00046599, 0.00050848, 0.00056679, 0.00045936,
        0.00047147, 0.00046012, 0.00048356, 0.00048089, 0.00045681,
        0.00049911, 0.00048347, 0.00047252, 0.00049384, 0.00048132,
        0.00049188, 0.00047333, 0.00046422, 0.00068915, 0.00065904,
        0.00062954, 0.00056622, 0.00054305, 0.00059044, 0.0006459 ,
        0.00082138, 0.00084486, 0.00064991, 0.00058324, 0.00062771,
        0.00068748, 0.00066903, 0.00067511, 0.00063951, 0.00064476,
        0.00074031, 0.0005512 , 0.00046761, 0.00047545, 0.0004667 ,
        0.00047026, 0.00046744, 0.00047481, 0.0004545 , 0.00048974,
        0.00045543, 0.00044775, 0.00046937, 0.00046291, 0.00046964,
        0.00047629, 0.00045624, 0.00045516, 0.00046625, 0.00050259,
        0.00045297, 0.00045934, 0.00046766, 0.00047092, 0.00046749,
        0.00047762, 0.00048187, 0.00051813, 0.00048394, 0.00045936,
        0.00049047, 0.00046141, 0.0004626 , 0.00047541, 0.0004622 ,
        0.00068967, 0.00045948, 0.00046954, 0.0004813 , 0.00049036,
        0.00047855, 0.00061021, 0.00048151, 0.00046263, 0.00048175,
        0.0004657 , 0.00049314, 0.00052047, 0.00059369, 0.00050061,
        0.00047967, 0.00054545, 0.0004756 , 0.00046844, 0.00046394,
        0.00046782, 0.00046198, 0.00046079, 0.00046177, 0.00047028,
        0.00046482, 0.00046024, 0.00046508, 0.0004679 , 0.00064919,
        0.00078175, 0.00048306, 0.00046632, 0.00048583, 0.00046146]), 'std_score_time': array([5.40706854e-04, 1.13928412
e-05, 7.77537051e-06, 1.95003808e-05,
        2.42015880e-05, 5.37551421e-05, 1.92212445e-04, 3.30715017e-06,
        5.47871214e-05, 3.66296609e-06, 2.40576161e-05, 1.05590364e-05,
        8.14459956e-05, 2.37138172e-04, 3.39002464e-05, 1.69106014e-05,
        1.47539497e-05, 1.22059092e-04, 1.41862983e-04, 1.69565902e-05,
        2.66905846e-05, 1.96886548e-05, 3.74587077e-05, 7.43115946e-05,
        1.52158879e-05, 5.96659536e-05, 7.48634946e-05, 4.73797707e-05,
        3.75101534e-05, 4.00229172e-05, 3.85752788e-05, 3.29504568e-05,
        1.35204357e-05, 2.22444566e-04, 1.40440799e-04, 8.58814864e-05,
        7.93681412e-05, 6.99950863e-05, 6.82458971e-05, 1.49213595e-04,
        2.42031133e-04, 3.15968475e-04, 1.08447355e-04, 7.74900765e-05,
        5.54315557e-05, 1.47245247e-04, 9.64844052e-05, 1.30461165e-04,
        8.80711938e-05, 1.14200312e-04, 1.58033088e-04, 1.36034314e-04,
        1.66944257e-05, 4.04728863e-05, 1.80167006e-05, 2.41128776e-05,
        2.58578551e-05, 4.85742676e-05, 3.46167028e-06, 7.70529196e-05,
        1.23190213e-05, 2.89460186e-06, 2.46382387e-05, 2.32240257e-05,
        5.07430469e-05, 4.31349289e-05, 1.26569505e-05, 7.23273323e-06,
        1.57060490e-05, 7.87894588e-05, 6.14680828e-06, 2.35897564e-05,
        1.75088378e-05, 2.22419873e-05, 2.88229820e-05, 5.69279516e-05,
        4.68044819e-05, 1.13338325e-04, 4.13294051e-05, 1.18444133e-05,
        7.42932951e-05, 1.85865748e-05, 2.24617761e-05, 4.74071829e-05,
        1.54866403e-05, 2.28985203e-04, 1.31090332e-05, 4.99929366e-05,
        3.99571401e-05, 6.94192081e-05, 3.03151450e-05, 1.66799662e-04,
        3.60481007e-05, 2.26402765e-05, 6.75261186e-05, 1.64059183e-05,
        4.56121848e-05, 1.16873556e-04, 1.58796885e-04, 9.97999241e-05,
        3.14512785e-05, 1.14547749e-04, 3.42347495e-05, 1.73397032e-05,
```

```
        1.32258718e-05, 2.29030303e-05, 1.44616346e-05, 1.03546427e-05,
        1.23402287e-05, 3.76417938e-05, 2.10401390e-05, 1.18315205e-05,
        2.34155813e-05, 2.06463469e-05, 2.21894199e-04, 2.31736787e-04,
        5.51762048e-05, 1.32503471e-05, 4.23890600e-05, 1.34508020e-05]), 'param_criterion': masked_array(data=['gini',
  'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                    'gini', 'gini', 'gini', 'gini', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                    'entropy', 'entropy', 'entropy'],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False],
        fill_value='?',
             dtype=object), 'param_max_depth': masked_array(data=[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5,
                    5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9,
                    10, 10, 10, 10, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13,
                    13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 1, 1, 1, 1, 2,
                    2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6,
                    6, 7, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9, 10, 10, 10, 10,
                    11, 11, 11, 11, 12, 12, 12, 12, 13, 13, 13, 13, 14, 14,
                    14, 14, 15, 15, 15, 15],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
```

```
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False],
             fill_value='?',
                  dtype=object), 'param_min_samples_leaf': masked_array(data=[5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5,
10, 15,
                   20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10,
                   15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5,
                   10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15,
                   20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10,
                   15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5,
                   10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15,
                   20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10, 15, 20, 5, 10,
                   15, 20],
                mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False],
             fill_value='?',
                  dtype=object), 'params': [{'criterion': 'gini', 'max_depth': 1, 'min_samples_leaf': 5}, {'criterion': 'gin
i', 'max_depth': 1, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 1, 'min_samples_leaf': 15}, {'criterio
n': 'gini', 'max_depth': 1, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 5}, {'cri
terion': 'gini', 'max_depth': 2, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 15},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf':
5}, {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 3, 'min_samples_le
af': 15}, {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 4, 'min_samp
les_leaf': 5}, {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 4, 'min
_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth':
5, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_dep
th': 5, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 20}, {'criterion': 'gini', 'm
ax_depth': 6, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 6, 'min_samples_leaf': 10}, {'criterion': 'gin
i', 'max_depth': 6, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 6, 'min_samples_leaf': 20}, {'criterio
```

n': 'gini', 'max_depth': 7, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 10}, {'cri
terion': 'gini', 'max_depth': 7, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 20},
{'criterion': 'gini', 'max_depth': 8, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 8, 'min_samples_leaf':
10}, {'criterion': 'gini', 'max_depth': 8, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 8, 'min_samples_l
eaf': 20}, {'criterion': 'gini', 'max_depth': 9, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 9, 'min_samp
les_leaf': 10}, {'criterion': 'gini', 'max_depth': 9, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 9, 'mi
n_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth':
10, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_
depth': 10, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 11, 'min_samples_leaf': 5}, {'criterion': 'gin
i', 'max_depth': 11, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 11, 'min_samples_leaf': 15}, {'criterio
n': 'gini', 'max_depth': 11, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 12, 'min_samples_leaf': 5}, {'c
riterion': 'gini', 'max_depth': 12, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 12, 'min_samples_leaf':
15}, {'criterion': 'gini', 'max_depth': 12, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 13, 'min_samples
_leaf': 5}, {'criterion': 'gini', 'max_depth': 13, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_depth': 13, 'min_
samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 13, 'min_samples_leaf': 20}, {'criterion': 'gini', 'max_depth': 1
4, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 14, 'min_samples_leaf': 10}, {'criterion': 'gini', 'max_de
pth': 14, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 14, 'min_samples_leaf': 20}, {'criterion': 'gini',
'max_depth': 15, 'min_samples_leaf': 5}, {'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 10}, {'criterion':
'gini', 'max_depth': 15, 'min_samples_leaf': 15}, {'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 20}, {'crit
erion': 'entropy', 'max_depth': 1, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 1, 'min_samples_leaf':
10}, {'criterion': 'entropy', 'max_depth': 1, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 1, 'min_sam
ples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 2, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth':
2, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 2, 'min_samples_leaf': 15}, {'criterion': 'entropy',
'max_depth': 2, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 5}, {'criterion':
'entropy', 'max_depth': 3, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 15},
{'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 4, 'min_samples_
leaf': 5}, {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 4, 'm
in_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_d
epth': 5, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 10}, {'criterion': 'entro
py', 'max_depth': 5, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 20}, {'criter
ion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 1
0}, {'criterion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 6, 'min_samp
les_leaf': 20}, {'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth':
7, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 15}, {'criterion': 'entropy',
'max_depth': 7, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 5}, {'criterion':
'entropy', 'max_depth': 8, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 15},
{'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 9, 'min_samples_
leaf': 5}, {'criterion': 'entropy', 'max_depth': 9, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 9, 'm
in_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 9, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_d
epth': 10, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 10}, {'criterion': 'ent
ropy', 'max_depth': 10, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 20}, {'cr
iterion': 'entropy', 'max_depth': 11, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 11, 'min_samples_lea
f': 10}, {'criterion': 'entropy', 'max_depth': 11, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 11, 'm
in_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 12, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_d
epth': 12, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 12, 'min_samples_leaf': 15}, {'criterion': 'en
tropy', 'max_depth': 12, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 13, 'min_samples_leaf': 5}, {'cr
iterion': 'entropy', 'max_depth': 13, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max_depth': 13, 'min_samples_le
af': 15}, {'criterion': 'entropy', 'max_depth': 13, 'min_samples_leaf': 20}, {'criterion': 'entropy', 'max_depth': 14,
'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 14, 'min_samples_leaf': 10}, {'criterion': 'entropy', 'max
_depth': 14, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 14, 'min_samples_leaf': 20}, {'criterion':
'entropy', 'max_depth': 15, 'min_samples_leaf': 5}, {'criterion': 'entropy', 'max_depth': 15, 'min_samples_leaf': 10},
{'criterion': 'entropy', 'max_depth': 15, 'min_samples_leaf': 15}, {'criterion': 'entropy', 'max_depth': 15, 'min_sample

s_leaf': 20}], 'split0_test_score': array([0.54022989, 0.54022989, 0.54022989, 0.54022989, 0.54022989,
       0.54022989, 0.54022989, 0.54022989, 0.56321839, 0.56321839,
       0.56321839, 0.56321839, 0.55172414, 0.55172414, 0.55172414,
       0.55172414, 0.57471264, 0.55172414, 0.55172414, 0.55172414,
       0.57471264, 0.55172414, 0.57471264, 0.55172414, 0.57471264,
       0.55172414, 0.57471264, 0.55172414, 0.59770115, 0.55172414,
       0.57471264, 0.55172414, 0.59770115, 0.55172414, 0.57471264,
       0.55172414, 0.59770115, 0.55172414, 0.57471264, 0.55172414,
       0.59770115, 0.55172414, 0.57471264, 0.55172414, 0.59770115,
       0.55172414, 0.57471264, 0.55172414, 0.59770115, 0.55172414,
       0.57471264, 0.55172414, 0.59770115, 0.55172414, 0.57471264,
       0.55172414, 0.59770115, 0.55172414, 0.57471264, 0.55172414,
       0.54022989, 0.54022989, 0.54022989, 0.54022989, 0.54022989,
       0.54022989, 0.54022989, 0.54022989, 0.56321839, 0.56321839,
       0.56321839, 0.56321839, 0.55172414, 0.55172414, 0.55172414,
       0.55172414, 0.59770115, 0.55172414, 0.56321839, 0.55172414,
       0.55172414, 0.55172414, 0.55172414, 0.56321839, 0.55172414,
       0.55172414, 0.55172414, 0.56321839, 0.57471264, 0.55172414,
       0.55172414, 0.56321839, 0.57471264, 0.55172414, 0.55172414,
       0.56321839, 0.57471264, 0.55172414, 0.55172414, 0.56321839,
       0.57471264, 0.55172414, 0.55172414, 0.56321839, 0.57471264,
       0.55172414, 0.55172414, 0.56321839, 0.57471264, 0.55172414,
       0.55172414, 0.56321839, 0.57471264, 0.55172414, 0.55172414,
       0.56321839, 0.57471264, 0.55172414, 0.55172414, 0.56321839]), 'split1_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.53488372,
       0.53488372, 0.53488372, 0.53488372, 0.52325581, 0.52325581,
       0.52325581, 0.52325581, 0.55813953, 0.54651163, 0.54651163,
       0.54651163, 0.55813953, 0.54651163, 0.54651163, 0.54651163,
       0.56976744, 0.54651163, 0.54651163, 0.58139535, 0.55813953,
       0.59302326, 0.59302326, 0.58139535, 0.55813953, 0.59302326,
       0.59302326, 0.58139535, 0.60465116, 0.59302326, 0.59302326,
       0.58139535, 0.60465116, 0.59302326, 0.59302326, 0.58139535,
       0.60465116, 0.59302326, 0.59302326, 0.58139535, 0.60465116,
       0.59302326, 0.59302326, 0.58139535, 0.60465116, 0.59302326,
       0.59302326, 0.58139535, 0.60465116, 0.59302326, 0.59302326,
       0.58139535, 0.60465116, 0.59302326, 0.59302326, 0.58139535,
       0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.53488372,
       0.53488372, 0.53488372, 0.53488372, 0.52325581, 0.52325581,
       0.52325581, 0.52325581, 0.55813953, 0.54651163, 0.54651163,
       0.54651163, 0.55813953, 0.54651163, 0.54651163, 0.54651163,
       0.56976744, 0.54651163, 0.56976744, 0.58139535, 0.55813953,
       0.53488372, 0.60465116, 0.58139535, 0.55813953, 0.58139535,
       0.60465116, 0.58139535, 0.61627907, 0.58139535, 0.60465116,
       0.58139535, 0.61627907, 0.58139535, 0.60465116, 0.58139535,
       0.61627907, 0.58139535, 0.60465116, 0.58139535, 0.61627907,
       0.58139535, 0.60465116, 0.58139535, 0.61627907, 0.58139535,
       0.60465116, 0.58139535, 0.61627907, 0.58139535, 0.60465116,
       0.58139535, 0.61627907, 0.58139535, 0.60465116, 0.58139535]), 'split2_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.54651163,
       0.54651163, 0.54651163, 0.54651163, 0.52325581, 0.52325581,
       0.52325581, 0.52325581, 0.52325581, 0.52325581, 0.52325581,
       0.52325581, 0.55813953, 0.55813953, 0.55813953, 0.59302326,

```
        0.55813953, 0.61627907, 0.61627907, 0.59302326, 0.61627907,
        0.62790698, 0.61627907, 0.59302326, 0.61627907, 0.62790698,
        0.61627907, 0.59302326, 0.61627907, 0.62790698, 0.61627907,
        0.59302326, 0.61627907, 0.62790698, 0.61627907, 0.59302326,
        0.61627907, 0.62790698, 0.61627907, 0.59302326, 0.61627907,
        0.62790698, 0.61627907, 0.59302326, 0.61627907, 0.62790698,
        0.61627907, 0.59302326, 0.61627907, 0.62790698, 0.61627907,
        0.59302326, 0.61627907, 0.62790698, 0.61627907, 0.59302326,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.52325581, 0.52325581,
        0.52325581, 0.52325581, 0.52325581, 0.52325581, 0.52325581,
        0.52325581, 0.60465116, 0.60465116, 0.60465116, 0.59302326,
        0.54651163, 0.55813953, 0.55813953, 0.59302326, 0.53488372,
        0.62790698, 0.62790698, 0.59302326, 0.60465116, 0.62790698,
        0.62790698, 0.59302326, 0.60465116, 0.62790698, 0.62790698,
        0.59302326, 0.60465116, 0.62790698, 0.62790698, 0.59302326,
        0.60465116, 0.62790698, 0.62790698, 0.59302326, 0.60465116,
        0.62790698, 0.62790698, 0.59302326, 0.60465116, 0.62790698,
        0.62790698, 0.59302326, 0.60465116, 0.62790698, 0.62790698,
        0.59302326, 0.60465116, 0.62790698, 0.62790698, 0.59302326]), 'split3_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.59302326, 0.59302326,
        0.59302326, 0.59302326, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.59302326, 0.58139535, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.58139535, 0.58139535, 0.55813953,
        0.59302326, 0.58139535, 0.58139535, 0.58139535, 0.59302326,
        0.58139535, 0.58139535, 0.58139535, 0.59302326, 0.58139535,
        0.58139535, 0.61627907, 0.59302326, 0.58139535, 0.58139535,
        0.61627907, 0.59302326, 0.58139535, 0.58139535, 0.61627907,
        0.59302326, 0.58139535, 0.58139535, 0.61627907, 0.59302326,
        0.58139535, 0.58139535, 0.61627907, 0.59302326, 0.58139535,
        0.58139535, 0.61627907, 0.59302326, 0.58139535, 0.58139535,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.59302326, 0.59302326,
        0.59302326, 0.59302326, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.59302326, 0.58139535, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.58139535, 0.58139535, 0.55813953,
        0.59302326, 0.58139535, 0.58139535, 0.58139535, 0.59302326,
        0.58139535, 0.58139535, 0.58139535, 0.59302326, 0.58139535,
        0.58139535, 0.61627907, 0.59302326, 0.58139535, 0.58139535,
        0.61627907, 0.59302326, 0.58139535, 0.58139535, 0.61627907,
        0.59302326, 0.58139535, 0.58139535, 0.61627907, 0.59302326,
        0.58139535, 0.58139535, 0.61627907, 0.59302326, 0.58139535,
        0.58139535, 0.61627907, 0.59302326, 0.58139535, 0.58139535]), 'split4_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.55813953, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.60465116, 0.59302326, 0.58139535,
        0.55813953, 0.58139535, 0.56976744, 0.58139535, 0.59302326,
        0.59302326, 0.58139535, 0.58139535, 0.55813953, 0.59302326,
        0.58139535, 0.58139535, 0.56976744, 0.59302326, 0.58139535,
        0.58139535, 0.56976744, 0.59302326, 0.58139535, 0.58139535,
```

```
        0.56976744, 0.59302326, 0.58139535, 0.58139535, 0.56976744,
        0.59302326, 0.58139535, 0.58139535, 0.56976744, 0.59302326,
        0.58139535, 0.58139535, 0.56976744, 0.59302326, 0.58139535,
        0.58139535, 0.56976744, 0.59302326, 0.58139535, 0.58139535,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.55813953, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.60465116, 0.59302326, 0.58139535,
        0.55813953, 0.58139535, 0.56976744, 0.58139535, 0.59302326,
        0.59302326, 0.58139535, 0.58139535, 0.55813953, 0.59302326,
        0.58139535, 0.58139535, 0.56976744, 0.59302326, 0.58139535,
        0.58139535, 0.58139535, 0.59302326, 0.58139535, 0.58139535,
        0.58139535, 0.59302326, 0.58139535, 0.58139535, 0.58139535,
        0.59302326, 0.58139535, 0.58139535, 0.58139535, 0.59302326,
        0.58139535, 0.58139535, 0.58139535, 0.59302326, 0.58139535,
        0.58139535, 0.58139535, 0.59302326, 0.58139535, 0.58139535]), 'split5_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.53488372,
        0.53488372, 0.53488372, 0.53488372, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.53488372, 0.53488372, 0.53488372,
        0.54651163, 0.54651163, 0.51162791, 0.51162791, 0.56976744,
        0.53488372, 0.51162791, 0.54651163, 0.55813953, 0.52325581,
        0.54651163, 0.54651163, 0.55813953, 0.55813953, 0.54651163,
        0.54651163, 0.55813953, 0.55813953, 0.54651163, 0.54651163,
        0.55813953, 0.55813953, 0.54651163, 0.54651163, 0.55813953,
        0.55813953, 0.54651163, 0.54651163, 0.55813953, 0.55813953,
        0.54651163, 0.54651163, 0.55813953, 0.55813953, 0.54651163,
        0.54651163, 0.55813953, 0.55813953, 0.54651163, 0.54651163,
        0.55813953, 0.55813953, 0.54651163, 0.54651163, 0.55813953,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.53488372,
        0.53488372, 0.53488372, 0.53488372, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.53488372, 0.53488372, 0.53488372,
        0.54651163, 0.54651163, 0.51162791, 0.51162791, 0.56976744,
        0.53488372, 0.51162791, 0.54651163, 0.55813953, 0.53488372,
        0.54651163, 0.54651163, 0.55813953, 0.55813953, 0.54651163,
        0.54651163, 0.55813953, 0.55813953, 0.54651163, 0.54651163,
        0.55813953, 0.55813953, 0.54651163, 0.54651163, 0.55813953,
        0.55813953, 0.54651163, 0.54651163, 0.55813953, 0.55813953,
        0.54651163, 0.54651163, 0.55813953, 0.55813953, 0.54651163,
        0.54651163, 0.55813953, 0.55813953, 0.54651163, 0.54651163,
        0.55813953, 0.55813953, 0.54651163, 0.54651163, 0.55813953]), 'split6_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.56976744, 0.56976744, 0.56976744,
        0.56976744, 0.55813953, 0.56976744, 0.56976744, 0.63953488,
        0.61627907, 0.61627907, 0.62790698, 0.61627907, 0.60465116,
        0.61627907, 0.62790698, 0.61627907, 0.65116279, 0.61627907,
        0.62790698, 0.61627907, 0.65116279, 0.61627907, 0.62790698,
        0.61627907, 0.65116279, 0.61627907, 0.62790698, 0.61627907,
        0.65116279, 0.61627907, 0.62790698, 0.61627907, 0.65116279,
        0.61627907, 0.62790698, 0.61627907, 0.65116279, 0.61627907,
        0.62790698, 0.61627907, 0.65116279, 0.61627907, 0.62790698,
        0.61627907, 0.65116279, 0.61627907, 0.62790698, 0.61627907,
```

```
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.55813953, 0.55813953,
        0.55813953, 0.55813953, 0.56976744, 0.56976744, 0.56976744,
        0.56976744, 0.55813953, 0.56976744, 0.56976744, 0.56976744,
        0.61627907, 0.61627907, 0.62790698, 0.62790698, 0.60465116,
        0.61627907, 0.62790698, 0.62790698, 0.65116279, 0.61627907,
        0.62790698, 0.62790698, 0.65116279, 0.61627907, 0.62790698,
        0.62790698, 0.65116279, 0.61627907, 0.62790698, 0.62790698,
        0.65116279, 0.61627907, 0.62790698, 0.62790698, 0.65116279,
        0.61627907, 0.62790698, 0.62790698, 0.65116279, 0.61627907,
        0.62790698, 0.62790698, 0.65116279, 0.61627907, 0.62790698,
        0.62790698, 0.65116279, 0.61627907, 0.62790698, 0.62790698]), 'split7_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.58139535, 0.58139535, 0.58139535,
        0.58139535, 0.58139535, 0.58139535, 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.55813953, 0.56976744, 0.55813953,
        0.56976744, 0.55813953, 0.56976744, 0.58139535, 0.56976744,
        0.55813953, 0.56976744, 0.58139535, 0.56976744, 0.55813953,
        0.56976744, 0.58139535, 0.56976744, 0.55813953, 0.56976744,
        0.58139535, 0.56976744, 0.55813953, 0.56976744, 0.58139535,
        0.56976744, 0.55813953, 0.56976744, 0.58139535, 0.56976744,
        0.55813953, 0.56976744, 0.58139535, 0.56976744, 0.55813953,
        0.56976744, 0.58139535, 0.56976744, 0.55813953, 0.56976744,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.58139535, 0.58139535, 0.58139535,
        0.58139535, 0.58139535, 0.58139535, 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.55813953, 0.56976744, 0.56976744,
        0.56976744, 0.55813953, 0.56976744, 0.58139535, 0.56976744,
        0.55813953, 0.56976744, 0.58139535, 0.56976744, 0.55813953,
        0.56976744, 0.58139535, 0.56976744, 0.55813953, 0.56976744,
        0.58139535, 0.56976744, 0.55813953, 0.56976744, 0.58139535,
        0.56976744, 0.55813953, 0.56976744, 0.58139535, 0.56976744,
        0.55813953, 0.56976744, 0.58139535, 0.56976744, 0.55813953,
        0.56976744, 0.58139535, 0.56976744, 0.55813953, 0.56976744]), 'split8_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.5       ,
        0.5       , 0.5       , 0.5       , 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.56976744, 0.56976744, 0.56976744,
        0.56976744, 0.55813953, 0.53488372, 0.53488372, 0.55813953,
        0.54651163, 0.56976744, 0.60465116, 0.61627907, 0.56976744,
        0.59302326, 0.60465116, 0.61627907, 0.60465116, 0.59302326,
        0.60465116, 0.61627907, 0.60465116, 0.59302326, 0.60465116,
        0.61627907, 0.60465116, 0.59302326, 0.60465116, 0.61627907,
        0.60465116, 0.59302326, 0.60465116, 0.61627907, 0.60465116,
        0.59302326, 0.60465116, 0.61627907, 0.60465116, 0.59302326,
        0.60465116, 0.61627907, 0.60465116, 0.59302326, 0.60465116,
        0.61627907, 0.60465116, 0.59302326, 0.60465116, 0.61627907,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.5       ,
        0.5       , 0.5       , 0.5       , 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.56976744, 0.56976744, 0.56976744,
        0.56976744, 0.55813953, 0.53488372, 0.53488372, 0.55813953,
```

```
        0.54651163, 0.56976744, 0.60465116, 0.61627907, 0.56976744,
        0.59302326, 0.60465116, 0.61627907, 0.60465116, 0.59302326,
        0.60465116, 0.61627907, 0.60465116, 0.59302326, 0.60465116,
        0.61627907, 0.60465116, 0.59302326, 0.60465116, 0.61627907,
        0.60465116, 0.59302326, 0.60465116, 0.61627907, 0.60465116,
        0.59302326, 0.60465116, 0.61627907, 0.60465116, 0.59302326,
        0.60465116, 0.61627907, 0.60465116, 0.59302326, 0.60465116,
        0.61627907, 0.60465116, 0.59302326, 0.60465116, 0.61627907]), 'split9_test_score': array([0.54651163, 0.54651163,
0.54651163, 0.54651163, 0.5       ,
        0.5       , 0.5       , 0.5       , 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.56976744, 0.56976744, 0.56976744, 0.56976744,
        0.56976744, 0.56976744, 0.56976744, 0.56976744, 0.55813953,
        0.58139535, 0.56976744, 0.56976744, 0.55813953, 0.58139535,
        0.56976744, 0.56976744, 0.55813953, 0.58139535, 0.56976744,
        0.56976744, 0.55813953, 0.58139535, 0.56976744, 0.56976744,
        0.55813953, 0.58139535, 0.56976744, 0.56976744, 0.55813953,
        0.58139535, 0.56976744, 0.56976744, 0.55813953, 0.58139535,
        0.56976744, 0.56976744, 0.55813953, 0.58139535, 0.56976744,
        0.56976744, 0.55813953, 0.58139535, 0.56976744, 0.56976744,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.5       ,
        0.5       , 0.5       , 0.5       , 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.55813953, 0.53488372, 0.52325581, 0.52325581, 0.52325581,
        0.53488372, 0.52325581, 0.52325581, 0.56976744, 0.54651163,
        0.56976744, 0.56976744, 0.56976744, 0.54651163, 0.56976744,
        0.56976744, 0.56976744, 0.58139535, 0.56976744, 0.56976744,
        0.56976744, 0.58139535, 0.56976744, 0.56976744, 0.56976744,
        0.55813953, 0.56976744, 0.56976744, 0.56976744, 0.55813953,
        0.56976744, 0.56976744, 0.56976744, 0.55813953, 0.56976744,
        0.56976744, 0.56976744, 0.55813953, 0.56976744, 0.56976744,
        0.56976744, 0.55813953, 0.56976744, 0.56976744, 0.56976744]), 'mean_test_score': array([0.54588345, 0.54588345,
0.54588345, 0.54588345, 0.53425555,
        0.53425555, 0.53425555, 0.53425555, 0.55515905, 0.55515905,
        0.55515905, 0.55515905, 0.55400962, 0.55284683, 0.55284683,
        0.55400962, 0.5656108 , 0.56098637, 0.5563352 , 0.57377706,
        0.5656108 , 0.5691259 , 0.57956429, 0.5819166 , 0.57142475,
        0.58656776, 0.58537824, 0.5819166 , 0.5865143 , 0.58656776,
        0.58537824, 0.5819166 , 0.59232825, 0.58656776, 0.58537824,
        0.5819166 , 0.59581663, 0.58656776, 0.58537824, 0.5819166 ,
        0.59581663, 0.58656776, 0.58537824, 0.5819166 , 0.59581663,
        0.58656776, 0.58537824, 0.5819166 , 0.59581663, 0.58656776,
        0.58537824, 0.5819166 , 0.59581663, 0.58656776, 0.58537824,
        0.5819166 , 0.59581663, 0.58656776, 0.58537824, 0.5819166 ,
        0.54588345, 0.54588345, 0.54588345, 0.54588345, 0.53425555,
        0.53425555, 0.53425555, 0.53425555, 0.55515905, 0.55515905,
        0.55515905, 0.55515905, 0.55400962, 0.55284683, 0.55284683,
        0.55517241, 0.56907244, 0.56098637, 0.55748463, 0.56214916,
        0.55866079, 0.55866079, 0.5691259 , 0.58422882, 0.56214916,
        0.57959102, 0.58540497, 0.58422882, 0.58188987, 0.58424218,
        0.58540497, 0.58422882, 0.59235499, 0.58424218, 0.58540497,
        0.58422882, 0.59700615, 0.58424218, 0.58540497, 0.58422882,
```

```
        0.59468057, 0.58424218, 0.58540497, 0.58422882, 0.59468057,
        0.58424218, 0.58540497, 0.58422882, 0.59468057, 0.58424218,
        0.58540497, 0.58422882, 0.59468057, 0.58424218, 0.58540497,
        0.58422882, 0.59468057, 0.58424218, 0.58540497, 0.58422882]), 'std_test_score': array([0.00188452, 0.00188452, 0.
00188452, 0.00188452, 0.01770431,
        0.01770431, 0.01770431, 0.01770431, 0.02031851, 0.02031851,
        0.02031851, 0.02031851, 0.02031851, 0.01646198, 0.01653968, 0.01653968,
        0.01561906, 0.01320831, 0.02512946, 0.02114349, 0.02553346,
        0.02044099, 0.0297314 , 0.02679023, 0.02063461, 0.02573362,
        0.024261  , 0.02415613, 0.02063461, 0.02965194, 0.024261  ,
        0.02415613, 0.02063461, 0.02730111, 0.024261  , 0.02415613,
        0.02063461, 0.02790328, 0.024261  , 0.02415613, 0.02063461,
        0.02790328, 0.024261  , 0.02415613, 0.02063461, 0.02790328,
        0.024261  , 0.02415613, 0.02063461, 0.02790328, 0.024261  ,
        0.02415613, 0.02063461, 0.02790328, 0.024261  , 0.02415613,
        0.02063461, 0.02790328, 0.024261  , 0.02415613, 0.02063461,
        0.00188452, 0.00188452, 0.00188452, 0.00188452, 0.01770431,
        0.01770431, 0.01770431, 0.01770431, 0.02031851, 0.02031851,
        0.02031851, 0.02031851, 0.01646198, 0.01653968, 0.01653968,
        0.01544949, 0.02228658, 0.03091908, 0.02779995, 0.0184277 ,
        0.02245249, 0.0278304 , 0.02833428, 0.02139341, 0.02178964,
        0.02878242, 0.02832111, 0.02139341, 0.02971657, 0.02464236,
        0.02832111, 0.02139341, 0.02585653, 0.02464236, 0.02832111,
        0.02139341, 0.02555666, 0.02464236, 0.02832111, 0.02139341,
        0.02782852, 0.02464236, 0.02832111, 0.02139341, 0.02782852,
        0.02464236, 0.02832111, 0.02139341, 0.02782852, 0.02464236,
        0.02832111, 0.02139341, 0.02782852, 0.02464236, 0.02832111,
        0.02139341, 0.02782852, 0.02464236, 0.02832111, 0.02139341]), 'rank_test_score': array([105, 105, 105, 105, 113,
113, 113, 113,  90,  90,  90,  90,  98,
        101, 101,  98,  79,  83,  88,  74,  80,  76,  73,  61,  75,  15,
         34,  61,  24,  15,  34,  61,  14,  15,  34,  61,   2,  15,  34,
         61,   2,  15,  34,  61,   2,  15,  34,  61,   2,  15,  34,  61,
          2,  15,  34,  61,   2,  15,  34,  61, 105, 105, 105, 105, 113,
        113, 113, 113,  90,  90,  90,  98, 101, 101,  89,  78,  83,
         87,  81,  86,  85,  76,  51,  82,  72,  25,  51,  71,  43,  25,
         51,  13,  43,  25,  51,   1,  43,  25,  51,   8,  43,  25,  51,
          8,  43,  25,  51,   8,  43,  25,  51,   8,  43,  25,  51,   8,
         43,  25,  51], dtype=int32), 'split0_train_score': array([0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.5465
1163,
        0.54651163, 0.54651163, 0.54651163, 0.58139535, 0.58139535,
        0.58139535, 0.58139535, 0.5878553 , 0.5878553 , 0.5878553 ,
        0.5878553 , 0.59173127, 0.60206718, 0.60206718, 0.60206718,
        0.61498708, 0.60981912, 0.60852713, 0.60206718, 0.62015504,
        0.60981912, 0.60852713, 0.60206718, 0.62790698, 0.60981912,
        0.60852713, 0.60206718, 0.62790698, 0.60981912, 0.60852713,
        0.60206718, 0.62790698, 0.60981912, 0.60852713, 0.60206718,
        0.62790698, 0.60981912, 0.60852713, 0.60206718, 0.62790698,
        0.60981912, 0.60852713, 0.60206718, 0.62790698, 0.60981912,
        0.60852713, 0.60206718, 0.62790698, 0.60981912, 0.60852713,
        0.60206718, 0.62790698, 0.60981912, 0.60852713, 0.60206718,
        0.54651163, 0.54651163, 0.54651163, 0.54651163, 0.54651163,
        0.54651163, 0.54651163, 0.54651163, 0.58139535, 0.58139535,
```

```
        0.58139535, 0.58139535, 0.58397933, 0.58397933, 0.58397933,
        0.58397933, 0.59173127, 0.59302326, 0.59431525, 0.60206718,
        0.5994832 , 0.60465116, 0.60465116, 0.60335917, 0.61498708,
        0.6124031 , 0.60465116, 0.60335917, 0.62403101, 0.6124031 ,
        0.60465116, 0.60335917, 0.62403101, 0.6124031 , 0.60465116,
        0.60335917, 0.62403101, 0.6124031 , 0.60465116, 0.60335917,
        0.62403101, 0.6124031 , 0.60465116, 0.60335917, 0.62403101,
        0.6124031 , 0.60465116, 0.60335917, 0.62403101, 0.6124031 ,
        0.60465116, 0.60335917, 0.62403101, 0.6124031 , 0.60465116,
        0.60335917, 0.62403101, 0.6124031 , 0.60465116, 0.60335917]), 'split1_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.5483871 ,
        0.5483871 , 0.5483871 , 0.5483871 , 0.57548387, 0.57548387,
        0.57548387, 0.57548387, 0.58322581, 0.58193548, 0.58193548,
        0.58193548, 0.59354839, 0.59225806, 0.59225806, 0.59225806,
        0.59870968, 0.59225806, 0.59225806, 0.59870968, 0.60645161,
        0.60903226, 0.60774194, 0.59870968, 0.60645161, 0.60903226,
        0.60774194, 0.59870968, 0.62064516, 0.60903226, 0.60774194,
        0.59870968, 0.62064516, 0.60903226, 0.60774194, 0.59870968,
        0.62064516, 0.60903226, 0.60774194, 0.59870968, 0.62064516,
        0.60903226, 0.60774194, 0.59870968, 0.62064516, 0.60903226,
        0.60774194, 0.59870968, 0.62064516, 0.60903226, 0.60774194,
        0.59870968, 0.62064516, 0.60903226, 0.60774194, 0.59870968,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.5483871 ,
        0.5483871 , 0.5483871 , 0.5483871 , 0.57548387, 0.57548387,
        0.57548387, 0.57548387, 0.58322581, 0.58193548, 0.58193548,
        0.58193548, 0.59096774, 0.58967742, 0.58967742, 0.59225806,
        0.59483871, 0.58967742, 0.59096774, 0.59870968, 0.60258065,
        0.59612903, 0.60516129, 0.59870968, 0.60258065, 0.60774194,
        0.60516129, 0.59870968, 0.61806452, 0.60774194, 0.60516129,
        0.59870968, 0.61806452, 0.60774194, 0.60516129, 0.59870968,
        0.61806452, 0.60774194, 0.60516129, 0.59870968, 0.61806452,
        0.60774194, 0.60516129, 0.59870968, 0.61806452, 0.60774194,
        0.60516129, 0.59870968, 0.61806452, 0.60774194, 0.60516129,
        0.59870968, 0.61806452, 0.60774194, 0.60516129, 0.59870968]), 'split2_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.58193548, 0.58193548,
        0.58193548, 0.58193548, 0.58322581, 0.58193548, 0.58193548,
        0.58193548, 0.59612903, 0.59483871, 0.59483871, 0.6       ,
        0.59741935, 0.60258065, 0.60258065, 0.6       , 0.60903226,
        0.60645161, 0.60258065, 0.6       , 0.60903226, 0.60645161,
        0.60258065, 0.6       , 0.61548387, 0.60645161, 0.60258065,
        0.6       , 0.61548387, 0.60645161, 0.60258065, 0.6       ,
        0.61548387, 0.60645161, 0.60258065, 0.6       , 0.61548387,
        0.60645161, 0.60258065, 0.6       , 0.61548387, 0.60645161,
        0.60258065, 0.6       , 0.61548387, 0.60645161, 0.60258065,
        0.6       , 0.61548387, 0.60645161, 0.60258065, 0.6       ,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.58193548, 0.58193548,
        0.58193548, 0.58193548, 0.58322581, 0.58193548, 0.58193548,
        0.58193548, 0.58580645, 0.58451613, 0.58451613, 0.6       ,
        0.60129032, 0.59741935, 0.59741935, 0.6       , 0.60129032,
        0.60645161, 0.60645161, 0.6       , 0.61290323, 0.60645161,
```

```
        0.60645161, 0.6       , 0.61419355, 0.60645161, 0.60645161,
        0.6       , 0.61419355, 0.60645161, 0.60645161, 0.6       ,
        0.61419355, 0.60645161, 0.60645161, 0.6       , 0.61419355,
        0.60645161, 0.60645161, 0.6       , 0.61419355, 0.60645161,
        0.60645161, 0.6       , 0.61419355, 0.60645161, 0.60645161,
        0.6       , 0.61419355, 0.60645161, 0.60645161, 0.6       ]), 'split3_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.57806452, 0.57806452,
        0.57806452, 0.57806452, 0.58451613, 0.58451613, 0.58451613,
        0.58451613, 0.59096774, 0.58709677, 0.58967742, 0.58967742,
        0.6       , 0.59225806, 0.60516129, 0.60516129, 0.6       ,
        0.60645161, 0.60516129, 0.60516129, 0.60774194, 0.60645161,
        0.60516129, 0.60516129, 0.60774194, 0.60645161, 0.60516129,
        0.60516129, 0.62322581, 0.60645161, 0.60516129, 0.60516129,
        0.62322581, 0.60645161, 0.60516129, 0.60516129, 0.62322581,
        0.60645161, 0.60516129, 0.60516129, 0.62322581, 0.60645161,
        0.60516129, 0.60516129, 0.62322581, 0.60645161, 0.60516129,
        0.60516129, 0.62322581, 0.60645161, 0.60516129, 0.60516129,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.57806452, 0.57806452,
        0.57806452, 0.57806452, 0.58451613, 0.58451613, 0.58451613,
        0.58451613, 0.59096774, 0.58709677, 0.58967742, 0.58967742,
        0.6       , 0.59225806, 0.60516129, 0.60516129, 0.6       ,
        0.60645161, 0.60516129, 0.60516129, 0.60774194, 0.60645161,
        0.60516129, 0.60516129, 0.60774194, 0.60645161, 0.60516129,
        0.60516129, 0.62322581, 0.60645161, 0.60516129, 0.60516129,
        0.62322581, 0.60645161, 0.60516129, 0.60516129, 0.62322581,
        0.60645161, 0.60516129, 0.60516129, 0.62322581, 0.60645161,
        0.60516129, 0.60516129, 0.62322581, 0.60645161, 0.60516129,
        0.60516129, 0.62322581, 0.60645161, 0.60516129, 0.60516129]), 'split4_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.58193548, 0.58193548,
        0.58193548, 0.58193548, 0.58322581, 0.58322581, 0.58322581,
        0.58193548, 0.59483871, 0.59096774, 0.59096774, 0.59870968,
        0.59483871, 0.59741935, 0.59354839, 0.60129032, 0.6       ,
        0.61419355, 0.61290323, 0.60129032, 0.60387097, 0.61419355,
        0.61290323, 0.60129032, 0.62451613, 0.61419355, 0.61290323,
        0.60129032, 0.62451613, 0.61419355, 0.61290323, 0.60129032,
        0.62451613, 0.61419355, 0.61290323, 0.60129032, 0.62451613,
        0.61419355, 0.61290323, 0.60129032, 0.62451613, 0.61419355,
        0.61290323, 0.60129032, 0.62451613, 0.61419355, 0.61290323,
        0.60129032, 0.62451613, 0.61419355, 0.61290323, 0.60129032,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.58193548, 0.58193548,
        0.58193548, 0.58193548, 0.58322581, 0.58322581, 0.58322581,
        0.58193548, 0.59483871, 0.59096774, 0.59096774, 0.59870968,
        0.59483871, 0.59741935, 0.59354839, 0.60129032, 0.6       ,
        0.61419355, 0.61290323, 0.60129032, 0.60129032, 0.61419355,
        0.61290323, 0.60129032, 0.61935484, 0.61419355, 0.61290323,
        0.60129032, 0.62322581, 0.61419355, 0.61290323, 0.60129032,
        0.62322581, 0.61419355, 0.61290323, 0.60129032, 0.62322581,
        0.61419355, 0.61290323, 0.60129032, 0.62322581, 0.61419355,
```

```
        0.61290323, 0.60129032, 0.62322581, 0.61419355, 0.61290323,
        0.60129032, 0.62322581, 0.61419355, 0.61290323, 0.60129032]), 'split5_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.5483871 ,
        0.5483871 , 0.5483871 , 0.5483871 , 0.58322581, 0.58322581,
        0.58322581, 0.58322581, 0.58580645, 0.58580645, 0.58580645,
        0.58322581, 0.59741935, 0.59612903, 0.59483871, 0.6       ,
        0.59870968, 0.59612903, 0.61032258, 0.60645161, 0.60774194,
        0.61419355, 0.61032258, 0.60645161, 0.62193548, 0.61419355,
        0.61032258, 0.60645161, 0.62193548, 0.61419355, 0.61032258,
        0.60645161, 0.62193548, 0.61419355, 0.61032258, 0.60645161,
        0.62193548, 0.61419355, 0.61032258, 0.60645161, 0.62193548,
        0.61419355, 0.61032258, 0.60645161, 0.62193548, 0.61419355,
        0.61032258, 0.60645161, 0.62193548, 0.61419355, 0.61032258,
        0.60645161, 0.62193548, 0.61419355, 0.61032258, 0.60645161,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.5483871 ,
        0.5483871 , 0.5483871 , 0.5483871 , 0.58322581, 0.58322581,
        0.58322581, 0.58322581, 0.58580645, 0.58580645, 0.58580645,
        0.58322581, 0.59741935, 0.59612903, 0.59483871, 0.6       ,
        0.59870968, 0.59612903, 0.61032258, 0.60645161, 0.60516129,
        0.61419355, 0.61032258, 0.60645161, 0.62193548, 0.61419355,
        0.61032258, 0.60645161, 0.62193548, 0.61419355, 0.61032258,
        0.60645161, 0.62193548, 0.61419355, 0.61032258, 0.60645161,
        0.62193548, 0.61419355, 0.61032258, 0.60645161, 0.62193548,
        0.61419355, 0.61032258, 0.60645161, 0.62193548, 0.61419355,
        0.61032258, 0.60645161, 0.62193548, 0.61419355, 0.61032258,
        0.60645161, 0.62193548, 0.61419355, 0.61032258, 0.60645161]), 'split6_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.58193548, 0.58193548,
        0.58193548, 0.58193548, 0.58580645, 0.58580645, 0.58580645,
        0.58451613, 0.59096774, 0.58580645, 0.58580645, 0.59096774,
        0.59870968, 0.6       , 0.60129032, 0.59870968, 0.60387097,
        0.60516129, 0.60129032, 0.59870968, 0.61677419, 0.60516129,
        0.60129032, 0.59870968, 0.61677419, 0.60516129, 0.60129032,
        0.59870968, 0.61677419, 0.60516129, 0.60129032, 0.59870968,
        0.61677419, 0.60516129, 0.60129032, 0.59870968, 0.61677419,
        0.60516129, 0.60129032, 0.59870968, 0.61677419, 0.60516129,
        0.60129032, 0.59870968, 0.61677419, 0.60516129, 0.60129032,
        0.59870968, 0.61677419, 0.60516129, 0.60129032, 0.59870968,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.58193548, 0.58193548,
        0.58193548, 0.58193548, 0.58580645, 0.58580645, 0.58580645,
        0.58451613, 0.59096774, 0.58580645, 0.58580645, 0.58451613,
        0.59870968, 0.6       , 0.60129032, 0.60258065, 0.60387097,
        0.60516129, 0.60129032, 0.60258065, 0.61677419, 0.60516129,
        0.60129032, 0.60258065, 0.61677419, 0.60516129, 0.60129032,
        0.60258065, 0.61677419, 0.60516129, 0.60129032, 0.60258065,
        0.61677419, 0.60516129, 0.60129032, 0.60258065, 0.61677419,
        0.60516129, 0.60129032, 0.60258065, 0.61677419, 0.60516129,
        0.60129032, 0.60258065, 0.61677419, 0.60516129, 0.60129032,
        0.60258065, 0.61677419, 0.60516129, 0.60129032, 0.60258065]), 'split7_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.57677419, 0.57677419,
```

```
        0.57677419, 0.57677419, 0.57806452, 0.57806452, 0.57806452,
        0.57806452, 0.59096774, 0.5883871 , 0.60258065, 0.60258065,
        0.6116129 , 0.60774194, 0.60516129, 0.60258065, 0.61419355,
        0.61032258, 0.60516129, 0.60258065, 0.62193548, 0.61032258,
        0.60516129, 0.60258065, 0.62193548, 0.61032258, 0.60516129,
        0.60258065, 0.62193548, 0.61032258, 0.60516129, 0.60258065,
        0.62193548, 0.61032258, 0.60516129, 0.60258065, 0.62193548,
        0.61032258, 0.60516129, 0.60258065, 0.62193548, 0.61032258,
        0.60516129, 0.60258065, 0.62193548, 0.61032258, 0.60516129,
        0.60258065, 0.62193548, 0.61032258, 0.60516129, 0.60258065,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.54580645,
        0.54580645, 0.54580645, 0.54580645, 0.57677419, 0.57677419,
        0.57677419, 0.57677419, 0.57806452, 0.57806452, 0.57806452,
        0.57806452, 0.5883871 , 0.5883871 , 0.60258065, 0.60258065,
        0.61032258, 0.60774194, 0.60516129, 0.60258065, 0.61548387,
        0.61032258, 0.60516129, 0.60258065, 0.6283871 , 0.61032258,
        0.60516129, 0.60258065, 0.6283871 , 0.61032258, 0.60516129,
        0.60258065, 0.6283871 , 0.61032258, 0.60516129, 0.60258065,
        0.6283871 , 0.61032258, 0.60516129, 0.60258065, 0.6283871 ,
        0.61032258, 0.60516129, 0.60258065, 0.6283871 , 0.61032258,
        0.60516129, 0.60258065, 0.6283871 , 0.61032258, 0.60516129,
        0.60258065, 0.6283871 , 0.61032258, 0.60516129, 0.60258065]), 'split8_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.55225806,
        0.55225806, 0.55225806, 0.55225806, 0.58322581, 0.58322581,
        0.58322581, 0.58322581, 0.58709677, 0.58580645, 0.58322581,
        0.58322581, 0.59096774, 0.59096774, 0.5883871 , 0.58967742,
        0.59870968, 0.59354839, 0.60129032, 0.60387097, 0.6       ,
        0.60645161, 0.60129032, 0.60387097, 0.61290323, 0.60645161,
        0.60129032, 0.60387097, 0.61290323, 0.60645161, 0.60129032,
        0.60387097, 0.61290323, 0.60645161, 0.60129032, 0.60387097,
        0.61290323, 0.60645161, 0.60129032, 0.60387097, 0.61290323,
        0.60645161, 0.60129032, 0.60387097, 0.61290323, 0.60645161,
        0.60129032, 0.60387097, 0.61290323, 0.60645161, 0.60129032,
        0.60387097, 0.61290323, 0.60645161, 0.60129032, 0.60387097,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.55225806,
        0.55225806, 0.55225806, 0.55225806, 0.58322581, 0.58322581,
        0.58322581, 0.58322581, 0.58709677, 0.58580645, 0.58322581,
        0.58322581, 0.58967742, 0.59096774, 0.5883871 , 0.58967742,
        0.59741935, 0.59354839, 0.60129032, 0.60387097, 0.60129032,
        0.60645161, 0.60129032, 0.60387097, 0.61548387, 0.60645161,
        0.60129032, 0.60387097, 0.61548387, 0.60645161, 0.60129032,
        0.60387097, 0.61548387, 0.60645161, 0.60129032, 0.60387097,
        0.61548387, 0.60645161, 0.60129032, 0.60387097, 0.61548387,
        0.60645161, 0.60129032, 0.60387097, 0.61548387, 0.60645161,
        0.60129032, 0.60387097, 0.61548387, 0.60645161, 0.60129032,
        0.60387097, 0.61548387, 0.60645161, 0.60129032, 0.60387097]), 'split9_train_score': array([0.54580645, 0.5458064
5, 0.54580645, 0.54580645, 0.55225806,
        0.55225806, 0.55225806, 0.55225806, 0.58580645, 0.58580645,
        0.58580645, 0.58580645, 0.58967742, 0.5883871 , 0.58580645,
        0.58580645, 0.60774194, 0.60645161, 0.60387097, 0.60387097,
        0.60774194, 0.60645161, 0.60387097, 0.60387097, 0.62193548,
        0.61419355, 0.60387097, 0.60387097, 0.62193548, 0.61419355,
```

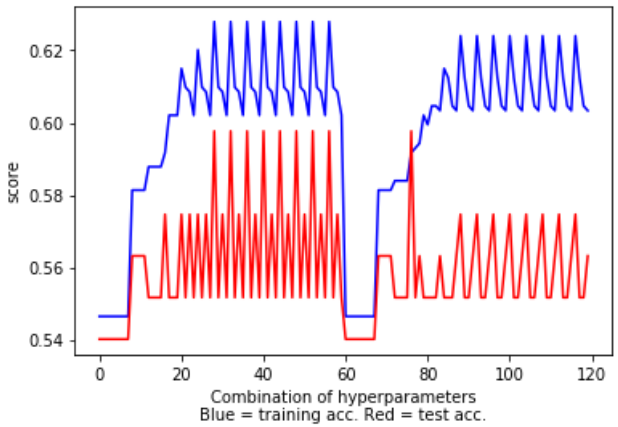
```
        0.60387097, 0.60387097, 0.62193548, 0.61419355, 0.60387097,
        0.60387097, 0.62193548, 0.61419355, 0.60387097, 0.60387097,
        0.62193548, 0.61419355, 0.60387097, 0.60387097, 0.62193548,
        0.61419355, 0.60387097, 0.60387097, 0.62193548, 0.61419355,
        0.60387097, 0.60387097, 0.62193548, 0.61419355, 0.60387097,
        0.60387097, 0.62193548, 0.61419355, 0.60387097, 0.60387097,
        0.54580645, 0.54580645, 0.54580645, 0.54580645, 0.55225806,
        0.55225806, 0.55225806, 0.55225806, 0.57806452, 0.57806452,
        0.57806452, 0.57806452, 0.57806452, 0.57806452, 0.57806452,
        0.58064516, 0.58709677, 0.59741935, 0.59741935, 0.59741935,
        0.58709677, 0.59741935, 0.59741935, 0.60774194, 0.59870968,
        0.61290323, 0.61290323, 0.60774194, 0.59870968, 0.61290323,
        0.61290323, 0.60774194, 0.61419355, 0.61290323, 0.61290323,
        0.60774194, 0.61419355, 0.61290323, 0.61290323, 0.60774194,
        0.62580645, 0.61290323, 0.61290323, 0.60774194, 0.62580645,
        0.61290323, 0.61290323, 0.60774194, 0.62580645, 0.61290323,
        0.61290323, 0.60774194, 0.62580645, 0.61290323, 0.61290323,
        0.60774194, 0.62580645, 0.61290323, 0.61290323, 0.60774194]), 'mean_train_score': array([0.54587697, 0.54587697,
0.54587697, 0.54587697, 0.54768342,
        0.54768342, 0.54768342, 0.54768342, 0.58097824, 0.58097824,
        0.58097824, 0.58097824, 0.58485005, 0.58433392, 0.58381779,
        0.58330166, 0.59452797, 0.59349704, 0.5945293 , 0.59698091,
        0.60214387, 0.59982062, 0.6024011 , 0.60227123, 0.60833808,
        0.60962707, 0.60588497, 0.60227123, 0.61504876, 0.60962707,
        0.60588497, 0.60227123, 0.61917779, 0.60962707, 0.60588497,
        0.60227123, 0.62072618, 0.60962707, 0.60588497, 0.60227123,
        0.62072618, 0.60962707, 0.60588497, 0.60227123, 0.62072618,
        0.60962707, 0.60588497, 0.60227123, 0.62072618, 0.60962707,
        0.60588497, 0.60227123, 0.62072618, 0.60962707, 0.60588497,
        0.60227123, 0.62072618, 0.60962707, 0.60588497, 0.60227123,
        0.54587697, 0.54587697, 0.54587697, 0.54587697, 0.54768342,
        0.54768342, 0.54768342, 0.54768342, 0.58020405, 0.58020405,
        0.58020405, 0.58020405, 0.58330116, 0.58291406, 0.582656  ,
        0.58239793, 0.59078603, 0.5903991 , 0.59181862, 0.59569059,
        0.5982709 , 0.59762641, 0.60072318, 0.60317463, 0.60433742,
        0.60846612, 0.60652963, 0.60317463, 0.61298375, 0.60962741,
        0.60652963, 0.60317463, 0.618016  , 0.60962741, 0.60652963,
        0.60317463, 0.61995149, 0.60962741, 0.60652963, 0.60317463,
        0.62111278, 0.60962741, 0.60652963, 0.60317463, 0.62111278,
        0.60962741, 0.60652963, 0.60317463, 0.62111278, 0.60962741,
        0.60652963, 0.60317463, 0.62111278, 0.60962741, 0.60652963,
        0.60317463, 0.62111278, 0.60962741, 0.60652963, 0.60317463]), 'std_train_score': array([0.00021155, 0.00021155,
0.00021155, 0.00021155, 0.00248663,
        0.00248663, 0.00248663, 0.00248663, 0.0030445 , 0.0030445 ,
        0.0030445 , 0.0030445 , 0.0030542 , 0.00295643, 0.00263706,
        0.00250994, 0.00495087, 0.00624936, 0.00603573, 0.00539503,
        0.00642967, 0.0062246 , 0.00549024, 0.00250147, 0.00769245,
        0.00337085, 0.00370974, 0.00250147, 0.00776941, 0.00337085,
        0.00370974, 0.00250147, 0.00565009, 0.00337085, 0.00370974,
        0.00250147, 0.00425286, 0.00337085, 0.00370974, 0.00250147,
        0.00425286, 0.00337085, 0.00370974, 0.00250147, 0.00425286,
        0.00337085, 0.00370974, 0.00250147, 0.00425286, 0.00337085,
```

```
0.00370974, 0.00250147, 0.00425286, 0.00337085, 0.00370974,
0.00250147, 0.00425286, 0.00337085, 0.00370974, 0.00250147,
0.00021155, 0.00021155, 0.00021155, 0.00021155, 0.00248663,
0.00248663, 0.00248663, 0.00248663, 0.00268093, 0.00268093,
0.00268093, 0.00268093, 0.00289409, 0.00279012, 0.00262513,
0.00187982, 0.00326816, 0.0040103 , 0.00523149, 0.00588512,
0.00555293, 0.00517988, 0.0056222 , 0.00265207, 0.00574029,
0.00528717, 0.00399605, 0.00265207, 0.009676  , 0.00338196,
0.00399605, 0.00265207, 0.00551026, 0.00338196, 0.00399605,
0.00265207, 0.00461632, 0.00338196, 0.00399605, 0.00265207,
0.00448045, 0.00338196, 0.00399605, 0.00265207, 0.00448045,
0.00338196, 0.00399605, 0.00265207, 0.00448045, 0.00338196,
0.00399605, 0.00265207, 0.00448045, 0.00338196, 0.00399605,
0.00265207, 0.00448045, 0.00338196, 0.00399605, 0.00265207])}
```

In [23]:

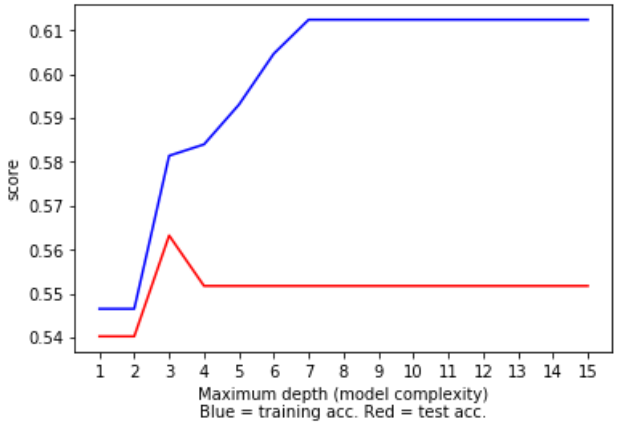
```
import matplotlib.pyplot as plt

train_result=result_set['split0_train_score']
test_result=result_set['split0_test_score']
print("Total number of models: ", len(test_result))
# plot max depth hyperparameter values vs training and test accuracy score
plt.plot(range(0,len(train_result)),train_result,'b',range(0,len(test_result)),test_result,'r')
plt.xlabel('Combination of hyperparameters\nBlue = training acc. Red = test acc.')
plt.ylabel('score')
plt.show()
```

```
Total number of models:  120
```



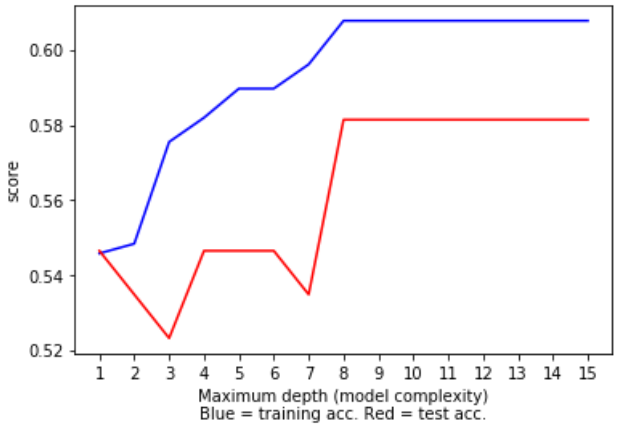
In [24]:

```
dd = pd.DataFrame(result_set['params'])

index_ = list(dd.index[(dd['criterion']=='entropy') & (dd['min_samples_leaf']==10)])
max_depth_train = []
max_depth_test = []
for i in range(0,len(index_)):
    max_depth_train.append(train_result[index_[i]])
    max_depth_test.append(test_result[index_[i]])
plt.plot(range(1, len(max_depth_train)+1), max_depth_train, 'b', range(1,len(max_depth_test)+1), max_depth_test, 'r')
plt.xlabel('Maximum depth (model complexity)\nBlue = training acc. Red = test acc.')
plt.ylabel('score')
plt.xticks(np.arange(1, len(max_depth_train)+1, 1))
plt.show()
```



In [25]:

```
train_result = result_set['split1_train_score']
test_result = result_set['split1_test_score']
max_depth_train = []
max_depth_test = []
for i in range(0,len(index_)):
    max_depth_train.append(train_result[index_[i]])
    max_depth_test.append(test_result[index_[i]])
plt.plot(range(1, len(max_depth_train)+1), max_depth_train, 'b', range(1,len(max_depth_test)+1), max_depth_test, 'r')
plt.xlabel('Maximum depth (model complexity)\nBlue = training acc. Red = test acc.')
plt.ylabel('score')
plt.xticks(np.arange(1, len(max_depth_train)+1, 1))
plt.show()
```



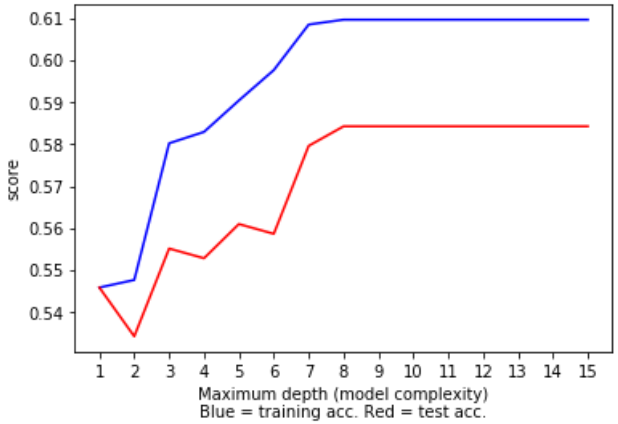
In [26]:

```
print(cv_1.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 5}
```

In [27]:

```
train_result = result_set['mean_train_score']
test_result = result_set['mean_test_score']

max_depth_train = []
max_depth_test = []
index_
for i in range(len(index_)):
    max_depth_train.append(train_result[index_[i]])
    max_depth_test.append(test_result[index_[i]])
plt.plot(range(1, len(max_depth_train)+1), max_depth_train, 'b', range(1,len(max_depth_test)+1), max_depth_test, 'r')
plt.xlabel('Maximum depth (model complexity)\nBlue = training acc. Red = test acc.')
plt.xticks(np.arange(1, len(max_depth_train)+1, 1))
plt.ylabel('score')
plt.show()
```



In [28]:

```
cv_1.fit(sample_fixed_train, target_train)

print("Train accuracy:", cv_1.score(sample_fixed_train, target_train))
print("Test accuracy:", cv_1.score(sample_fixed_test, target_test))
```

```
Train accuracy: 0.6225319396051103
Test accuracy: 0.5826558265582655
```

In [29]:

```
# inside `dm_tools.py' together with data_prep()
import numpy as np
import pydot
from io import StringIO
from sklearn.tree import export_graphviz

def analyse_feature_importance(dm_model, feature_names, n_to_display=20):
    # grab feature importances from the model
    importances = dm_model.feature_importances_

    # sort them out in descending order
    indices = np.argsort(importances)
    indices = np.flip(indices, axis=0)

    # limit to 20 features, you can leave this out to print out everything
    indices = indices[:n_to_display]

    for i in indices:
        print(feature_names[i], ':', importances[i])

def visualize_decision_tree(dm_model, feature_names, save_name):
    dotfile = StringIO()
    export_graphviz(dm_model, out_file=dotfile, feature_names=feature_names)
    graph = pydot.graph_from_dot_data(dotfile.getvalue())
    graph[0].write_png(save_name) # saved in the following file
```

In [ ]:

```

```

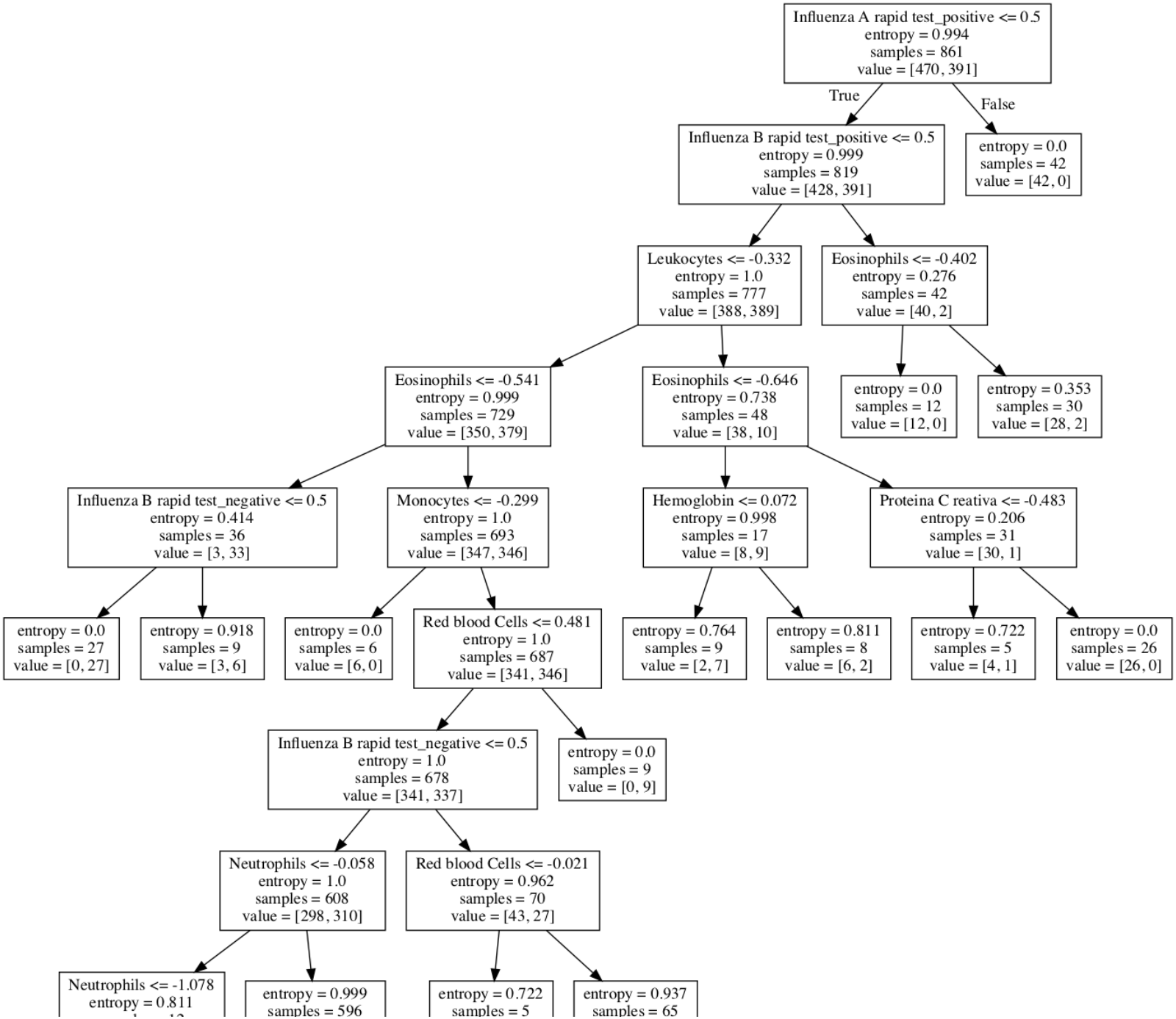
In [30]:

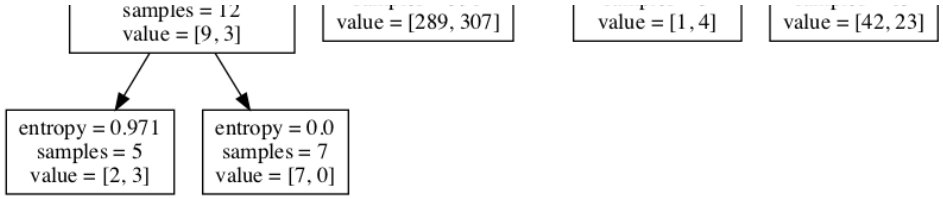
```
# do the feature importance and visualization analysis on GridSearchCV
#from dm_tools import analyse_feature_importance, visualize_decision_tree

#analyse_feature_importance(cv_1.best_estimator_, sample_fixed.columns, 20)
#visualize_decision_tree(cv_1.best_estimator_, sample_fixed.columns, "optimal_tree.png")
```

In [31]:

```
Image(filename="optimal_tree.png",width=1000,height=1000)
```

Out[31]:


Influenza A rapid test_positive <= 0.5
entropy = 0.994
samples = 861
value = [470, 391]
True
False
Influenza B rapid test_positive <= 0.5
entropy = 0.999
samples = 819
value = [428, 391]
entropy = 0.0
samples = 42
value = [42, 0]
Leukocytes <= -0.332
entropy = 1.0
samples = 777
value = [388, 389]
Eosinophils <= -0.402
entropy = 0.276
samples = 42
value = [40, 2]
Eosinophils <= -0.541
entropy = 0.999
samples = 729
value = [350, 379]
Eosinophils <= -0.646
entropy = 0.738
samples = 48
value = [38, 10]
entropy = 0.0
samples = 12
value = [12, 0]
entropy = 0.353
samples = 30
value = [28, 2]
Influenza B rapid test_negative <= 0.5
entropy = 0.414
samples = 36
value = [3, 33]
Monocytes <= -0.299
entropy = 1.0
samples = 693
value = [347, 346]
Hemoglobin <= 0.072
entropy = 0.998
samples = 17
value = [8, 9]
Proteina C reativa <= -0.483
entropy = 0.206
samples = 31
value = [30, 1]
entropy = 0.0
samples = 27
value = [0, 27]
entropy = 0.918
samples = 9
value = [3, 6]
entropy = 0.0
samples = 6
value = [6, 0]
Red blood Cells <= 0.481
entropy = 1.0
samples = 687
value = [341, 346]
entropy = 0.764
samples = 9
value = [2, 7]
entropy = 0.811
samples = 8
value = [6, 2]
entropy = 0.722
samples = 5
value = [4, 1]
entropy = 0.0
samples = 26
value = [26, 0]
Influenza B rapid test_negative <= 0.5
entropy = 1.0
samples = 678
value = [341, 337]
entropy = 0.0
samples = 9
value = [0, 9]
Neutrophils <= -0.058
entropy = 1.0
samples = 608
value = [298, 310]
Red blood Cells <= -0.021
entropy = 0.962
samples = 70
value = [43, 27]
Neutrophils <= -1.078
entropy = 0.811
entropy = 0.999
samples = 596
entropy = 0.722
samples = 5
entropy = 0.937
samples = 65



In [32]:

```
target_pred_dt = model.predict(sample_fixed_test)
target_pred_dt_small = model_small.predict(sample_fixed_test)
target_pred_dt_cv = cv_1.predict(sample_fixed_test)

print("Accuracy score on test for DT_default:", accuracy_score(target_test, target_pred_dt))
print("Accuracy score on test for DT_small:", accuracy_score(target_test, target_pred_dt_small))
print("Accuracy score on test for DT_optimal", accuracy_score(target_test, target_pred_dt_cv))
```

```
Accuracy score on test for DT_default: 0.5880758807588076
Accuracy score on test for DT_small: 0.5663956639566395
Accuracy score on test for DT_optimal 0.5826558265582655
```

In [33]:

```
dt_cv_best = cv_1.best_estimator_
# probability prediction from decision tree
target_pred_proba_dt = dt_cv_best.predict_proba(sample_fixed_test)

print("Probability produced by decision tree for each class vs actual prediction on TargetB (0 = non-donor, 1 = donor). You should b
e able to see the default threshold of 0.5.")
print("(Probs on zero)\t(probs on one)\t(prediction made)")
# print top 10
for i in range(20):
    print(target_pred_proba_dt[i][0], '\t', target_pred_proba_dt[i][1], '\t', target_pred[i])
```

```
Probability produced by decision tree for each class vs actual prediction on TargetB (0 = non-donor, 1 = donor). You sho
uld be able to see the default threshold of 0.5.
(Probs on zero) (probs on one)  (prediction made)
0.9333333333333333      0.06666666666666667     0
0.4848993288590604      0.5151006711409396      0
0.6461538461538462      0.35384615384615387     0
0.4848993288590604      0.5151006711409396      0
0.9333333333333333      0.06666666666666667     0
0.4848993288590604      0.5151006711409396      0
0.4848993288590604      0.5151006711409396      0
0.4848993288590604      0.5151006711409396      0
0.0     1.0     1
0.0     1.0     1
0.4848993288590604      0.5151006711409396      0
0.4848993288590604      0.5151006711409396      1
1.0     0.0     0
0.4848993288590604      0.5151006711409396      0
1.0     0.0     0
0.4848993288590604      0.5151006711409396      0
0.4848993288590604      0.5151006711409396      0
0.4848993288590604      0.5151006711409396      0
0.4848993288590604      0.5151006711409396      0
```

In [34]:

```
from sklearn.metrics import roc_auc_score

target_pred_proba_dt = model.predict_proba(sample_fixed_test)
target_pred_proba_dt_small = model_small.predict_proba(sample_fixed_test)
target_pred_proba_dt_cv = dt_cv_best.predict_proba(sample_fixed_test)

roc_index_dt = roc_auc_score(target_test, target_pred_proba_dt[:, 1])
roc_index_dt_small = roc_auc_score(target_test, target_pred_proba_dt_small[:, 1])
roc_index_dt_cv = roc_auc_score(target_test, target_pred_proba_dt_cv[:, 1])

print("ROC index on test for DT_default:", roc_index_dt)
print("ROC index on test for DT_small:", roc_index_dt_small)
print("ROC index on test for DT_optimal:", roc_index_dt_cv)
```

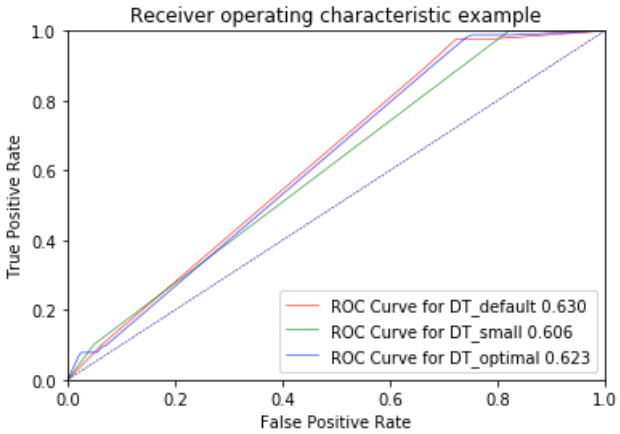
```
ROC index on test for DT_default: 0.6296762909823916
ROC index on test for DT_small: 0.6061836722594416
ROC index on test for DT_optimal: 0.6233177209936562
```

In [35]:

```
from sklearn.metrics import roc_curve


fpr_dt, tpr_dt, thresholds_dt = roc_curve(target_test, target_pred_proba_dt[:,1])
fpr_dt_small, tpr_dt_small, thresholds_dt_small = roc_curve(target_test, target_pred_proba_dt_small[:,1])
fpr_dt_cv, tpr_dt_cv, thresholds_dt_cv = roc_curve(target_test, target_pred_proba_dt_cv[:,1])
```

In [36]:

```
import matplotlib.pyplot as plt

plt.plot(fpr_dt, tpr_dt, label='ROC Curve for DT_default {:.3f}'.format(roc_index_dt), color='red', lw=0.5)
plt.plot(fpr_dt_small, tpr_dt_small, label='ROC Curve for DT_small {:.3f}'.format(roc_index_dt_small), color='green', lw=0.5)
plt.plot(fpr_dt_cv, tpr_dt_cv, label='ROC Curve for DT_optimal {:.3f}'.format(roc_index_dt_cv), color='blue', lw=0.5)

# plt.plot(fpr[2], tpr[2], color='darkorange',
#          lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
plt.plot([0, 1], [0, 1], color='navy', lw=0.5, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



In [37]:

```
import pickle
dt_best = model
with open('DT.pickle', 'wb') as f:
    pickle.dump([dt_best,roc_index_dt, fpr_dt, tpr_dt], f)
```

In [ ]:

```

```