### Offshore distribution of yelkouan shearwaters in the north-western Adriatic Sea: insight from machine learning
### Authors: Silvia Bonizzoni, Blair D. Sterba-Boatwright, Sarah Piwetz, Giovanni Bearzi

Prep

python imports

```
import numpy as np
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
pip install interpret
from interpret.glassbox import ExplainableBoostingClassifier
from interpret import show
```

Import data

```
df_sw = pd.read_csv("myNewDataForInterpret.csv")
X_sw = df_sw.drop(columns = ['presence'])
y_sw = df_sw['presence']
```

Models

Model on full data

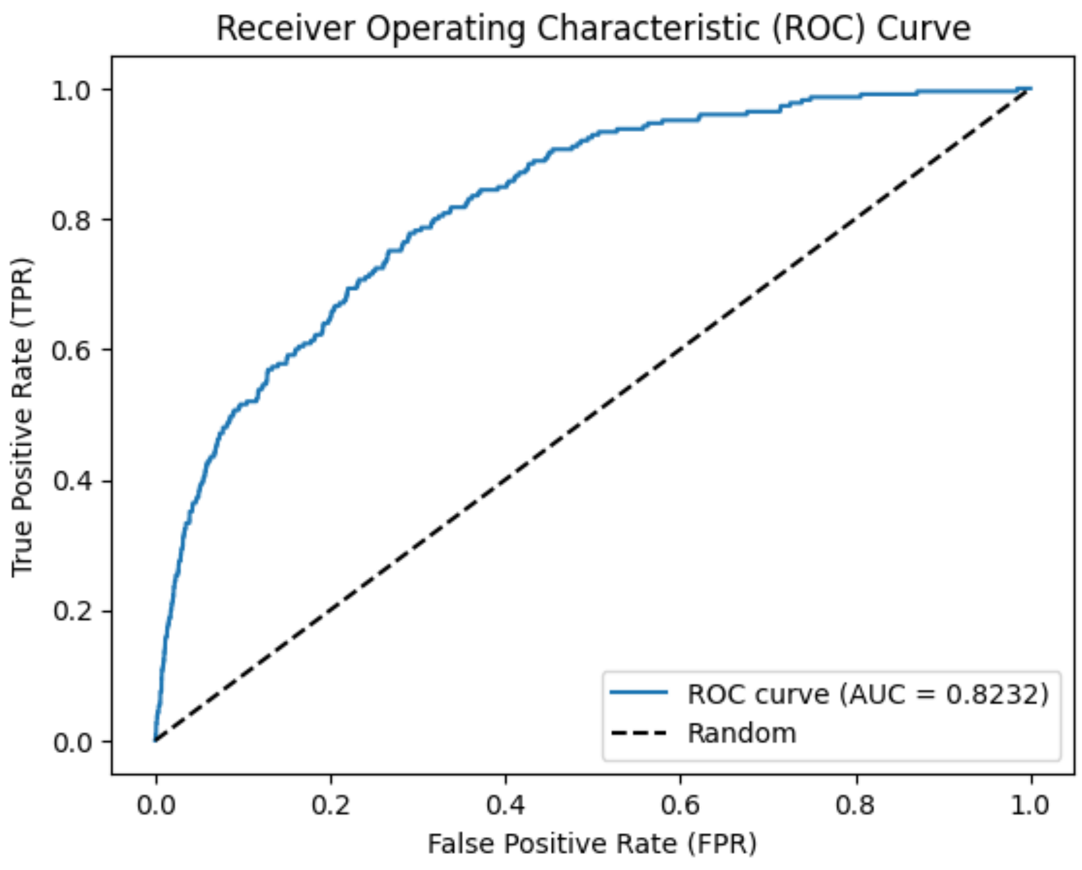
```
ebm_sw = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw.fit(X_sw, y_sw)
ExplainableBoostingClassifier(inner_bags=25, outer_bags=25)
show(ebm_sw.explain_global())
auc = roc_auc_score(y_sw, ebm_sw.predict_proba(X_sw)[:, 1])
print("AUC: {:.3f}".format(auc))
AUC: 0.908
```

AUC is 0.908

Models using five-fold cross-validation

```
seed = 20240103

k_fold_predictions = np.arange(0, 33227, 1.0) # just to create a vector of the correct length
pyando = np.arange(33227) % 5 # 0 to 653 mod 5; i.e., 0's through 4's
pyando = pyando.tolist()
random.shuffle(pyando) # shuffles pyando
for i in np.arange(5):
  train_set = [True if x != i else False for x in pyando]
  test_set = [True if x == i else False for x in pyando]
  X_sw_i = X_sw[train_set]
  y_sw_i = y_sw[train_set]
  X_sw_test_i = X_sw[test_set]
  y_sw_test_i = y_sw[test_set]
  ebm_sw_i = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
  ebm_sw_i.fit(X_sw_i, y_sw_i)
  k_fold_predictions[test_set] = ebm_sw_i.predict_proba(X_sw_test_i)[:, 1]
```

```
y_true_k = y_sw
y_scores_k = k_fold_predictions
fpr_k, tpr_k, thresholds_k = roc_curve(y_true_k, y_scores_k)

# Calculate the area under the ROC curve (AUC)
auc_k = roc_auc_score(y_true_k, y_scores_k)

# Plot the ROC curve
plt.plot(fpr_k, tpr_k, label='ROC curve (AUC = {:.4f})'.format(auc_k))
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```



Runs 1-10, AUCs are 0.8158, 0.8227, 0.8182, 0.8266, 0.8172, 0.8167, 0.8226, 0.8101, 0.8141, 0.8232
mean is 0.819, sd is 0.005

```
# Export predicted values for further analysis
df = pd.DataFrame([y_sw, k_fold_predictions])
df = df.T
df.to_csv('modelOutput.csv')
```

What happens if random variables are included?

Data for these tests was generated externally using R.

One uniform random variable

```
df_honk = pd.read_csv("dataPlusRandom/greatHonkU1_5.csv")
X_honk = df_honk.drop(columns = ['presence'])
y_honk = df_honk['presence']
ebm_honk = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_honk.fit(X_honk, y_honk)
show(ebm_honk.explain_global())
```

test 1: unif comes after all other variables in importance

tests 2-5: same, or even after the first interaction term

One normal random variable

```
df_honk = pd.read_csv("dataPlusRandom/greatHonkN1_5.csv")
X_honk = df_honk.drop(columns = ['presence'])
y_honk = df_honk['presence']
ebm_honk = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_honk.fit(X_honk, y_honk)
show(ebm_honk.explain_global())
```

test 1: norm comes after all other variables in importance

tests 2-5: same, or even after the first interaction term

Five uniform random variables

```
df_honk = pd.read_csv("dataPlusRandom/greatHonkU5_5.csv")
X_honk = df_honk.drop(columns = ['presence'])
y_honk = df_honk['presence']
ebm_honk = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_honk.fit(X_honk, y_honk)
show(ebm_honk.explain_global())
```

test 1: all uniform variables fall below all other main variables, some below interactions

tests 2-5: same

Five normal random variables

```
df_honk = pd.read_csv("dataPlusRandom/greatHonkN5_5.csv")
X_honk = df_honk.drop(columns = ['presence'])
y_honk = df_honk['presence']
ebm_honk = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_honk.fit(X_honk, y_honk)
show(ebm_honk.explain_global())
```

test 1: all normal variables fall below all other main variables, some below interactions

tests 2-5: same
NB, dist X sst interaction is the only one that commonly (but not universally) falls above random

Export plots for each predictor for improvement (?) in R

In retrospect, I'm not sure this is any better than just using the plots from the entire dataset, but here's the strategy: partition the dataset into five pieces, and generate and export the plots of each variable for each piece. Then average the plots together in R and add a smoother.

```
pyando = np.arange(33227) % 5 # 0 to 653 mod 5; i.e., 0's through 4's
pyando = pyando.tolist()
random.shuffle(pyando) # shuffles pyando
```

```
# Run the following for each i from 0 to 4:
i = 0
train_set = [True if x != i else False for x in pyando]
test_set = [True if x == i else False for x in pyando]
X_sw_i = X_sw[train_set]
y_sw_i = y_sw[train_set]
X_sw_test_i = X_sw[test_set]
y_sw_test_i = y_sw[test_set]
ebm_sw_i = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw_i.fit(X_sw_i, y_sw_i)
ebm_sw_i_expl = ebm_sw_i.explain_global()
show(ebm_sw_i_expl)
```

In the following, use 0 for dist_coast, 1 for depth, 2 for chla_month, 3 for sst_day, 4 for salinity_month, 5 for cell_effort_value, 6 for day_of_year, and 9 for dist X sst. Then export the resulting JSON data to R for further use.

```
ebm_sw_i_expl.data(0)
{'type': 'univariate',
 'names': [0.2,
  1.05,
  1.15,
  1.25,
  1.35,
  1.45,
  1.55,
  1.65,
  1.75,
  1.85,
  1.95,
  2.05,
  2.1500000000000004,
  2.25,
  2.3499999999999996,
  2.45,
  2.55,
  2.6500000000000004,
  2.75,
  2.8499999999999996,
  2.95,
  3.05,
  3.1500000000000004,
  3.25,
  3.3499999999999996,
  3.45,
  3.55,
  3.6500000000000004,
  3.75,
  3.8499999999999996,
  3.95,
  4.05,
  4.15,
  4.25,
  4.35,
  4.45,
4.55,
4.65,
4.75,
4.85,
4.95,
5.05,
5.15,
5.25,
5.35,
5.45,
5.55,
5.65,
5.75,
5.85,
5.95,
6.05,
6.15,
6.25,
6.35,
6.45,
6.55,
6.65,
6.75,
6.85,
6.95,
7.05,
7.15,
7.25,
7.35,
7.45,
7.55,
7.65,
7.75,
7.85,
7.95,
8.05,
8.149999999999999,
8.25,
8.350000000000001,
8.45,
8.55,
8.649999999999999,
8.75,
8.850000000000001,
8.95,
9.05,
9.149999999999999,
9.25,
9.350000000000001,
9.45,
9.55,
9.649999999999999,
9.75,
9.850000000000001,
9.95,
10.05,
10.149999999999999,
10.25,
10.350000000000001,
10.45,
10.55,
10.649999999999999,
10.75,
10.850000000000001,
10.95,
11.05,
11.149999999999999,
11.25,
11.350000000000001,
11.45,
11.55,
11.649999999999999,
11.75,
11.850000000000001,
11.95,
12.05,
12.149999999999999,
12.25,
12.350000000000001,
12.45,
12.55,
12.649999999999999,
12.75,
12.850000000000001,
12.95,
13.05,
13.149999999999999,
13.25,
13.350000000000001,
13.45,
13.55,
13.649999999999999,
13.75,
13.850000000000001,
13.95,
14.05,
14.149999999999999,
14.25,
14.350000000000001,
14.45,
14.55,
14.649999999999999,
14.75,
14.850000000000001,
14.95,
15.05,
15.149999999999999,
15.25,
15.350000000000001,
15.45,
15.55,
15.649999999999999,
15.75,
15.850000000000001,
15.95,
16.05,
16.15,
16.25,
16.35,
16.45,
16.55,
16.65,
16.75,
16.85,
16.95,
17.05,
17.15,
17.25,
17.35,
17.45,
17.55,
17.65,
17.75,
17.85,
17.95,
18.05,
18.15,
18.25,
18.35,
18.45,
18.55,
18.65,
18.75,
18.85,
18.95,
19.05,
19.15,
19.25,
19.35,
19.45,
19.55,
19.65,
19.75,
19.85,
19.95,
20.05,
20.15,
20.25,
20.35,
20.45,
20.55,
20.65,
 20.75,
 20.85,
 20.95,
 21.05,
 21.15,
 21.25,
 21.35,
 21.45,
 21.55,
 21.65,
 21.75,
 21.85,
 21.95,
 22.05,
 22.15,
 22.25,
 22.35,
 22.45,
 22.55,
 22.65,
 22.75,
 22.85,
 22.95,
 23.05,
 23.15,
 23.25,
 23.35,
 23.45,
 23.55,
 23.65,
 23.75,
 23.85,
 23.95,
 24.05,
 24.15,
 24.25,
 24.35,
 24.45,
 24.55,
 24.75,
 24.95,
 25.15,
 25.25,
 25.35,
 25.55,
 25.75,
 26.05,
 26.25,
 26.45,
 26.85,
 27.35,
 30.4],
'scores': array([-0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 ,
       -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 ,
       -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 ,
       -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 ,
       -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 , -0.2995615 ,
       -0.2995615 , -0.2995615 , -0.29824667, -0.29437885, -0.29772958,
       -0.29772958, -0.29772958, -0.29772958, -0.29772958, -0.29772958,
       -0.29772958, -0.27251835, -0.28933322, -0.28933322, -0.28634225,
       -0.28634225, -0.28634225, -0.27994943, -0.26875879, -0.28008186,
       -0.28008186, -0.28008186, -0.28008186, -0.27888751, -0.27888751,
       -0.27816684, -0.27816684, -0.27385504, -0.27454132, -0.27454132,
       -0.27454132, -0.27454132, -0.26364397, -0.26364397, -0.26554766,
       -0.27223603, -0.27223603, -0.2652929 , -0.27512646, -0.27520383,
       -0.27520383, -0.27520383, -0.27520383, -0.27520383, -0.27520383,
       -0.27520383, -0.27520383, -0.27033321, -0.27144535, -0.27144535,
       -0.21656722, -0.12623455, -0.23529766, -0.23529766, -0.23529766,
       -0.23529766, -0.20000036, -0.18462317, -0.20692032, -0.17544247,
       -0.17544247, -0.1692646 , -0.1770408 , -0.1770408 , -0.09984146,
       -0.05159566, -0.1000555 , -0.06564448, -0.07926287, -0.13167085,
       -0.13167085, -0.13006961, -0.13807553, -0.15450472, -0.15436751,
       -0.15986396, -0.15986396, -0.15892411, -0.15892411, -0.15889452,
       -0.15889452, -0.15889452, -0.14386663, -0.15828549, -0.15828549,
       -0.15828549, -0.15482545, -0.15482545, -0.15482545, -0.15482545,
       -0.15007437, -0.15092486, -0.15200919, -0.15200919, -0.14063032,
       -0.14562208, -0.13443198, -0.1446879 , -0.1446879 , -0.1446879 ,
       -0.09180298, -0.09560944, -0.10958517, -0.10958517,  0.04990315,
       -0.04981258, -0.0830208 , -0.0830208 , -0.07666534, -0.07838741,
       -0.06958515, -0.07318981, -0.05263978, -0.05263978,  0.04035672,
        0.08512379,  0.02792643,  0.10786051,  0.03816354,  0.03225973,
        0.02633198,  0.01840733,  0.02944754,  0.02879492,  0.10586885,
        0.04371575,  0.05592413,  0.05061779,  0.04269334,  0.04269334,
        0.18450102,  0.15468897,  0.12761795,  0.09713051,  0.09713051,
        0.1097526 ,  0.1070641 ,  0.1070641 ,  0.346244  ,  0.32782492,
        0.79134703,  1.00258263,  0.94667334,  0.74501377,  0.51447776,
        0.51813372,  0.53011322,  0.5142555 ,  0.51613336,  0.49872136,
        0.50124575,  0.50031781,  0.48288562,  0.47369961,  0.47369961,
        0.47369961,  0.56204314,  0.51945112,  0.50029917,  0.50029917,
        0.83951232,  0.81343686,  0.80180005,  0.97445177,  0.97026594,
        0.96246806,  0.58871356,  0.54920157,  0.56264514,  0.54710498,
        0.48316363,  0.47003433,  0.4590112 ,  0.45680331,  0.53662738,
        0.53814374,  0.5359908 ,  0.55648802,  0.40559638,  0.40964524,
        0.41186999,  0.34769822,  0.34769822,  0.34769822,  0.49561119,
        0.32380194,  0.32417746,  0.28771337,  0.28771337,  0.32983924,
        0.27251848,  0.27251848,  0.29149129,  0.33859647,  0.30303008,
        0.35418386,  0.56953471,  0.18093952,  0.18229375,  0.12360135,
        0.12360135,  0.16040045,  0.15978434,  0.07130188,  0.07130188,
        0.07130188,  0.10543395,  0.07794012,  0.07794012,  0.07794012,
        0.07794012,  0.28507197,  0.29974708,  0.28140673,  0.45389997,
        0.17996155, -0.05366333, -0.05366333, -0.05366333,  0.37618493,
       -0.01407345, -0.0141671 ,  0.72620622,  0.83008878]),
 'scores_range': (-0.9277708011444358, 4.832127905656579),
 'upper_bounds': array([-2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20890704e-01,
       -2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20890704e-01,
       -2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20890704e-01,
       -2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20890704e-01,
       -2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20890704e-01,
       -2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20890704e-01,
       -2.20890704e-01, -2.20890704e-01, -2.20890704e-01, -2.20126511e-01,
       -2.18481819e-01, -2.20031094e-01, -2.20031094e-01, -2.20031094e-01,
       -2.20031094e-01, -2.20031094e-01, -2.20031094e-01, -2.20031094e-01,
       -1.84077376e-01, -2.11757857e-01, -2.11757857e-01, -2.09237162e-01,
       -2.09237162e-01, -2.09237162e-01, -2.08447189e-01, -1.72326360e-01,
       -2.09563676e-01, -2.09563676e-01, -2.09563676e-01, -2.09563676e-01,
       -2.07131435e-01, -2.07131435e-01, -2.07192889e-01, -2.07192889e-01,
       -2.00229144e-01, -2.01616113e-01, -2.01616113e-01, -2.01616113e-01,
       -2.01616113e-01, -1.82415543e-01, -1.82415543e-01, -1.86199370e-01,
       -1.98191329e-01, -1.98191329e-01, -1.91255978e-01, -2.01711229e-01,
       -2.01830398e-01, -2.01830398e-01, -2.01830398e-01, -2.01830398e-01,
       -2.01830398e-01, -2.01830398e-01, -2.01830398e-01, -2.01830398e-01,
       -1.97886088e-01, -1.99291180e-01, -1.99291180e-01, -1.20143010e-01,
        2.79663408e-02, -1.77007100e-01, -1.77007100e-01, -1.77007100e-01,
       -1.77007100e-01, -1.31594680e-01, -1.08966708e-01, -1.58256059e-01,
       -1.00528960e-01, -1.00528960e-01, -9.43429931e-02, -1.15053689e-01,
       -1.15053689e-01, -2.39810818e-03,  6.00066403e-02, -1.99184000e-02,
        6.81454249e-02,  5.09119180e-02, -5.61835186e-02, -5.61835186e-02,
       -5.53375793e-02, -6.36584036e-02, -8.57721137e-02, -8.56262916e-02,
       -9.05223544e-02, -9.05223544e-02, -8.93966486e-02, -8.93966486e-02,
       -8.93562585e-02, -8.93562585e-02, -8.93562585e-02, -5.15859110e-02,
       -8.89087166e-02, -8.89087166e-02, -8.89087166e-02, -8.63638233e-02,
       -8.63638233e-02, -8.63638233e-02, -8.63638233e-02, -7.86407688e-02,
       -8.01152806e-02, -8.15099925e-02, -8.15099925e-02, -8.11535939e-02,
       -8.09620676e-02, -5.64542898e-02, -7.53382018e-02, -7.53382018e-02,
       -7.53382018e-02,  1.37953610e-02, -2.70052760e-03, -3.12618520e-02,
       -3.12618520e-02,  2.15044880e-01,  5.20025443e-02,  2.98768066e-04,
        2.98768066e-04,  6.10369959e-03,  3.32828907e-03,  6.20176480e-03,
        2.84256529e-03,  3.97246085e-02,  3.97246085e-02,  1.79664379e-01,
        2.03686957e-01,  1.33014213e-01,  2.77115370e-01,  1.33786926e-01,
        1.26323773e-01,  1.21278626e-01,  1.08518564e-01,  1.26235839e-01,
        1.25276481e-01,  2.61206119e-01,  1.28640575e-01,  1.31841718e-01,
        1.31383661e-01,  1.15240978e-01,  1.15240978e-01,  3.32184462e-01,
        2.74919649e-01,  2.11669901e-01,  1.71798290e-01,  1.71798290e-01,
        1.74973976e-01,  1.74364464e-01,  1.74364464e-01,  4.82099957e-01,
        4.53782905e-01,  1.02011237e+00,  1.24460380e+00,  1.13670965e+00,
        9.09370590e-01,  6.08252909e-01,  6.16396907e-01,  6.23719047e-01,
        5.99340150e-01,  6.03522304e-01,  5.75771699e-01,  5.81686177e-01,
        5.76986643e-01,  5.46223650e-01,  5.29526349e-01,  5.29526349e-01,
        5.29526349e-01,  6.95878539e-01,  5.90937632e-01,  5.57447090e-01,
        5.57447090e-01,  1.03543097e+00,  9.70802636e-01,  9.54199983e-01,
        1.16740243e+00,  1.15344535e+00,  1.15832784e+00,  6.57415554e-01,
        6.12373489e-01,  6.30958586e-01,  6.15226089e-01,  5.46203423e-01,
        5.31624419e-01,  5.22328279e-01,  5.18576659e-01,  6.41487722e-01,
        6.68751892e-01,  6.50667673e-01,  7.12205508e-01,  4.94704108e-01,
        5.06386030e-01,  5.15790755e-01,  4.25296116e-01,  4.25296116e-01,
        4.25296116e-01,  6.68539850e-01,  4.07964414e-01,  4.08677493e-01,
        3.61380983e-01,  3.61380983e-01,  5.06199146e-01,  3.58087228e-01,
        3.58087228e-01,  4.13294920e-01,  4.99192585e-01,  4.41297520e-01,
        5.21690186e-01,  8.36497240e-01,  3.21317962e-01,  3.24801638e-01,
        2.45618899e-01,  2.45618899e-01,  3.50552086e-01,  3.22453558e-01,
        1.97224448e-01,  1.97224448e-01,  1.97224448e-01,  2.37435726e-01,
        1.91874192e-01,  1.91874192e-01,  1.91874192e-01,  1.91874192e-01,
               4.95755426e-01, 5.35645408e-01, 5.28199287e-01, 8.03227168e-01,
               4.30176216e-01, 1.23115366e-01, 1.23115366e-01, 1.23115366e-01,
               7.36373712e-01, 1.65235878e-01, 1.65261320e-01, 1.11325699e+00,
               1.28548435e+00]),
        'lower_bounds': array([-0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 ,
              -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 ,
              -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 ,
              -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 ,
              -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 , -0.3782323 ,
              -0.3782323 , -0.3782323 , -0.37636684, -0.37027588, -0.37542806,
              -0.37542806, -0.37542806, -0.37542806, -0.37542806, -0.37542806,
              -0.37542806, -0.36095932, -0.36690858, -0.36690858, -0.36344734,
              -0.36344734, -0.36344734, -0.35145168, -0.36519121, -0.35060004,
              -0.35060004, -0.35060004, -0.35060004, -0.35064358, -0.35064358,
              -0.34914079, -0.34914079, -0.34748094, -0.34746653, -0.34746653,
              -0.34746653, -0.34746653, -0.3448724 , -0.3448724 , -0.34489596,
              -0.34628074, -0.34628074, -0.33932983, -0.3485417 , -0.34857726,
              -0.34857726, -0.34857726, -0.34857726, -0.34857726, -0.34857726,
              -0.34857726, -0.34857726, -0.34278032, -0.34359952, -0.34359952,
              -0.31299143, -0.28043543, -0.29358822, -0.29358822, -0.29358822,
              -0.29358822, -0.26840604, -0.26027963, -0.25558459, -0.25035598,
              -0.25035598, -0.2441862 , -0.23902792, -0.23902792, -0.19728482,
              -0.16319797, -0.1801926 , -0.19943439, -0.20943766, -0.20715818,
              -0.20715818, -0.20480163, -0.21249267, -0.22323733, -0.22310872,
              -0.22920557, -0.22920557, -0.22845158, -0.22845158, -0.22843278,
              -0.22843278, -0.22843278, -0.23614735, -0.22766226, -0.22766226,
              -0.22766226, -0.22328708, -0.22328708, -0.22328708, -0.22328708,
              -0.22150798, -0.22173443, -0.22250838, -0.22250838, -0.20010705,
              -0.21028209, -0.21240968, -0.21403761, -0.21403761, -0.21403761,
              -0.19740133, -0.18851835, -0.18790848, -0.18790848, -0.11523857,
              -0.15162771, -0.16634037, -0.16634037, -0.15943437, -0.1601031 ,
              -0.14537207, -0.14922219, -0.14500418, -0.14500418, -0.09895094,
              -0.03343938, -0.07716135, -0.06139435, -0.05745984, -0.06180431,
              -0.06861466, -0.07170391, -0.06734077, -0.06768665, -0.04946841,
              -0.04120908, -0.01999345, -0.03014808, -0.02985431, -0.02985431,
               0.03681758,  0.03445829,  0.04356599,  0.02246273,  0.02246273,
               0.04453122,  0.03976374,  0.03976374,  0.21038804,  0.20186693,
               0.56258169,  0.76056146,  0.75663703,  0.58065695,  0.42070261,
               0.41987054,  0.43650739,  0.42917085,  0.42874442,  0.42167103,
               0.42080531,  0.42364898,  0.41954759,  0.41787287,  0.41787287,
               0.41787287,  0.42820773,  0.44796461,  0.44315126,  0.44315126,
               0.64359366,  0.65607108,  0.64940011,  0.7815011 ,  0.78708653,
               0.76660828,  0.52001157,  0.48602965,  0.4943317 ,  0.47898388,
               0.42012383,  0.40844424,  0.39569411,  0.39502996,  0.43176704,
               0.40753558,  0.42131392,  0.40077052,  0.31648865,  0.31290445,
               0.30794923,  0.27010033,  0.27010033,  0.27010033,  0.32268254,
               0.23963947,  0.23967742,  0.21404576,  0.21404576,  0.15347933,
               0.18694974,  0.18694974,  0.16968765,  0.17800036,  0.16476264,
               0.18667754,  0.30257217,  0.04056109,  0.03978586,  0.00158381,
               0.00158381, -0.02975119, -0.00288488, -0.05462068, -0.05462068,
              -0.05462068, -0.02656783, -0.03599396, -0.03599396, -0.03599396,
              -0.03599396,  0.07438851,  0.06384875,  0.03461416,  0.10457278,
              -0.07025312, -0.23044202, -0.23044202, -0.23044202,  0.01599615,
              -0.19338279, -0.19359552,  0.33915545,  0.3746932 ]),
       'density': {'names': [0.2,
         1.71,
         3.22,
         4.7299999999999995,
         6.24,
         7.75,
         9.259999999999998,
         10.769999999999998,
         12.28,
         13.79,
         15.299999999999999,
         16.81,
         18.32,
         19.83,
         21.34,
         22.849999999999998,
         24.36,
         25.869999999999997,
         27.38,
         28.889999999999997,
         30.4],
        'scores': [544.0,
         1973.0,
         1922.0,
         2130.0,
         2188.0,
         1784.0,
         1530.0,
         1581.0,
         1548.0,
         1751.0,
         1637.0,
         1531.0,
         1741.0,
         1825.0,
         1110.0,
         1044.0,
         528.0,
         173.0,
         37.0,
         4.0]},
       'meta': {'label_names': [0, 1]}}
```

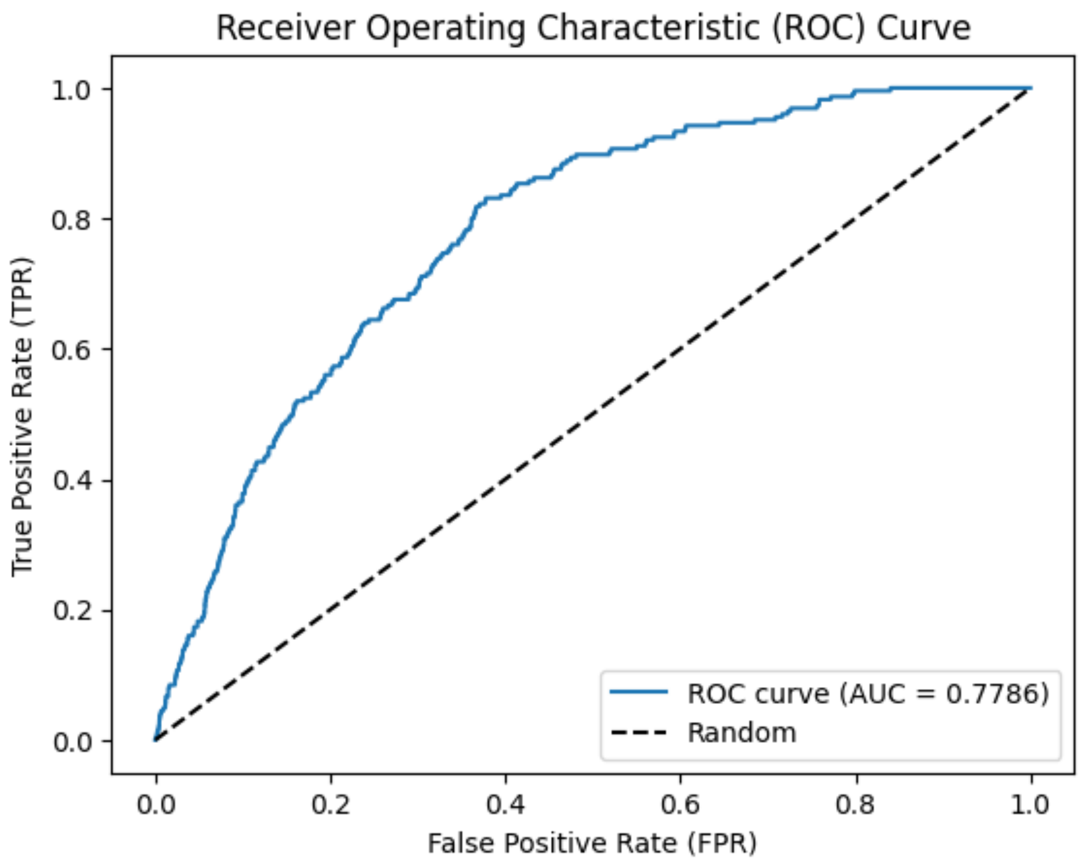
Does using ROSE improve anything on cross-validation?

Whether or not ROSE is implemented in python, it is easier to use it in R and import. The same 5-fold cross-validation procedure is followed as above, except each training set is augmented using the ROSE technique.

```
ROSE_train_0 = pd.read_csv("../TryROSE/ROSEtrain0.csv")
ROSE_X_train_0 = ROSE_train_0.drop(columns = ['presence'])
ROSE_y_train_0 = ROSE_train_0['presence']
ROSE_test_0 = pd.read_csv("../TryROSE/ROSEtest0.csv")
ROSE_X_test_0 = ROSE_test_0.drop(columns = ['presence'])
ROSE_y_test_0 = ROSE_test_0['presence']
```

```
ebm_sw_ROSE_0 = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw_ROSE_0.fit(ROSE_X_train_0, ROSE_y_train_0)
ROSE_pred = ebm_sw_ROSE_0.predict_proba(ROSE_X_test_0)[:, 1]
ROSE_truth = ROSE_y_test_0.to_numpy()
ROSE_train_1 = pd.read_csv("../TryROSE/ROSEtrain1.csv")
ROSE_X_train_1 = ROSE_train_1.drop(columns = ['presence'])
ROSE_y_train_1 = ROSE_train_1['presence']
ROSE_test_1 = pd.read_csv("../TryROSE/ROSEtest1.csv")
ROSE_X_test_1 = ROSE_test_1.drop(columns = ['presence'])
ROSE_y_test_1 = ROSE_test_1['presence']

ebm_sw_ROSE_1 = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw_ROSE_1.fit(ROSE_X_train_1, ROSE_y_train_1)
ExplainableBoostingClassifier(inner_bags=25, outer_bags=25)
ROSE_pred = np.concatenate((ROSE_pred, ebm_sw_ROSE_1.predict_proba(ROSE_X_test_1)[:, 1]),
axis = 0)
ROSE_truth = np.concatenate((ROSE_truth, ROSE_y_test_1.to_numpy()), axis = 0)
ROSE_train_2 = pd.read_csv("../TryROSE/ROSEtrain2.csv")
ROSE_X_train_2 = ROSE_train_2.drop(columns = ['presence'])
ROSE_y_train_2 = ROSE_train_2['presence']
ROSE_test_2 = pd.read_csv("../TryROSE/ROSEtest2.csv")
ROSE_X_test_2 = ROSE_test_2.drop(columns = ['presence'])
ROSE_y_test_2 = ROSE_test_2['presence']

ebm_sw_ROSE_2 = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw_ROSE_2.fit(ROSE_X_train_2, ROSE_y_train_2)

ROSE_pred = np.concatenate((ROSE_pred, ebm_sw_ROSE_2.predict_proba(ROSE_X_test_2)[:, 1]),
axis = 0)
ROSE_truth = np.concatenate((ROSE_truth, ROSE_y_test_2.to_numpy()), axis = 0)
ROSE_train_3 = pd.read_csv("../TryROSE/ROSEtrain3.csv")
ROSE_X_train_3 = ROSE_train_3.drop(columns = ['presence'])
ROSE_y_train_3 = ROSE_train_3['presence']
ROSE_test_3 = pd.read_csv("../TryROSE/ROSEtest3.csv")
ROSE_X_test_3 = ROSE_test_3.drop(columns = ['presence'])
ROSE_y_test_3 = ROSE_test_3['presence']

ebm_sw_ROSE_3 = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw_ROSE_3.fit(ROSE_X_train_3, ROSE_y_train_3)

ROSE_pred = np.concatenate((ROSE_pred, ebm_sw_ROSE_3.predict_proba(ROSE_X_test_3)[:, 1]),
axis = 0)
ROSE_truth = np.concatenate((ROSE_truth, ROSE_y_test_3.to_numpy()), axis = 0)
ROSE_train_4 = pd.read_csv("../TryROSE/ROSEtrain4.csv")
ROSE_X_train_4 = ROSE_train_4.drop(columns = ['presence'])
ROSE_y_train_4 = ROSE_train_4['presence']
ROSE_test_4 = pd.read_csv("../TryROSE/ROSEtest4.csv")
ROSE_X_test_4 = ROSE_test_4.drop(columns = ['presence'])
ROSE_y_test_4 = ROSE_test_4['presence']

ebm_sw_ROSE_4 = ExplainableBoostingClassifier(outer_bags = 25, inner_bags = 25)
ebm_sw_ROSE_4.fit(ROSE_X_train_4, ROSE_y_train_4)
```

```
ROSE_pred = np.concatenate((ROSE_pred, ebm_sw_ROSE_3.predict_proba(ROSE_X_test_4)[:, 1]),
axis = 0)
ROSE_truth = np.concatenate((ROSE_truth, ROSE_y_test_4.to_numpy()), axis = 0)
ROSE_fpr_k, ROSE_tpr_k, ROSE_thresholds_k = roc_curve(ROSE_truth, ROSE_pred)

# Calculate the area under the ROC curve (AUC)
ROSE_auc_k = roc_auc_score(ROSE_truth, ROSE_pred)

# Plot the ROC curve
plt.plot(ROSE_fpr_k, ROSE_tpr_k, label='ROC curve (AUC = {:.4f})'.format(ROSE_auc_k))
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```



Worse