S4 - HOPE random forest-V2

September 16, 2020

1 Import data from DB.

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
dfOrg = pd.read_csv('hope_dataset_cleaned.csv')

print(dfOrg.shape[0])
```

```
1243
```

[3]:
```
dfOrg.head(10)
```

[3]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	pedido.data.attributes.gender	articulo	respuesta.articlesRevisedYear
0	75.0	FISTULA PERITONEAL	male	27395425	2018
1	75.0	FISTULA PERITONEAL	male	28560554	2018
2	75.0	FISTULA PERITONEAL	male	28641726	2017
3	75.0	FISTULA PERITONEAL	male	26245344	2016
4	75.0	FISTULA PERITONEAL	male	28942543	2018
5	75.0	FISTULA PERITONEAL	male	24782153	2014
6	75.0	FISTULA PERITONEAL	male	28002229	2018
7	75.0	FISTULA PERITONEAL	male	27505109	2017
8	75.0	FISTULA PERITONEAL	male	24850546	2015
9	75.0	FISTULA PERITONEAL	male	29371050	2019

```
   respuesta.articlesRevisedMonth  \
0                               1
1                               4
2                              12
3                              12
4                               6
5                               6
6                               9
7                               4
8                               1
9                               4

                              respuesta.pubmed_keys  utilidad
0  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       1.0
1  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
2  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
3  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
4  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
5  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
6  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
7  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
8  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
9  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
```

Expand pubmed_keys attribute

[4]:
```
dfOrg['respuesta.pubmed_keys'] = dfOrg['respuesta.pubmed_keys'].apply(lambda x :
 ↪ str(x).split(','))

dfOrg = dfOrg.explode('respuesta.pubmed_keys').reset_index(drop=True)

dfOrg.head(10)
```

[4]:
```
   pedido.data.attributes.age pedido.data.attributes.diagnostic_main  \
0                        75.0                     FISTULA PERITONEAL
1                        75.0                     FISTULA PERITONEAL
2                        75.0                     FISTULA PERITONEAL
3                        75.0                     FISTULA PERITONEAL
4                        75.0                     FISTULA PERITONEAL
5                        75.0                     FISTULA PERITONEAL
6                        75.0                     FISTULA PERITONEAL
7                        75.0                     FISTULA PERITONEAL
8                        75.0                     FISTULA PERITONEAL
9                        75.0                     FISTULA PERITONEAL

  pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                          male  27395425                           2018
```

```
1                          male  27395425                          2018
2                          male  27395425                          2018
3                          male  27395425                          2018
4                          male  27395425                          2018
5                          male  27395425                          2018
6                          male  27395425                          2018
7                          male  27395425                          2018
8                          male  27395425                          2018
9                          male  27395425                          2018

   respuesta.articlesRevisedMonth respuesta.pubmed_keys  utilidad
0                               1                Abdomen       1.0
1                               1         Adenocarcinoma       1.0
2                               1            Antiemetics       1.0
3                               1          Blood Culture       1.0
4                               1              Catharsis       1.0
5                               1               Diuresis       1.0
6                               1                Fistula       1.0
7                               1            Gastrectomy       1.0
8                               1      Incisional Hernia       1.0
9                               1             Intestines       1.0
```

2 Transform (factorice) from Categories to continuous atributes

Transform 'pedido.data.attributes.diagnostic_main' atribute

[5]:
```
dataDiagnosticMain, categoriesDiagnosticMain = pd.factorize(dfOrg['pedido.data.
↪attributes.diagnostic_main'])

dfOrg['pedido.data.attributes.diagnostic_main'] = dataDiagnosticMain
```

Transform 'gender' atribute

[6]:
```
dataGender, categoriesGender = pd.factorize(dfOrg['pedido.data.attributes.
↪gender'])

dfOrg['pedido.data.attributes.gender'] = dataGender
```

Transform 'respuesta.pubmed_keys' atribute

[7]:
```
categoriesORGPubMedKeys = dfOrg['respuesta.pubmed_keys'].value_counts()

print("total: " + str(categoriesORGPubMedKeys.size))
```

```
total: 353
```

[8]:
```
dataPubMedKeys, categoriesPubMedKeys = pd.factorize(dfOrg['respuesta.
↪pubmed_keys'])

dfOrg['respuesta.pubmed_keys'] = dataPubMedKeys
```

[9]:
```
dfOrg.head(10)
```

[9]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	\
0	75.0	0	
1	75.0	0	
2	75.0	0	
3	75.0	0	
4	75.0	0	
5	75.0	0	
6	75.0	0	
7	75.0	0	
8	75.0	0	
9	75.0	0	

	pedido.data.attributes.gender	articulo	respuesta.articlesRevisedYear	\
0	0	27395425	2018	
1	0	27395425	2018	
2	0	27395425	2018	
3	0	27395425	2018	
4	0	27395425	2018	
5	0	27395425	2018	
6	0	27395425	2018	
7	0	27395425	2018	
8	0	27395425	2018	
9	0	27395425	2018	

	respuesta.articlesRevisedMonth	respuesta.pubmed_keys	utilidad
0	1	0	1.0
1	1	1	1.0
2	1	2	1.0
3	1	3	1.0
4	1	4	1.0
5	1	5	1.0
6	1	6	1.0
7	1	7	1.0
8	1	8	1.0
9	1	9	1.0

[10]:
```
print("age NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.attributes.age'])].
↪shape[0]))
print("diagnostic_main NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.
↪attributes.diagnostic_main'])].shape[0]))
```

```
print("gender NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.attributes.
 ↪gender'])].shape[0]))
print("articulo NaN => " + str(dfOrg[pd.isnull(dfOrg['articulo'])].shape[0]))
print("articlesRevisedYear NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
 ↪articlesRevisedYear'])].shape[0]))
print("articlesRevisedMonth NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
 ↪articlesRevisedMonth'])].shape[0]))
print("pubmed_keys NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
 ↪pubmed_keys'])].shape[0]))
print("utilidad NaN => " + str(dfOrg[pd.isnull(dfOrg['utilidad'])].shape[0]))
```

```
age NaN => 10
diagnostic_main NaN => 0
gender NaN => 0
articulo NaN => 0
articlesRevisedYear NaN => 0
articlesRevisedMonth NaN => 0
pubmed_keys NaN => 0
utilidad NaN => 14758
```

Remove row with age eq NaN

[11]:
```
dfOrg = dfOrg[pd.notnull(dfOrg['pedido.data.attributes.age'])]
```

3 Standardize the Data

Choosed “age”, “diagnostic_main”, “month” and “pubmed_keys” attributes (based on PCA_V3 study)

[12]:
```
from sklearn.preprocessing import StandardScaler

features = ["pedido.data.attributes.age",
    "pedido.data.attributes.diagnostic_main",
    "respuesta.articlesRevisedMonth",
    "respuesta.pubmed_keys",
    "utilidad"
]

# Separating out the features
x = dfOrg.loc[:, features].values

featuresTransformed = StandardScaler().fit_transform(x)

dfStandarized = pd.DataFrame(featuresTransformed, index=dfOrg.index,␣
 ↪columns=features)
dfStandarized['utilidad'] = dfOrg['utilidad']
```

```
dfStandarized
```

[12]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	\
0	1.285887	-1.503163	
1	1.285887	-1.503163	
2	1.285887	-1.503163	
3	1.285887	-1.503163	
4	1.285887	-1.503163	
…	…	…	
15583	-0.607930	-0.586347	
15584	-0.607930	-0.586347	
15585	-0.607930	-0.586347	
15586	-0.607930	-0.586347	
15587	-0.607930	-0.586347	

	respuesta.articlesRevisedMonth	respuesta.pubmed_keys	utilidad
0	-1.463658	-1.089722	1.0
1	-1.463658	-1.080463	1.0
2	-1.463658	-1.071203	1.0
3	-1.463658	-1.061944	1.0
4	-1.463658	-1.052684	1.0
…	…	…	…
15583	-1.178433	-0.330441	NaN
15584	-1.178433	-0.978608	NaN
15585	-1.178433	0.891817	NaN
15586	-1.178433	-0.876753	NaN
15587	-1.178433	0.901077	NaN

[15578 rows x 5 columns]

4 Separe data by utilidad is defined

[13]:
```
dfDataSetComplete = dfStandarized[pd.notnull(dfStandarized['utilidad'])]

print(dfDataSetComplete.shape[0])

dfDataSetToPredict = dfStandarized[pd.isnull(dfStandarized['utilidad'])]

print(dfDataSetToPredict.shape[0])
```

```
830
14748
```

5 Random Forest

We check the number of results

[14]:
```
dfDataSetComplete.groupby('utilidad').size()
```

[14]:
```
utilidad
0.0    346
1.0    484
dtype: int64
```

Separe “utilidad” atribute from dataToTrain

[15]:
```
X = np.array(dfDataSetComplete.drop(['utilidad'],1))
y = np.array(dfDataSetComplete['utilidad'])
X.shape
```

[15]:
```
(830, 4)
```

[16]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

6 Exploring number of estimators

Via the sample size n of the bootstrap sample, we control the bias-variance tradeoff of the random forest. By choosing a larger value for n, we decrease the randomness and thus the forest is more likely to overfit. On the other hand, we can reduce the degree of overfitting by choosing smaller values for n at the expense of the model performance. In most implementations, including the RandomForestClassifier implementation in scikit-learn, the sample size of the bootstrap sample is chosen to be equal to the number of samples in the original training set, which usually provides a good bias-variance tradeoff.

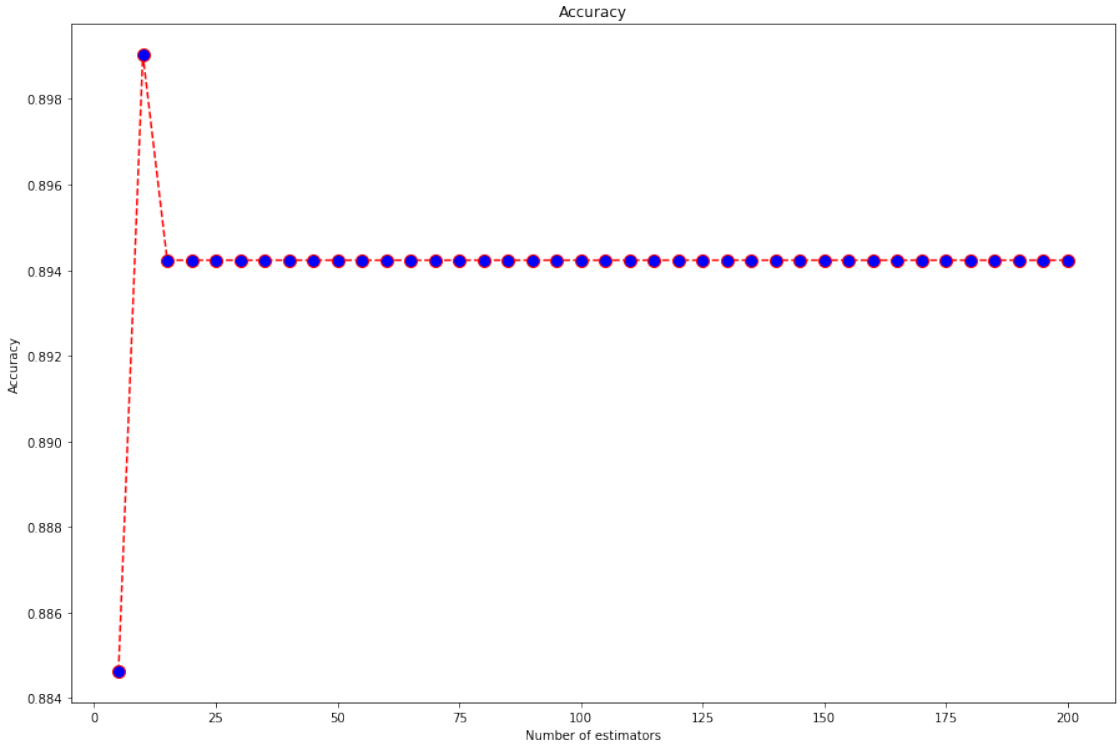
https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb

[17]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score

k_range = range(5, 205, 5)
accuracy = []

for k in k_range:
    forest_test = RandomForestClassifier(
        criterion='entropy',
        n_estimators=k,
        random_state=0
```

```
    )
    forest_test.fit(X_train, y_train)
    y_pred_test = forest_test.predict(X_test)

    accuracy.append(accuracy_score(y_test, y_pred_test))
```

[18]:
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(figsize=(15, 10))
axs.plot(k_range, accuracy, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
axs.set_title('Accuracy')
axs.set_xlabel('Number of estimators')
axs.set_ylabel('Accuracy')
```

[18]:
```
Text(0, 0.5, 'Accuracy')
```



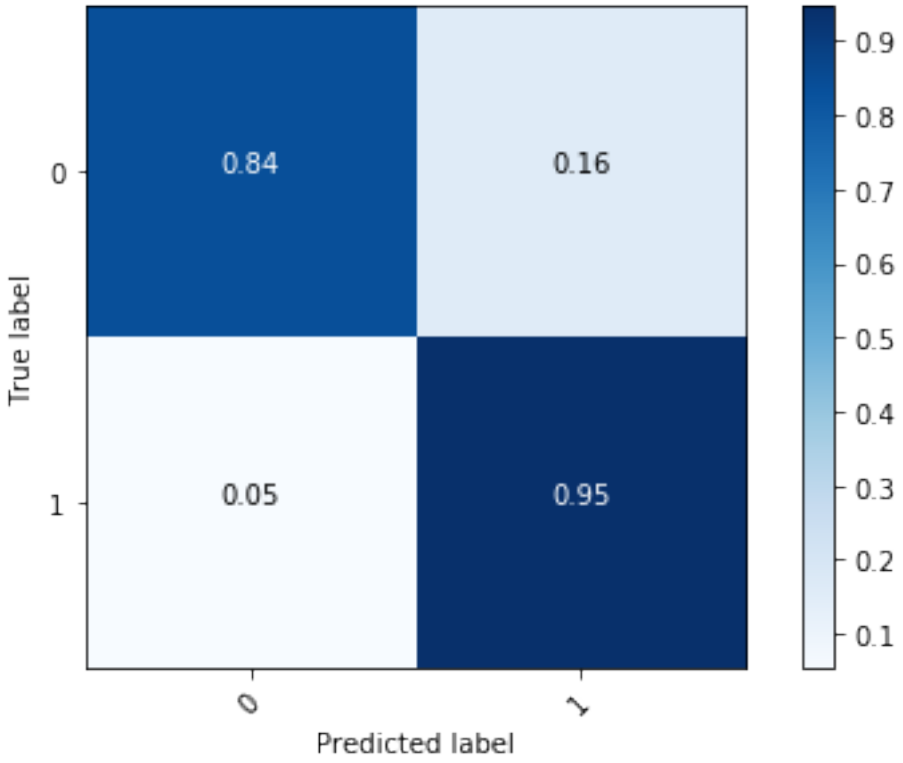
6.1 Evaluating the Algorithm

[19]:
```
forest = RandomForestClassifier(
    criterion='entropy',
    n_estimators=10,
    random_state=0
)

forest.fit(X_train, y_train)

y_pred = forest.predict(X_test)

print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.84      0.88        95
         1.0       0.88      0.95      0.91       113

    accuracy                           0.90       208
   macro avg       0.90      0.89      0.90       208
weighted avg       0.90      0.90      0.90       208

0.8990384615384616
```

[20]:
```
import itertools


cnf_matrix = confusion_matrix(y_test, y_pred)

def plot_confusion_matrix(cm, classes):
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cmap=plt.cm.Blues

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], ".2f"),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```

```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

n_classes=["0","1"]
plot_confusion_matrix(cnf_matrix, classes=n_classes)
```



7 Run Prediction

[21]:
```
result = forest.predict(dfDataSetToPredict[["pedido.data.attributes.age",
    "pedido.data.attributes.diagnostic_main",
    "respuesta.articlesRevisedMonth",
    "respuesta.pubmed_keys"
]])

result
```

[21]:
```
array([1., 1., 1., …, 0., 0., 0.])
```