Emma Mayes (eemayes2)

IE517 MLF F21

Module 6 Homework (Cross validation)

Using the ccdefault dataset, with 90% for training and 10% for test (stratified sampling) and the decision tree model that you did in Module 2:

Part 1: Random test train splits

Run in-sample and out-of-sample accuracy scores for 10 different samples by changing random_state from 1 to 10 in sequence.

Display the individual scores, then calculate the mean and standard deviation on the set of scores. Report in a table format.

Part 2: Cross validation

Now rerun your model using cross_val_scores with stratified k-fold CV (k=10).

Report the individual fold accuracy scores, the mean CV score and the standard deviation of the fold scores. Now run the out-of-sample accuracy score. Report in a table format.

Part 3: Conclusions

Write a short paragraph summarizing your findings. Which method of measuring accuracy provides the best estimate of how a model will do against unseen data? Which one is more efficient to run?

The out-of-sample accuracy score provides the best estimate of how a model will do against unseen data. Since the data used to find this accuracy score is unseen by the model since it is not in the training sample, it provides the best estimate. Method-wise, the K-fold method does the best by making sure sampling is done without-replacement. K-fold is also the most efficient method of doing this, as opposed to writing and running multiple for-loops, the K-fold does this more effectively under the hood.

Part 4: Appendix

Link to github repo: https://github.com/eemayes2/IE517_F21_HW6

IE517_HWK6

October 1, 2021

[1]:
```
#Import libraries needed
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

#Check if any null values we need to change
def num_missing(x):
    return sum(x.isnull())
```

[5]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

[6]:
```
#Read in Data
df = pd.read_csv('gdrive/MyDrive/Colab Notebooks/ccdefault.csv', header=0)
df.head()
```

[6]:

	ID	LIMIT_BAL	SEX	EDUCATION	...	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULT
0	1	20000	2	2	...	0	0	0	1
1	2	120000	2	2	...	1000	0	2000	1
2	3	90000	2	2	...	1000	1000	5000	0
3	4	50000	2	2	...	1100	1069	1000	0
4	5	50000	1	2	...	9000	689	679	0

[5 rows x 25 columns]

[7]:
```
df.describe()
```

[7]:

	ID	LIMIT_BAL	...	PAY_AMT6	DEFAULT
count	30000.000000	30000.000000	...	30000.000000	30000.000000
mean	15000.500000	167484.322667	...	5215.502567	0.221200
std	8660.398374	129747.661567	...	17777.465775	0.415062
min	1.000000	10000.000000	...	0.000000	0.000000
25%	7500.750000	50000.000000	...	117.750000	0.000000
50%	15000.500000	140000.000000	...	1500.000000	0.000000

```
75%     22500.250000   240000.000000  ...     4000.000000       0.000000
max     30000.000000  1000000.000000  ...   528666.000000       1.000000

[8 rows x 25 columns]
```

0.0.1 Try different random states in test-train-split

[13]:
```
#Split into training-test sets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
x = df.drop(columns = ['DEFAULT'])
y = df['DEFAULT']
out_of = []
in_of = []
for i in range(1,11):
  X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,␣
 ↪random_state=i, stratify = y)
  tree = DecisionTreeClassifier(criterion = 'gini', max_depth = 4, random_state␣
 ↪= 1)
  tree.fit(X_train, y_train)
  y_pred = tree.predict(X_test)
  acc_out = accuracy_score(y_test, y_pred)
  out_of.append(acc_out)
  y_pred_train = tree.predict(X_train)
  acc_in = accuracy_score(y_train, y_pred_train)
  in_of.append(acc_in)
  print("Seed Number:" + str(i) + "\n\tOut of Sample Accuracy: " + str(acc_out))
  print("\tIn Sample Accuracy: " + str(acc_in))
```

```
Seed Number:1
        Out of Sample Accuracy: 0.8283333333333334
        In Sample Accuracy: 0.8225555555555556
Seed Number:2
        Out of Sample Accuracy: 0.824
        In Sample Accuracy: 0.8225925925925925
Seed Number:3
        Out of Sample Accuracy: 0.8173333333333334
        In Sample Accuracy: 0.8241111111111111
Seed Number:4
        Out of Sample Accuracy: 0.8196666666666667
        In Sample Accuracy: 0.8237037037037037
Seed Number:5
        Out of Sample Accuracy: 0.818
        In Sample Accuracy: 0.8231481481481482
Seed Number:6
        Out of Sample Accuracy: 0.819
        In Sample Accuracy: 0.823925925925926
```

```
Seed Number:7
        Out of Sample Accuracy: 0.8253333333333334
        In Sample Accuracy: 0.8232592592592592
Seed Number:8
        Out of Sample Accuracy: 0.8166666666666667
        In Sample Accuracy: 0.8238148148148148
Seed Number:9
        Out of Sample Accuracy: 0.8173333333333334
        In Sample Accuracy: 0.823925925925926
Seed Number:10
        Out of Sample Accuracy: 0.82
        In Sample Accuracy: 0.8236296296296296
```

[20]:
```
print("Out of Sample: \n\tMean: " + str(np.mean(out_of)) + "\n\tStand. Dev: " +␣
 ↪str(np.std(out_of)))
print("In Sample: \n\tMean: " + str(np.mean(in_of)) + "\n\tStand. Dev: " +␣
 ↪str(np.std(in_of)))
```

```
Out of Sample:
        Mean: 0.8205666666666666
        Stand. Dev: 0.0037566237797019685
In Sample:
        Mean: 0.8234666666666666
        Stand. Dev: 0.0005272284340525735
```

0.0.2 Cross Validation with K-fold CV

[40]:
```
from sklearn.model_selection import StratifiedKFold, cross_val_score
x = df.drop(columns = ['DEFAULT'])
y = df['DEFAULT']

cv = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
scores = cross_val_score(tree, x, y, scoring='accuracy', cv=cv)

scores
```

[40]:
```
array([0.819     , 0.827     , 0.82366667, 0.82466667, 0.82033333,
       0.817     , 0.81233333, 0.81833333, 0.81366667, 0.82566667])
```

[35]:
```
print("Cross Val Scores: \n\tMean: " + str(np.mean(scores)) + "\n\tStand. Dev:␣
 ↪" + str(np.std(scores)))
```

```
Cross Val Scores:
        Mean: 0.8207333333333333
        Stand. Dev: 0.006848844184726854
```

[41]:
```
#Out of sample accuracy
kf = StratifiedKFold(n_splits=10, random_state = 1, shuffle = True)
```

```
kf.get_n_splits(x,y)
i = 1
out_of = []

for train_index, test_index in kf.split(x):
  X_train, X_test = x.iloc[list(train_index)], x.iloc[list(test_index)]
  y_train, y_test = y.iloc[list(train_index)], y.iloc[list(test_index)]
  tree.fit(X_train, y_train)
  y_pred = tree.predict(X_test)
  acc_out = accuracy_score(y_test, y_pred)
  out_of.append(acc_out)
  y_pred_train = tree.predict(X_train)
  print("Fold Index:" + str(i) + "\n\tOut of Sample Accuracy: " + str(acc_out))
  print("\tIn Sample Accuracy: " + str(acc_in))
  i += 1
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-41-63251d908dc7> in <module>()
         5 out_of = []
         6
   ----> 7 for train_index, test_index in kf.split(x):
         8   X_train, X_test = x.iloc[list(train_index)], x.
↪iloc[list(test_index)]
         9   y_train, y_test = y.iloc[list(train_index)], y.
↪iloc[list(test_index)]


       TypeError: split() missing 1 required positional argument: 'y'
```

[38]:
```
print("Out of Sample: \n\tMean: " + str(np.mean(out_of)) + "\n\tStand. Dev: " +␣
 ↪str(np.std(out_of)))
```

```
Out of Sample:
       Mean: 0.8207333333333333
       Stand. Dev: 0.006848844184726854
```

[39]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('IE517_HWK6.ipynb')
```

```
--2021-10-01 16:48:11--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.110.133, 185.199.109.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-10-01 16:48:11 (27.8 MB/s) - colab_pdf.py saved [1864/1864]

Mounted at /content/drive/

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab
Notebooks/IE517_HWK6.ipynb to pdf
[NbConvertApp] Writing 38290 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: [u'xelatex', u'./notebook.tex',
'-quiet']
[NbConvertApp] Running bibtex 1 time: [u'bibtex', u'./notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 35185 bytes to /content/drive/My Drive/IE517_HWK6.pdf

<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>
```

[39]: 'File ready to be Downloaded and Saved to Drive'