trees_poner

June 9, 2021

Al ser una nueva librería, nos parece interesante ir comentando cómo construir los modelos.

La lectura de datos se realiza de la misma forma que en casos anteriores

[72]:
```
import pandas as pd
import numpy as np

import tensorflow_decision_forests as tfdf
from wurlitzer import sys_pipes
```

[73]:
```
data = pd.read_csv("krkopt.data", header=None)
data.columns = ["wkc", "wkr", "wrc", "wrr", "bkc", "bkr", "opt rank" ]
```

[74]:
```
data.head()
```

[74]:

	wkc	wkr	wrc	wrr	bkc	bkr	opt rank
0	a	1	b	3	c	2	draw
1	a	1	c	1	c	2	draw
2	a	1	c	1	d	1	draw
3	a	1	c	1	d	2	draw
4	a	1	c	2	c	1	draw

Es necesario destacar cuál va a ser la columna que contenga la variable a predecir para usar *tfdf*. En nuestro caso, *opt rank*.

Mostramos también las clases de esta variable

[75]:
```
label = "opt rank"

classes = data[label].unique().tolist()
print(f"Label classes: {classes}")

#convert to integer category

data[label] = data[label].map(classes.index)
```

```
Label classes: ['draw', 'zero', 'one', 'two', 'three', 'four', 'five', 'six',
'seven', 'eight', 'nine', 'ten', 'eleven', 'twelve', 'thirteen', 'fourteen',
'fifteen', 'sixteen']
```

Creamos el conjunto de entrenamiento y test con una proporción 80-20.

[76]:
```
seed = 1
np.random.seed(seed)
```

[77]:
```
def split_dataset(dataset, test_ratio=0.2):
    test_indices = np.random.rand(len(dataset)) < test_ratio
    return dataset[~test_indices], dataset[test_indices]

train, test = split_dataset(data)
test_aux = test.copy()
```

[78]:
```
print("{} examples in training, {} examples for testing.".format(
    len(train), len(test)))
```

```
22450 examples in training, 5606 examples for testing.
```

Encontramos las primeras peculariedades. Es necesario utilizar la función *pd_dataframe_to_tf_dataset* para transformar nuestro conjunto de datos a un objeto que pueda ser procesado por la librería *yggrdrasil*. Más adelante comentaremos un *bug* que hemos encontrado en esta función que nos impide realizar ciertas pruebas.

[79]:
```
train = tfdf.keras.pd_dataframe_to_tf_dataset(train, label=label)
test = tfdf.keras.pd_dataframe_to_tf_dataset(test, label=label)
```

0.1 Creación de Random Forest con datos *raw*

Creamos un modelo de Random Forest usando la función *RandomForestModel*. Es necesario compilar el modelo con la métrica que nosotros queramos. Inicialmente intentamos usar Precision, para poder comparar con las redes neuronales, pero obtuvimos error ya que no parece completamente compatible esta métrica aún. Por ello, usamos *accuracy* y más tarde usamos precision.

Además, en este primer modelo, mostramos un log usando *sys_pipes* que permite ver los entresijos y detalles del entrenamiento. Como vemos, la propia librería detecta las categorías de las variables predictoras.

En el caso de los árboles de decisión, no se utilizan épocas ya que los distintos árboles que componen el bosque de árboles se entrenan con todo el conjunto de entrenamiento. Además, el propio algoritmo tiene una medida de estimación de validación, por lo que tampoco hay que usar conjunto de validación.

[80]:
```
from tensorflow.keras.metrics import Precision
model_rf= tfdf.keras.RandomForestModel()

model_rf.compile(
                metrics=["accuracy"])

with sys_pipes():
    model_rf.fit(x=train)
```

```
351/351 [==============================] - 1s 1ms/step
[INFO kernel.cc:746] Start Yggdrasil model training
[INFO kernel.cc:747] Collect training examples
[INFO kernel.cc:392] Number of batches: 351
[INFO kernel.cc:393] Number of examples: 22450
[INFO kernel.cc:769] Dataset:
Number of records: 22450
Number of columns: 7

Number of columns by type:
        CATEGORICAL: 4 (57.1429%)
        NUMERICAL: 3 (42.8571%)

Columns:

CATEGORICAL: 4 (57.1429%)
        0: "bkc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 3859 (17.1893%)
        2: "wkc" CATEGORICAL has-dict vocab-size:5 zero-ood-items most-
frequent:"d" 9677 (43.1047%)
        4: "wrc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"f" 2930 (13.0512%)
        6: "__LABEL" CATEGORICAL integerized vocab-size:19 no-ood-item

NUMERICAL: 3 (42.8571%)
        1: "bkr" NUMERICAL mean:4.45768 min:1 max:8 sd:2.24698
        3: "wkr" NUMERICAL mean:1.84909 min:1 max:4 sd:0.924757
        5: "wrr" NUMERICAL mean:4.51042 min:1 max:8 sd:2.27881

Terminology:
        nas: Number of non-available (i.e. missing) values.
        ood: Out of dictionary.
        manually-defined: Attribute which type is manually defined by the user
i.e. the type was not automatically inferred.
        tokenized: The attribute value is obtained through tokenization.
        has-dict: The attribute is attached to a string dictionary e.g. a
categorical attribute stored as a string.
        vocab-size: Number of unique values.

[INFO kernel.cc:772] Configure learner
[INFO kernel.cc:797] Training config:
learner: "RANDOM_FOREST"
features: "bkc"
features: "bkr"
features: "wkc"
features: "wkr"
features: "wrc"
```

```
features: "wrr"
label: "__LABEL"
task: CLASSIFICATION
[yggdrasil_decision_forests.model.random_forest.proto.random_forest_config] {
  num_trees: 300
  decision_tree {
    max_depth: 16
    min_examples: 5
    in_split_min_examples_check: true
    missing_value_policy: GLOBAL_IMPUTATION
    allow_na_conditions: false
    categorical_set_greedy_forward {
      sampling: 0.1
      max_num_items: -1
      min_item_frequency: 1
    }
    growing_strategy_local {
    }
    categorical {
      cart {
      }
    }
    num_candidate_attributes_ratio: -1
    axis_aligned_split {
    }
  }
  winner_take_all_inference: true
  compute_oob_performances: true
  compute_oob_variable_importances: false
  adapt_bootstrap_size_ratio_for_maximum_training_duration: false
}

[INFO kernel.cc:800] Deployment config:

[INFO kernel.cc:837] Train model
[INFO random_forest.cc:303] Training random forest on 22450 example(s) and 6
feature(s).
[INFO random_forest.cc:578] Training of tree  1/300 (tree index:0) done
accuracy:0.635458 logloss:13.1394
[INFO random_forest.cc:578] Training of tree  11/300 (tree index:10) done
accuracy:0.690411 logloss:5.00099
[INFO random_forest.cc:578] Training of tree  21/300 (tree index:20) done
accuracy:0.730788 logloss:2.67814
[INFO random_forest.cc:578] Training of tree  31/300 (tree index:30) done
accuracy:0.742316 logloss:1.95862
[INFO random_forest.cc:578] Training of tree  41/300 (tree index:40) done
accuracy:0.750379 logloss:1.58524
[INFO random_forest.cc:578] Training of tree  51/300 (tree index:50) done
```

```
accuracy:0.753497 logloss:1.37915
[INFO random_forest.cc:578] Training of tree  61/300 (tree index:60) done
accuracy:0.755768 logloss:1.26819
[INFO random_forest.cc:578] Training of tree  71/300 (tree index:70) done
accuracy:0.758708 logloss:1.17384
[INFO random_forest.cc:578] Training of tree  81/300 (tree index:80) done
accuracy:0.75902 logloss:1.11116
[INFO random_forest.cc:578] Training of tree  91/300 (tree index:91) done
accuracy:0.760624 logloss:1.05555
[INFO random_forest.cc:578] Training of tree  101/300 (tree index:101) done
accuracy:0.762272 logloss:1.02269
[INFO random_forest.cc:578] Training of tree  111/300 (tree index:108) done
accuracy:0.764276 logloss:0.992012
[INFO random_forest.cc:578] Training of tree  121/300 (tree index:116) done
accuracy:0.763875 logloss:0.96005
[INFO random_forest.cc:578] Training of tree  131/300 (tree index:130) done
accuracy:0.764365 logloss:0.940543
[INFO random_forest.cc:578] Training of tree  141/300 (tree index:140) done
accuracy:0.764009 logloss:0.928943
[INFO random_forest.cc:578] Training of tree  151/300 (tree index:150) done
accuracy:0.764365 logloss:0.913852
[INFO random_forest.cc:578] Training of tree  161/300 (tree index:160) done
accuracy:0.764811 logloss:0.89613
[INFO random_forest.cc:578] Training of tree  171/300 (tree index:170) done
accuracy:0.765479 logloss:0.881226
[INFO random_forest.cc:578] Training of tree  181/300 (tree index:180) done
accuracy:0.764944 logloss:0.866638
[INFO random_forest.cc:578] Training of tree  191/300 (tree index:190) done
accuracy:0.765523 logloss:0.864695
[INFO random_forest.cc:578] Training of tree  201/300 (tree index:200) done
accuracy:0.764855 logloss:0.848468
[INFO random_forest.cc:578] Training of tree  211/300 (tree index:210) done
accuracy:0.765434 logloss:0.843513
[INFO random_forest.cc:578] Training of tree  221/300 (tree index:220) done
accuracy:0.766414 logloss:0.837226
[INFO random_forest.cc:578] Training of tree  231/300 (tree index:231) done
accuracy:0.765746 logloss:0.833018
[INFO random_forest.cc:578] Training of tree  241/300 (tree index:240) done
accuracy:0.765924 logloss:0.828863
[INFO random_forest.cc:578] Training of tree  251/300 (tree index:250) done
accuracy:0.766503 logloss:0.822741
[INFO random_forest.cc:578] Training of tree  261/300 (tree index:260) done
accuracy:0.766236 logloss:0.819977
[INFO random_forest.cc:578] Training of tree  271/300 (tree index:270) done
accuracy:0.766503 logloss:0.815393
[INFO random_forest.cc:578] Training of tree  281/300 (tree index:280) done
accuracy:0.767528 logloss:0.810447
[INFO random_forest.cc:578] Training of tree  291/300 (tree index:290) done
```

```
accuracy:0.767305 logloss:0.804649
[INFO random_forest.cc:578] Training of tree  300/300 (tree index:299) done
accuracy:0.765702 logloss:0.803169
[INFO random_forest.cc:645] Final OOB metrics: accuracy:0.765702
logloss:0.803169
[INFO kernel.cc:856] Export model in log directory: /tmp/tmptiumy4d4
[INFO kernel.cc:864] Save model in resources
[INFO kernel.cc:929] Loading model from path
[INFO decision_forest.cc:590] Model loaded with 300 root(s), 1374910 node(s),
and 6 input feature(s).
[INFO abstract_model.cc:876] Engine "RandomForestGeneric" built
[INFO kernel.cc:797] Use fast generic engine
```

Se han creado trescientos árboles. El log nos muestra algunos de estos. Vemos que en general se obtiene un 76% de *accuracy*. Podemos obtener un resumen más reducido usando *summary*

[81]:
```
model_rf.summary()
```

```
Model: "random_forest_model_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Total params: 1
Trainable params: 0
Non-trainable params: 1
_________________________________________________________________
Type: "RANDOM_FOREST"
Task: CLASSIFICATION
Label: "__LABEL"

Input Features (6):
        bkc
        bkr
        wkc
        wkr
        wrc
        wrr

No weights

Variable Importance: NUM_NODES:
    1. "wrr" 216342.000000 ################
    2. "wrc" 165397.000000 ###########
    3. "bkr" 119478.000000 #######
    4. "bkc" 112101.000000 #######
    5. "wkc" 51255.000000 ##
    6. "wkr" 22732.000000
```

```
Variable Importance: NUM_AS_ROOT:
    1. "bkr" 205.000000 ################
    2. "wkr" 78.000000 #####
    3. "wkc" 12.000000
    4. "bkc"  5.000000

Variable Importance: SUM_SCORE:
    1. "wrr" 2938632.366295 ################
    2. "wrc" 2833222.359850 ##############
    3. "bkr" 2734362.605383 #############
    4. "bkc" 2527649.034721 ###########
    5. "wkr" 1702105.100839 ###
    6. "wkc" 1365919.120243

Variable Importance: MEAN_MIN_DEPTH:
    1. "__LABEL" 12.360672 ################
    2.     "wrr"  7.176756 #########
    3.     "wrc"  5.558875 ######
    4.     "wkc"  4.410273 #####
    5.     "wkr"  2.251511 ##
    6.     "bkc"  2.117759 ##
    7.     "bkr"  0.481818



Winner take all: true
Out-of-bag evaluation: accuracy:0.765702 logloss:0.803169
Number of trees: 300
Total number of nodes: 1374910

Number of nodes by tree:
Count: 300 Average: 4583.03 StdDev: 81.4853
Min: 4331 Max: 4813 Ignored: 0
----------------------------------------------
[ 4331, 4355)  1   0.33%   0.33%
[ 4355, 4379)  0   0.00%   0.33%
[ 4379, 4403)  1   0.33%   0.67%
[ 4403, 4427)  2   0.67%   1.33% #
[ 4427, 4451)  6   2.00%   3.33% ##
[ 4451, 4475) 14   4.67%   8.00% ####
[ 4475, 4500) 25   8.33%  16.33% ######
[ 4500, 4524) 27   9.00%  25.33% #######
[ 4524, 4548) 24   8.00%  33.33% ######
[ 4548, 4572) 40  13.33%  46.67% ##########
[ 4572, 4596) 38  12.67%  59.33% ##########
[ 4596, 4620) 29   9.67%  69.00% #######
[ 4620, 4644) 30  10.00%  79.00% ########
[ 4644, 4669) 18   6.00%  85.00% #####
```

```
[ 4669, 4693) 15   5.00%  90.00% ####
[ 4693, 4717) 12   4.00%  94.00% ###
[ 4717, 4741)  8   2.67%  96.67% ##
[ 4741, 4765)  4   1.33%  98.00% #
[ 4765, 4789)  1   0.33%  98.33%
[ 4789, 4813]  5   1.67% 100.00% #

Depth by leafs:
Count: 687605 Average: 12.3605 StdDev: 1.81321
Min: 5 Max: 15 Ignored: 0
----------------------------------------------
[  5,  6)     10   0.00%   0.00%
[  6,  7)    215   0.03%   0.03%
[  7,  8)   2159   0.31%   0.35%
[  8,  9)  10417   1.51%   1.86% #
[  9, 10)  31503   4.58%   6.44% ##
[ 10, 11)  68800  10.01%  16.45% #####
[ 11, 12) 108362  15.76%  32.21% ########
[ 12, 13) 129972  18.90%  51.11% ##########
[ 13, 14) 130129  18.92%  70.04% ##########
[ 14, 15) 106086  15.43%  85.46% ########
[ 15, 15]  99952  14.54% 100.00% ########

Number of training obs by leaf:
Count: 687605 Average: 9.79487 StdDev: 10.2582
Min: 5 Max: 257 Ignored: 0
----------------------------------------------
[   5,  17) 623807  90.72%  90.72% ##########
[  17,  30)  40317   5.86%  96.59% #
[  30,  42)  11791   1.71%  98.30%
[  42,  55)   5563   0.81%  99.11%
[  55,  68)   2509   0.36%  99.47%
[  68,  80)   1172   0.17%  99.64%
[  80,  93)    775   0.11%  99.76%
[  93, 106)    492   0.07%  99.83%
[ 106, 118)    269   0.04%  99.87%
[ 118, 131)    202   0.03%  99.90%
[ 131, 144)    173   0.03%  99.92%
[ 144, 156)    111   0.02%  99.94%
[ 156, 169)     73   0.01%  99.95%
[ 169, 182)     67   0.01%  99.96%
[ 182, 194)     65   0.01%  99.97%
[ 194, 207)    113   0.02%  99.98%
[ 207, 220)     62   0.01%  99.99%
[ 220, 232)     41   0.01% 100.00%
[ 232, 245)      2   0.00% 100.00%
[ 245, 257]      1   0.00% 100.00%
```

```
Attribute in nodes:
	216342 : wrr [NUMERICAL]
	165397 : wrc [CATEGORICAL]
	119478 : bkr [NUMERICAL]
	112101 : bkc [CATEGORICAL]
	51255 : wkc [CATEGORICAL]
	22732 : wkr [NUMERICAL]

Attribute in nodes with depth <= 0:
	205 : bkr [NUMERICAL]
	78 : wkr [NUMERICAL]
	12 : wkc [CATEGORICAL]
	5 : bkc [CATEGORICAL]

Attribute in nodes with depth <= 1:
	328 : bkr [NUMERICAL]
	225 : wkr [NUMERICAL]
	176 : bkc [CATEGORICAL]
	162 : wkc [CATEGORICAL]
	9 : wrr [NUMERICAL]

Attribute in nodes with depth <= 2:
	630 : bkc [CATEGORICAL]
	508 : wkr [NUMERICAL]
	455 : bkr [NUMERICAL]
	399 : wkc [CATEGORICAL]
	96 : wrr [NUMERICAL]
	12 : wrc [CATEGORICAL]

Attribute in nodes with depth <= 3:
	1211 : bkc [CATEGORICAL]
	982 : wkr [NUMERICAL]
	875 : wkc [CATEGORICAL]
	779 : bkr [NUMERICAL]
	411 : wrr [NUMERICAL]
	242 : wrc [CATEGORICAL]

Attribute in nodes with depth <= 5:
	4253 : wrc [CATEGORICAL]
	3484 : bkc [CATEGORICAL]
	3272 : bkr [NUMERICAL]
	2809 : wrr [NUMERICAL]
	2720 : wkc [CATEGORICAL]
	2352 : wkr [NUMERICAL]

Condition type in nodes:
	358552 : HigherCondition
	328753 : ContainsBitmapCondition
```

```
Condition type in nodes with depth <= 0:
        283 : HigherCondition
        17 : ContainsBitmapCondition
Condition type in nodes with depth <= 1:
        562 : HigherCondition
        338 : ContainsBitmapCondition
Condition type in nodes with depth <= 2:
        1059 : HigherCondition
        1041 : ContainsBitmapCondition
Condition type in nodes with depth <= 3:
        2328 : ContainsBitmapCondition
        2172 : HigherCondition
Condition type in nodes with depth <= 5:
        10457 : ContainsBitmapCondition
        8433 : HigherCondition

Training OOB:
        trees: 1, Out-of-bag evaluation: accuracy:0.635458 logloss:13.1394
        trees: 11, Out-of-bag evaluation: accuracy:0.690411 logloss:5.00099
        trees: 21, Out-of-bag evaluation: accuracy:0.730788 logloss:2.67814
        trees: 31, Out-of-bag evaluation: accuracy:0.742316 logloss:1.95862
        trees: 41, Out-of-bag evaluation: accuracy:0.750379 logloss:1.58524
        trees: 51, Out-of-bag evaluation: accuracy:0.753497 logloss:1.37915
        trees: 61, Out-of-bag evaluation: accuracy:0.755768 logloss:1.26819
        trees: 71, Out-of-bag evaluation: accuracy:0.758708 logloss:1.17384
        trees: 81, Out-of-bag evaluation: accuracy:0.75902 logloss:1.11116
        trees: 91, Out-of-bag evaluation: accuracy:0.760624 logloss:1.05555
        trees: 101, Out-of-bag evaluation: accuracy:0.762272 logloss:1.02269
        trees: 111, Out-of-bag evaluation: accuracy:0.764276 logloss:0.992012
        trees: 121, Out-of-bag evaluation: accuracy:0.763875 logloss:0.96005
        trees: 131, Out-of-bag evaluation: accuracy:0.764365 logloss:0.940543
        trees: 141, Out-of-bag evaluation: accuracy:0.764009 logloss:0.928943
        trees: 151, Out-of-bag evaluation: accuracy:0.764365 logloss:0.913852
        trees: 161, Out-of-bag evaluation: accuracy:0.764811 logloss:0.89613
        trees: 171, Out-of-bag evaluation: accuracy:0.765479 logloss:0.881226
        trees: 181, Out-of-bag evaluation: accuracy:0.764944 logloss:0.866638
        trees: 191, Out-of-bag evaluation: accuracy:0.765523 logloss:0.864695
        trees: 201, Out-of-bag evaluation: accuracy:0.764855 logloss:0.848468
        trees: 211, Out-of-bag evaluation: accuracy:0.765434 logloss:0.843513
        trees: 221, Out-of-bag evaluation: accuracy:0.766414 logloss:0.837226
        trees: 231, Out-of-bag evaluation: accuracy:0.765746 logloss:0.833018
        trees: 241, Out-of-bag evaluation: accuracy:0.765924 logloss:0.828863
        trees: 251, Out-of-bag evaluation: accuracy:0.766503 logloss:0.822741
        trees: 261, Out-of-bag evaluation: accuracy:0.766236 logloss:0.819977
        trees: 271, Out-of-bag evaluation: accuracy:0.766503 logloss:0.815393
        trees: 281, Out-of-bag evaluation: accuracy:0.767528 logloss:0.810447
        trees: 291, Out-of-bag evaluation: accuracy:0.767305 logloss:0.804649
        trees: 300, Out-of-bag evaluation: accuracy:0.765702 logloss:0.803169
```

También recibimos información de las variables más importantes, que coinciden con el resultado de EDA (lógico ya que usar randomForest). También podemos ver el número de nodos y de hojas.

Procedemos a probar el árbol con el conjunto test:

[82]:
```
evaluation = model_rf.evaluate(test, return_dict=True)
print()

for name, value in evaluation.items():
    print(f"{name}: {value:.4f}")
```

```
88/88 [==============================] - 1s 6ms/step - loss: 0.0000e+00 -
accuracy: 0.7683

loss: 0.0000
accuracy: 0.7683
```

Obtenemos las predicciones para poder utilizar distintas métricas.

[83]:
```
preds = model_rf.predict(test)
```

[85]:
```
y_test_aux = test_aux["opt rank"]
```

[84]:
```
from sklearn.metrics import confusion_matrix, precision_score, \
f1_score, cohen_kappa_score, recall_score

def compute_metrics_multiclass(y_test, y_pred):
    results=[]
    results.append(precision_score(y_test, np.round(y_pred), average="micro"))
    results.append(recall_score(y_test, np.round(y_pred), average="micro"))
    results.append(f1_score(y_test, np.round(y_pred), average="micro"))
    results.append(cohen_kappa_score(y_test, np.round(y_pred)))
    return results
```

Utilizamos la función argmax para poder obtener la clase predicha en cada caso.

[86]:
```
preds = np.argmax(preds, axis=1)

metrics_rf = compute_metrics_multiclass(y_test_aux, preds)
metrics_rf
```

Los resultados oscilan el 76% en todas las métricas. No son mejores que los obtenidos por las redes neuronales usando datos por defecto.

Podemos realizar una visualización del árbol promedio de los 300, aunque, al ser multiclase, no nos es fácil interpretarlo.

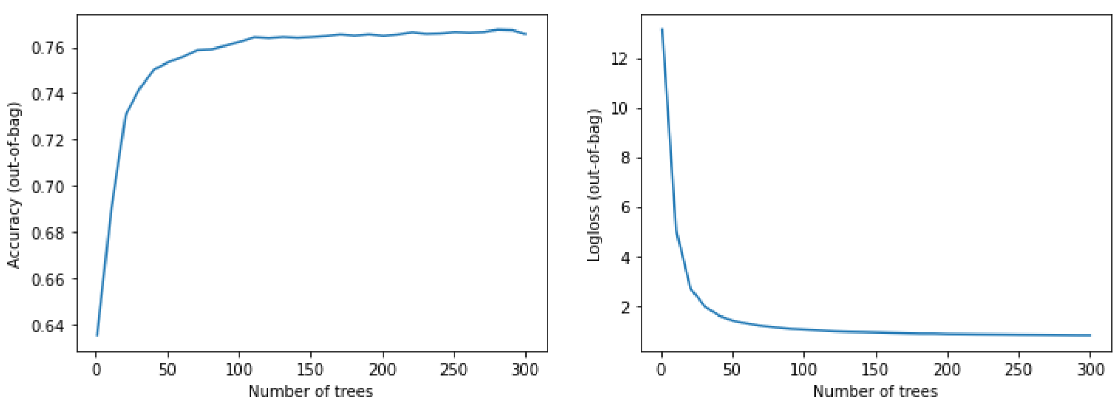
[89]:
```
tfdf.model_plotter.plot_model_in_colab(model_rf, tree_idx=0, max_depth=3)
```

[89]: <IPython.core.display.HTML object>

Podemos ver cómo ha evolucionado el entrenamiento del modelo:

[92]:
```
import matplotlib.pyplot as plt

def make_figure(model):
    logs = model.make_inspector().training_logs()

    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.plot([log.num_trees for log in logs], [log.evaluation.accuracy for log␣
 ↪in logs])
    plt.xlabel("Number of trees")
    plt.ylabel("Accuracy (out-of-bag)")

    plt.subplot(1, 2, 2)
    plt.plot([log.num_trees for log in logs], [log.evaluation.loss for log in␣
 ↪logs])
    plt.xlabel("Number of trees")
    plt.ylabel("Logloss (out-of-bag)")

    plt.show()
```

[93]:
```
make_figure(model_rf)
```



[51]:
```
model_6 = tfdf.keras.GradientBoostedTreesModel(
                                                num_trees=500,␣
 ↪growing_strategy="BEST_FIRST_GLOBAL", max_depth=8)

model_6.compile(metrics=['accuracy'])
with sys_pipes():
```

```python
model_6.fit(x=train)
```

```
350/350 [==============================] - 0s 986us/step

[INFO kernel.cc:746] Start Yggdrasil model training
[INFO kernel.cc:747] Collect training examples
[INFO kernel.cc:392] Number of batches: 350
[INFO kernel.cc:393] Number of examples: 22372
[INFO kernel.cc:769] Dataset:
Number of records: 22372
Number of columns: 7

Number of columns by type:
        CATEGORICAL: 4 (57.1429%)
        NUMERICAL: 3 (42.8571%)

Columns:

CATEGORICAL: 4 (57.1429%)
        0: "bkc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 3878 (17.3342%)
        2: "wkc" CATEGORICAL has-dict vocab-size:5 zero-ood-items most-
frequent:"d" 9698 (43.3488%)
        4: "wrc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 2904 (12.9805%)
        6: "__LABEL" CATEGORICAL integerized vocab-size:19 no-ood-item

NUMERICAL: 3 (42.8571%)
        1: "bkr" NUMERICAL mean:4.44761 min:1 max:8 sd:2.25212
        3: "wkr" NUMERICAL mean:1.85692 min:1 max:4 sd:0.930804
        5: "wrr" NUMERICAL mean:4.50903 min:1 max:8 sd:2.28302

Terminology:
        nas: Number of non-available (i.e. missing) values.
        ood: Out of dictionary.
        manually-defined: Attribute which type is manually defined by the user
i.e. the type was not automatically inferred.
        tokenized: The attribute value is obtained through tokenization.
        has-dict: The attribute is attached to a string dictionary e.g. a
categorical attribute stored as a string.
        vocab-size: Number of unique values.

[INFO kernel.cc:772] Configure learner
[WARNING gradient_boosted_trees.cc:1532] Subsample hyperparameter given but
sampling method does not match.
[WARNING gradient_boosted_trees.cc:1545] GOSS alpha hyperparameter given but
GOSS is disabled.
[WARNING gradient_boosted_trees.cc:1554] GOSS beta hyperparameter given but GOSS
```

```
is disabled.
[WARNING gradient_boosted_trees.cc:1566] SelGB ratio hyperparameter given but
SelGB is disabled.
[INFO kernel.cc:797] Training config:
learner: "GRADIENT_BOOSTED_TREES"
features: "bkc"
features: "bkr"
features: "wkc"
features: "wkr"
features: "wrc"
features: "wrr"
label: "__LABEL"
task: CLASSIFICATION
[yggdrasil_decision_forests.model.gradient_boosted_trees.proto.gradient_boosted_
trees_config] {
  num_trees: 500
  decision_tree {
    max_depth: 8
    min_examples: 5
    in_split_min_examples_check: true
    missing_value_policy: GLOBAL_IMPUTATION
    allow_na_conditions: false
    categorical_set_greedy_forward {
      sampling: 0.1
      max_num_items: -1
      min_item_frequency: 1
    }
    growing_strategy_best_first_global {
    }
    categorical {
      cart {
      }
    }
    num_candidate_attributes_ratio: -1
    axis_aligned_split {
    }
  }
  shrinkage: 0.1
  validation_set_ratio: 0.1
  early_stopping: VALIDATION_LOSS_INCREASE
  early_stopping_num_trees_look_ahead: 30
  l2_regularization: 0
  lambda_loss: 1
  mart {
  }
  adapt_subsample_for_maximum_training_duration: false
  l1_regularization: 0
  use_hessian_gain: false
```

```
  l2_regularization_categorical: 1
}

[INFO kernel.cc:800] Deployment config:

[INFO kernel.cc:837] Train model
[INFO gradient_boosted_trees.cc:480] Default loss set to
MULTINOMIAL_LOG_LIKELIHOOD
[INFO gradient_boosted_trees.cc:1358]   num-trees:1 train-loss:2.365042 train-
accuracy:0.527873 valid-loss:2.392094 valid-accuracy:0.493938
[INFO gradient_boosted_trees.cc:1360]   num-trees:46 train-loss:0.676747 train-
accuracy:0.828047 valid-loss:0.827015 valid-accuracy:0.748990
[INFO gradient_boosted_trees.cc:2506] Early stop of the training because the
validation loss does not decrease anymore. Best valid-loss: 0.353456
[INFO gradient_boosted_trees.cc:319] Truncates the model to 8730 tree(s) i.e.
485  iteration(s).
[INFO gradient_boosted_trees.cc:348] Final model valid-loss:0.353456 valid-
accuracy:0.898518
[INFO kernel.cc:856] Export model in log directory: /tmp/tmpebwx3s8s
[INFO kernel.cc:864] Save model in resources
[INFO kernel.cc:929] Loading model from path
[INFO decision_forest.cc:590] Model loaded with 8730 root(s), 532524 node(s),
and 6 input feature(s).
[INFO abstract_model.cc:876] Engine "GradientBoostedTreesGeneric" built
[INFO kernel.cc:797] Use fast generic engine
```

[52]:
```
model_6.summary()
```

```
Model: "gradient_boosted_trees_model_3"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Total params: 1
Trainable params: 0
Non-trainable params: 1
_________________________________________________________________
Type: "GRADIENT_BOOSTED_TREES"
Task: CLASSIFICATION
Label: "__LABEL"

Input Features (6):
        bkc
        bkr
        wkc
        wkr
        wrc
        wrr
```

```
No weights

Variable Importance: NUM_NODES:
    1. "wrr" 59698.000000 ################
    2. "wrc" 54912.000000 #############
    3. "bkc" 46848.000000 ##########
    4. "bkr" 44814.000000 #########
    5. "wkr" 30969.000000 ##
    6. "wkc" 24656.000000

Variable Importance: NUM_AS_ROOT:
    1. "bkr" 3319.000000 ################
    2. "bkc" 2056.000000 #########
    3. "wkc" 1324.000000 #####
    4. "wkr" 1147.000000 ####
    5. "wrr" 520.000000
    6. "wrc" 364.000000

Variable Importance: SUM_SCORE:
    1. "bkr" 9656.621065 ################
    2. "wrr" 9331.077819 ##############
    3. "bkc" 9151.806027 ##############
    4. "wrc" 8880.009429 #############
    5. "wkr" 6373.709679 ####
    6. "wkc" 4913.474923

Variable Importance: MEAN_MIN_DEPTH:
    1. "__LABEL"  6.632488 ################
    2.     "wkr"  3.988948 #######
    3.     "wkc"  3.665291 ######
    4.     "wrr"  3.489067 #####
    5.     "wrc"  3.066322 ####
    6.     "bkc"  2.004604
    7.     "bkr"  1.786424



Loss: MULTINOMIAL_LOG_LIKELIHOOD
Validation loss value: 0.353456
Number of trees per iteration: 18
Number of trees: 8730
Total number of nodes: 532524

Number of nodes by tree:
Count: 8730 Average: 60.9993 StdDev: 0.0642125
Min: 55 Max: 61 Ignored: 0
----------------------------------------------
[ 55, 56)    1   0.01%   0.01%
```

```
[ 56, 57)    0   0.00%   0.01%
[ 57, 58)    0   0.00%   0.01%
[ 58, 59)    0   0.00%   0.01%
[ 59, 60)    0   0.00%   0.01%
[ 60, 61)    0   0.00%   0.01%
[ 61, 61] 8729  99.99% 100.00% ##########

Depth by leafs:
Count: 270627 Average: 6.63251 StdDev: 1.80387
Min: 1 Max: 8 Ignored: 0
----------------------------------------------
[ 1, 2)   5267   1.95%   1.95%
[ 2, 3)   7396   2.73%   4.68% #
[ 3, 4)   9831   3.63%   8.31% #
[ 4, 5)  14968   5.53%  13.84% #
[ 5, 6)  21665   8.01%  21.85% ##
[ 6, 7)  33646  12.43%  34.28% ###
[ 7, 8)  47522  17.56%  51.84% ####
[ 8, 8] 130332  48.16% 100.00% ##########

Number of training obs by leaf:
Count: 270627 Average: 0 StdDev: 0
Min: 0 Max: 0 Ignored: 0
----------------------------------------------
[ 0, 0] 270627 100.00% 100.00% ##########

Attribute in nodes:
        59698 : wrr [NUMERICAL]
        54912 : wrc [CATEGORICAL]
        46848 : bkc [CATEGORICAL]
        44814 : bkr [NUMERICAL]
        30969 : wkr [NUMERICAL]
        24656 : wkc [CATEGORICAL]

Attribute in nodes with depth <= 0:
        3319 : bkr [NUMERICAL]
        2056 : bkc [CATEGORICAL]
        1324 : wkc [CATEGORICAL]
        1147 : wkr [NUMERICAL]
        520 : wrr [NUMERICAL]
        364 : wrc [CATEGORICAL]

Attribute in nodes with depth <= 1:
        6266 : bkr [NUMERICAL]
        5747 : bkc [CATEGORICAL]
        2700 : wkc [CATEGORICAL]
        2159 : wrc [CATEGORICAL]
        2137 : wrr [NUMERICAL]
```

```
        1914 : wkr [NUMERICAL]

Attribute in nodes with depth <= 2:
        9280 : bkc [CATEGORICAL]
        8643 : bkr [NUMERICAL]
        6258 : wrc [CATEGORICAL]
        5582 : wrr [NUMERICAL]
        4534 : wkc [CATEGORICAL]
        3616 : wkr [NUMERICAL]

Attribute in nodes with depth <= 3:
        14058 : bkc [CATEGORICAL]
        12244 : bkr [NUMERICAL]
        12000 : wrc [CATEGORICAL]
        10733 : wrr [NUMERICAL]
        6999 : wkc [CATEGORICAL]
        6028 : wkr [NUMERICAL]

Attribute in nodes with depth <= 5:
        29795 : wrr [NUMERICAL]
        28561 : wrc [CATEGORICAL]
        26977 : bkc [CATEGORICAL]
        24695 : bkr [NUMERICAL]
        16030 : wkr [NUMERICAL]
        14329 : wkc [CATEGORICAL]

Condition type in nodes:
        135481 : HigherCondition
        126416 : ContainsBitmapCondition
Condition type in nodes with depth <= 0:
        4986 : HigherCondition
        3744 : ContainsBitmapCondition
Condition type in nodes with depth <= 1:
        10606 : ContainsBitmapCondition
        10317 : HigherCondition
Condition type in nodes with depth <= 2:
        20072 : ContainsBitmapCondition
        17841 : HigherCondition
Condition type in nodes with depth <= 3:
        33057 : ContainsBitmapCondition
        29005 : HigherCondition
Condition type in nodes with depth <= 5:
        70520 : HigherCondition
        69867 : ContainsBitmapCondition
```

[53]:
```python
evaluation = model_6.evaluate(test, return_dict=True)
print()

for name, value in evaluation.items():
    print(f"{name}: {value:.4f}")
```

```
89/89 [==============================] - 2s 23ms/step - loss: 0.0000e+00 -
accuracy: 0.8909

loss: 0.0000
accuracy: 0.8909
```

[62]:
```python
model_7 = tfdf.keras.GradientBoostedTreesModel(
    num_trees=500,
    growing_strategy="BEST_FIRST_GLOBAL",
    max_depth=8,
    split_axis="SPARSE_OBLIQUE",
    categorical_algorithm="RANDOM",
    )

model_7.compile(metrics=['accuracy'])
with sys_pipes():
    model_7.fit(x=train)
```

```
350/350 [==============================] - 0s 986us/step

[INFO kernel.cc:746] Start Yggdrasil model training
[INFO kernel.cc:747] Collect training examples
[INFO kernel.cc:392] Number of batches: 350
[INFO kernel.cc:393] Number of examples: 22372
[INFO kernel.cc:769] Dataset:
Number of records: 22372
Number of columns: 7

Number of columns by type:
        CATEGORICAL: 4 (57.1429%)
        NUMERICAL: 3 (42.8571%)

Columns:

CATEGORICAL: 4 (57.1429%)
        0: "bkc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 3878 (17.3342%)
        2: "wkc" CATEGORICAL has-dict vocab-size:5 zero-ood-items most-
frequent:"d" 9698 (43.3488%)
        4: "wrc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 2904 (12.9805%)
        6: "__LABEL" CATEGORICAL integerized vocab-size:19 no-ood-item
```

```
NUMERICAL: 3 (42.8571%)
        1: "bkr" NUMERICAL mean:4.44761 min:1 max:8 sd:2.25212
        3: "wkr" NUMERICAL mean:1.85692 min:1 max:4 sd:0.930804
        5: "wrr" NUMERICAL mean:4.50903 min:1 max:8 sd:2.28302

Terminology:
        nas: Number of non-available (i.e. missing) values.
        ood: Out of dictionary.
        manually-defined: Attribute which type is manually defined by the user
i.e. the type was not automatically inferred.
        tokenized: The attribute value is obtained through tokenization.
        has-dict: The attribute is attached to a string dictionary e.g. a
categorical attribute stored as a string.
        vocab-size: Number of unique values.

[INFO kernel.cc:772] Configure learner
[WARNING gradient_boosted_trees.cc:1532] Subsample hyperparameter given but
sampling method does not match.
[WARNING gradient_boosted_trees.cc:1545] GOSS alpha hyperparameter given but
GOSS is disabled.
[WARNING gradient_boosted_trees.cc:1554] GOSS beta hyperparameter given but GOSS
is disabled.
[WARNING gradient_boosted_trees.cc:1566] SelGB ratio hyperparameter given but
SelGB is disabled.
[INFO kernel.cc:797] Training config:
learner: "GRADIENT_BOOSTED_TREES"
features: "bkc"
features: "bkr"
features: "wkc"
features: "wkr"
features: "wrc"
features: "wrr"
label: "__LABEL"
task: CLASSIFICATION
[yggdrasil_decision_forests.model.gradient_boosted_trees.proto.gradient_boosted_
trees_config] {
  num_trees: 500
  decision_tree {
    max_depth: 8
    min_examples: 5
    in_split_min_examples_check: true
    missing_value_policy: GLOBAL_IMPUTATION
    allow_na_conditions: false
    categorical_set_greedy_forward {
      sampling: 0.1
      max_num_items: -1
      min_item_frequency: 1
```

```
    }
    growing_strategy_best_first_global {
    }
    categorical {
      random {
      }
    }
    num_candidate_attributes_ratio: -1
    sparse_oblique_split {
    }
  }
  shrinkage: 0.1
  validation_set_ratio: 0.1
  early_stopping: VALIDATION_LOSS_INCREASE
  early_stopping_num_trees_look_ahead: 30
  l2_regularization: 0
  lambda_loss: 1
  mart {
  }
  adapt_subsample_for_maximum_training_duration: false
  l1_regularization: 0
  use_hessian_gain: false
  l2_regularization_categorical: 1
}

[INFO kernel.cc:800] Deployment config:

[INFO kernel.cc:837] Train model
[INFO gradient_boosted_trees.cc:480] Default loss set to
MULTINOMIAL_LOG_LIKELIHOOD
[INFO gradient_boosted_trees.cc:1358]   num-trees:1 train-loss:2.374941 train-
accuracy:0.522512 valid-loss:2.397382 valid-accuracy:0.500674
[INFO gradient_boosted_trees.cc:1360]   num-trees:2 train-loss:2.108289 train-
accuracy:0.578109 valid-loss:2.142498 valid-accuracy:0.549169
[INFO gradient_boosted_trees.cc:1360]   num-trees:81 train-loss:0.445374 train-
accuracy:0.904790 valid-loss:0.634584 valid-accuracy:0.805119
[INFO gradient_boosted_trees.cc:1360]   num-trees:175 train-loss:0.236031 train-
accuracy:0.971258 valid-loss:0.472494 valid-accuracy:0.856758
[INFO gradient_boosted_trees.cc:1360]   num-trees:273 train-loss:0.139098 train-
accuracy:0.992008 valid-loss:0.401477 valid-accuracy:0.870678
[INFO gradient_boosted_trees.cc:1360]   num-trees:367 train-loss:0.088143 train-
accuracy:0.998064 valid-loss:0.362643 valid-accuracy:0.887741
[INFO gradient_boosted_trees.cc:2506] Early stop of the training because the
validation loss does not decrease anymore. Best valid-loss: 0.357081
[INFO gradient_boosted_trees.cc:319] Truncates the model to 6858 tree(s) i.e.
381  iteration(s).
[INFO gradient_boosted_trees.cc:348] Final model valid-loss:0.357081 valid-
accuracy:0.889088
```

```
[INFO kernel.cc:856] Export model in log directory: /tmp/tmp7x_7n3sg
[INFO kernel.cc:864] Save model in resources
[INFO kernel.cc:929] Loading model from path
[INFO decision_forest.cc:590] Model loaded with 6858 root(s), 418334 node(s),
and 6 input feature(s).
[INFO abstract_model.cc:876] Engine "GradientBoostedTreesGeneric" built
[INFO kernel.cc:797] Use fast generic engine
```

[63]:
```
print(tfdf.keras.GradientBoostedTreesModel.predefined_hyperparameters())
```

```
[HyperParameterTemplate(name='better_default', version=1,
parameters={'growing_strategy': 'BEST_FIRST_GLOBAL'}, description='A
configuration that is generally better than the default parameters without being
more expensive.'), HyperParameterTemplate(name='benchmark_rank1', version=1,
parameters={'growing_strategy': 'BEST_FIRST_GLOBAL', 'categorical_algorithm':
'RANDOM', 'split_axis': 'SPARSE_OBLIQUE', 'sparse_oblique_normalization':
'MIN_MAX', 'sparse_oblique_num_projections_exponent': 1.0}, description='Top
ranking hyper-parameters on our benchmark slightly modified to run in reasonable
time.')]
```

[59]:
```
model_8 = tfdf.keras.
 ↪GradientBoostedTreesModel(hyperparameter_template="benchmark_rank1")

model_8.compile(metrics=['accuracy'])
with sys_pipes():
    model_8.fit(x=train)
```

```
350/350 [==============================] - 0s 997us/step
```

```
[INFO kernel.cc:746] Start Yggdrasil model training
[INFO kernel.cc:747] Collect training examples
[INFO kernel.cc:392] Number of batches: 350
[INFO kernel.cc:393] Number of examples: 22372
[INFO kernel.cc:769] Dataset:
Number of records: 22372
Number of columns: 7

Number of columns by type:
        CATEGORICAL: 4 (57.1429%)
        NUMERICAL: 3 (42.8571%)

Columns:

CATEGORICAL: 4 (57.1429%)
        0: "bkc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 3878 (17.3342%)
        2: "wkc" CATEGORICAL has-dict vocab-size:5 zero-ood-items most-
frequent:"d" 9698 (43.3488%)
```

```
        4: "wrc" CATEGORICAL has-dict vocab-size:9 zero-ood-items most-
frequent:"h" 2904 (12.9805%)
        6: "__LABEL" CATEGORICAL integerized vocab-size:19 no-ood-item

NUMERICAL: 3 (42.8571%)
        1: "bkr" NUMERICAL mean:4.44761 min:1 max:8 sd:2.25212
        3: "wkr" NUMERICAL mean:1.85692 min:1 max:4 sd:0.930804
        5: "wrr" NUMERICAL mean:4.50903 min:1 max:8 sd:2.28302

Terminology:
        nas: Number of non-available (i.e. missing) values.
        ood: Out of dictionary.
        manually-defined: Attribute which type is manually defined by the user
i.e. the type was not automatically inferred.
        tokenized: The attribute value is obtained through tokenization.
        has-dict: The attribute is attached to a string dictionary e.g. a
categorical attribute stored as a string.
        vocab-size: Number of unique values.

[INFO kernel.cc:772] Configure learner
[WARNING gradient_boosted_trees.cc:1532] Subsample hyperparameter given but
sampling method does not match.
[WARNING gradient_boosted_trees.cc:1545] GOSS alpha hyperparameter given but
GOSS is disabled.
[WARNING gradient_boosted_trees.cc:1554] GOSS beta hyperparameter given but GOSS
is disabled.
[WARNING gradient_boosted_trees.cc:1566] SelGB ratio hyperparameter given but
SelGB is disabled.
[INFO kernel.cc:797] Training config:
learner: "GRADIENT_BOOSTED_TREES"
features: "bkc"
features: "bkr"
features: "wkc"
features: "wkr"
features: "wrc"
features: "wrr"
label: "__LABEL"
task: CLASSIFICATION
[yggdrasil_decision_forests.model.gradient_boosted_trees.proto.gradient_boosted_
trees_config] {
  num_trees: 300
  decision_tree {
    max_depth: 6
    min_examples: 5
    in_split_min_examples_check: true
    missing_value_policy: GLOBAL_IMPUTATION
    allow_na_conditions: false
    categorical_set_greedy_forward {
```

```
      sampling: 0.1
      max_num_items: -1
      min_item_frequency: 1
    }
    growing_strategy_best_first_global {
    }
    categorical {
      random {
      }
    }
    num_candidate_attributes_ratio: -1
    sparse_oblique_split {
      num_projections_exponent: 1
      normalization: MIN_MAX
    }
  }
  shrinkage: 0.1
  validation_set_ratio: 0.1
  early_stopping: VALIDATION_LOSS_INCREASE
  early_stopping_num_trees_look_ahead: 30
  l2_regularization: 0
  lambda_loss: 1
  mart {
  }
  adapt_subsample_for_maximum_training_duration: false
  l1_regularization: 0
  use_hessian_gain: false
  l2_regularization_categorical: 1
}

[INFO kernel.cc:800] Deployment config:

[INFO kernel.cc:837] Train model
[INFO gradient_boosted_trees.cc:480] Default loss set to
MULTINOMIAL_LOG_LIKELIHOOD
[INFO gradient_boosted_trees.cc:1358]   num-trees:1 train-loss:2.433006 train-
accuracy:0.480715 valid-loss:2.460682 valid-accuracy:0.456219
[INFO gradient_boosted_trees.cc:1360]   num-trees:2 train-loss:2.180985 train-
accuracy:0.544850 valid-loss:2.213965 valid-accuracy:0.515492
[INFO gradient_boosted_trees.cc:1360]   num-trees:138 train-loss:0.332551 train-
accuracy:0.937900 valid-loss:0.552413 valid-accuracy:0.821284
[INFO gradient_boosted_trees.cc:1360]   num-trees:279 train-loss:0.157787 train-
accuracy:0.985158 valid-loss:0.415097 valid-accuracy:0.870678
[INFO gradient_boosted_trees.cc:1358]   num-trees:300 train-loss:0.143388 train-
accuracy:0.987540 valid-loss:0.404849 valid-accuracy:0.876066
[INFO gradient_boosted_trees.cc:319] Truncates the model to 5400 tree(s) i.e.
300  iteration(s).
[INFO gradient_boosted_trees.cc:348] Final model valid-loss:0.404849 valid-
```

```
accuracy:0.876066
[INFO kernel.cc:856] Export model in log directory: /tmp/tmpqvidrx75
[INFO kernel.cc:864] Save model in resources
[INFO kernel.cc:929] Loading model from path
[INFO decision_forest.cc:590] Model loaded with 5400 root(s), 329366 node(s),
and 6 input feature(s).
[INFO abstract_model.cc:876] Engine "GradientBoostedTreesGeneric" built
[INFO kernel.cc:797] Use fast generic engine
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function
CoreModel.make_predict_function.<locals>.predict_function_trained at
0x7f242402b430> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```


```
WARNING:tensorflow:6 out of the last 6 calls to <function
CoreModel.make_predict_function.<locals>.predict_function_trained at
0x7f242402b430> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```


[64]:
```
model_8.evaluate(test)
```

```
89/89 [==============================] - 2s 27ms/step - loss: 0.0000e+00 -
accuracy: 0.8571
```

[64]: [0.0, 0.8571428656578064]

[65]:
```
model_7.evaluate(test)
```

```
89/89 [==============================] - 3s 30ms/step - loss: 0.0000e+00 -
accuracy: 0.8830
```

[65]: [0.0, 0.883004903793335]

[69]:
```
np.argmax(model_7.predict(test), axis=1)
```

[69]: array([0, 0, 0, …, 16, 16, 16])

1 SMOTE

[70]:
```
from imblearn.over_sampling import SMOTE
```

[72]:
```
sm = SMOTE(random_state=2)
```

[130]:
```
data.head()
```

[130]:

	wkc	wkr	wrc	wrr	bkc	bkr	opt rank
0	0	1	1	3	2	2	0
1	0	1	2	1	2	2	0
2	0	1	2	1	3	1	0
3	0	1	2	1	3	2	0
4	0	1	2	2	2	1	0

[131]:
```
#We need to convert categorical columns to integer in order to use SMOTE
cat_columns = ['wkc', 'wrc', 'bkc']

data[cat_columns]=data[cat_columns].astype("category")
```

[132]:
```
data[cat_columns]=data[cat_columns].apply(lambda x: x.cat.codes)
```

[133]:
```
data.astype('int64')
```

[133]:

	wkc	wkr	wrc	wrr	bkc	bkr	opt rank
0	0	1	1	3	2	2	0
1	0	1	2	1	2	2	0
2	0	1	2	1	3	1	0
3	0	1	2	1	3	2	0
4	0	1	2	2	2	1	0
…	…	…	…	…	…	…	…
28051	1	1	6	7	4	5	17
28052	1	1	6	7	4	6	17
28053	1	1	6	7	4	7	17
28054	1	1	6	7	5	5	17
28055	1	1	6	7	6	5	17

[28056 rows x 7 columns]

[134]:
```
xsmote, ysmote = sm.fit_resample(data.drop(label, axis=1), data[label])
```

[141]:
```
data_smote = pd.concat([xsmote, ysmote], axis=1)
data_smote.astype('int64')
```

[141]:
```
       wkc  wkr  wrc  wrr  bkc  bkr  opt rank
0        0    1    1    3    2    2         0
1        0    1    2    1    2    2         0
2        0    1    2    1    3    1         0
3        0    1    2    1    3    2         0
4        0    1    2    2    2    1         0
…     …  …  …  …  …  …        …
81949    0    1    5    7    3    2        17
81950    0    1    1    2    4    3        17
81951    0    1    6    6    4    3        17
81952    0    1    7    6    5    3        17
81953    0    1    3    6    5    3        17

[81954 rows x 7 columns]
```

[142]:
```python
train_smote, test_smote = split_dataset(data_smote)
```

[143]:
```python
print("{} examples in training, {} examples for testing.".format(
    len(train_smote), len(test_smote)))
```

```
65571 examples in training, 16383 examples for testing.
```

[144]:
```python
train_smote = tfdf.keras.pd_dataframe_to_tf_dataset(train_smote, label=label, )
test_smote = tfdf.keras.pd_dataframe_to_tf_dataset(test_smote, label=label)
```

[147]:
```python
train_smote
```

[147]:
```
<BatchDataset shapes: ({wkc: (None,), wkr: (None,), wrc: (None,), wrr: (None,),
bkc: (None,), bkr: (None,)}, (None,)), types: ({wkc: tf.int8, wkr: tf.int64,
wrc: tf.int8, wrr: tf.int64, bkc: tf.int8, bkr: tf.int64}, tf.int64)>
```

[146]:
```python
model_1_smote = tfdf.keras.RandomForestModel()

model_1_smote.compile(
                metrics=["accuracy"])

model_1_smote.fit(x=train_smote)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-146-18e6e4ad8e58> in <module>
      4                 metrics=["accuracy"])
      5
----> 6 model_1_smote.fit(x=train_smote)

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow_decision_forests/
↪keras/core.py in fit(self, x, y, callbacks, **kwargs)
```

```
    769
    770     try:
--> 771       history = super(CoreModel, self).fit(
    772           x=x, y=y, epochs=1, callbacks=callbacks, **kwargs)
    773     finally:

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/training.py in fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
↪validation_freq, max_queue_size, workers, use_multiprocessing)
   1181                 _r=1):
   1182               callbacks.on_train_batch_begin(step)
-> 1183               tmp_logs = self.train_function(iterator)
   1184               if data_handler.should_sync:
   1185                 context.async_wait()

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
↪def_function.py in __call__(self, *args, **kwds)
    887
    888       with OptionalXlaContext(self._jit_compile):
--> 889         result = self._call(*args, **kwds)
    890
    891       new_tracing_count = self.experimental_get_tracing_count()

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
↪def_function.py in _call(self, *args, **kwds)
    931       # This is the first call of __call__, so we have to initialize.
    932       initializers = []
--> 933       self._initialize(args, kwds, add_initializers_to=initializers)
    934     finally:
    935       # At this point we know that the initialization is complete (or␣
↪less

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
↪def_function.py in _initialize(self, args, kwds, add_initializers_to)
    761     self._graph_deleter = FunctionDeleter(self._lifted_initializer_grap )
    762     self._concrete_stateful_fn = (
--> 763         self._stateful_fn.
↪_get_concrete_function_internal_garbage_collected(  # pylint:␣
↪disable=protected-access
    764             *args, **kwds))
    765

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
↪function.py in _get_concrete_function_internal_garbage_collected(self, *args,␣
↪**kwargs)
   3048       args, kwargs = None, None
```

```
   3049     with self._lock:
-> 3050       graph_function, _ = self._maybe_define_function(args, kwargs)
   3051     return graph_function
   3052 

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
 ↪function.py in _maybe_define_function(self, args, kwargs)
   3442 
   3443           self._function_cache.missed.add(call_context_key)
-> 3444           graph_function = self._create_graph_function(args, kwargs)
   3445           self._function_cache.primary[cache_key] = graph_function
   3446 

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
 ↪function.py in _create_graph_function(self, args, kwargs,␣
 ↪override_flat_arg_shapes)
   3277     arg_names = base_arg_names + missing_arg_names
   3278     graph_function = ConcreteFunction(
-> 3279         func_graph_module.func_graph_from_py_func(

   3280             self._name,
   3281             self._python_function,

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/framework/
 ↪func_graph.py in func_graph_from_py_func(name, python_func, args, kwargs,␣
 ↪signature, func_graph, autograph, autograph_options, add_control_dependencies,␣
 ↪arg_names, op_return_value, collections, capture_by_value,␣
 ↪override_flat_arg_shapes)
    997         _, original_func = tf_decorator.unwrap(python_func)
    998 
--> 999       func_outputs = python_func(*func_args, **func_kwargs)
   1000 
   1001       # invariant: `func_outputs` contains only Tensors,␣
 ↪CompositeTensors,

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/eager/
 ↪def_function.py in wrapped_fn(*args, **kwds)
    670         # the function a weak reference to itself to avoid a reference␣
 ↪cycle.
    671         with OptionalXlaContext(compile_with_xla):
--> 672           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    673         return out
    674 

~/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/python/framework/
 ↪func_graph.py in wrapper(*args, **kwargs)
    984           except Exception as e:  # pylint:disable=broad-except
    985             if hasattr(e, "ag_error_metadata"):
--> 986               raise e.ag_error_metadata.to_exception(e)
```

```
    987              else:
    988                raise

ValueError: in user code:

    /home/antonio/anaconda3/envs/tf2.5/lib/python3.8/site-packages/tensorflow/
 ↪python/keras/engine/training.py:855 train_function  *
        return step_function(self, iterator)
    /home/antonio/anaconda3/envs/tf2.5/lib/python3.8/site-packages/
 ↪tensorflow_decision_forests/keras/core.py:646 train_step  *
        normalized_semantic_inputs = tf_core.normalize_inputs(semantic_inputs)
    /home/antonio/anaconda3/envs/tf2.5/lib/python3.8/site-packages/
 ↪tensorflow_decision_forests/tensorflow/core.py:255 normalize_inputs  *
        raise ValueError(

    ValueError: Non supported tensor dtype <dtype: 'int8'> for semantic Semantic.
 ↪CATEGORICAL of feature wkc
```