AdvDSI-A2-beer-train-catboost-1

March 20, 2022

1 AdvDSI - Assignment 2: Multi-Class Classification - Beer Style Predictor - Train - Catboost

Train a machine learning model (using sklearn) or a custom neural networks (using pytorch) that will

accurately predict a type of beer based on some users' rating criterias such as appearance, aroma, palate or taste.

You will also need to build a web app and deploy it online (using Heroku) in order to serve your model for real time predictions.

Student Name: Nathan Fragar

Student No. : 93087548

Week: 6

Date: 20MAR2022

Notebook: This notebook is investigating the effectiveness of the Catboost Algorithm for predicting beer type

1.1 Install Pre-requisites

1. Set Google Colab Notebook to use GPU Hardware accelerator

   `Edit > Notebook Settings > Hardware Accelerator = GPU`

2. Install Catboost that can run on a Google Colab GPU
3. Import Libraries

[ ]:
```
cd /notebooks
```

```
/notebooks
```

[ ]:
```
ls
```

```
catboost-hyperopt-log.txt  notebooks/
catboost_info/             nvidia-docker_1.0.1-1_amd64.deb
data/                      nvidia-docker_1.0.1-1_amd64.deb.1
file_save.ipynb            nvidia-docker_1.0.1-1_amd64.deb.2
```

```
model/                          nvidia-docker_1.0.1-1_amd64.deb.3
models/
```

[ ]:
```
! pip install catboost
! pip install hpsklearn
! pip install colorama
! pip install shap
! pip install lime
```

[ ]:
```
# Import packages and libraries

# Pandas Numpy
import pandas as pd
import os
import numpy as np
np.set_printoptions(precision=4)
from scipy import stats
from joblib import dump
from joblib import load


# Hyperparameter tuning
from hpsklearn import HyperoptEstimator, any_classifier
import hyperopt
from hyperopt import Trials, STATUS_OK, tpe, hp, fmin

# Performance
from sklearn.metrics import accuracy_score
import sklearn
import shap

# Visualisation
import colorama
from lime.lime_tabular import LimeTabularExplainer

# Task: Import Scaler
from sklearn.preprocessing import MinMaxScaler

# Algorithm - Catboos
import catboost
from catboost import *
from catboost import CatBoostClassifier
from catboost import Pool

# Task: Import Pipeline
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
```

```
WARN: OMP_NUM_THREADS=None =>
… If you are using openblas if you are using openblas set OMP_NUM_THREADS=1 or
risk subprocess calls hanging indefinitely
```

[**Task**] Change Notebook directory

[ ]:
```
cd ..
```

```
/notebooks
```

[ ]:
```
pwd
```

[ ]:
```
'/notebooks'
```

[ ]:
```
ls -al
```

```
total 8882
drwxrwxrwx 8 root root      12 Mar 17 07:21 ./
drwxr-xr-x 1 root root    4096 Mar 17 07:19 ../
-rw-r--r-- 1 root root   14697 Mar 15 12:13 catboost-hyperopt-log.txt
drwxr-xr-x 5 root root       7 Mar 13 06:54 catboost_info/
drwxr-xr-x 6 root root       4 Mar 14 09:09 data/
-rw-r--r-- 1 root root   11498 Mar 13 11:37 file_save.ipynb
drwxr-xr-x 2 root root       1 Mar 13 11:37 .ipynb_checkpoints/
drwxr-xr-x 2 root root       0 Mar 13 06:10 model/
drwxr-xr-x 2 root root       1 Mar 14 10:46 models/
drwxr-xr-x 3 root root      14 Mar 17 07:23 notebooks/
-rw-r--r-- 1 root root 2266050 Dec  8 03:30 nvidia-docker_1.0.1-1_amd64.deb
-rw-r--r-- 1 root root 2266050 Dec  8 03:30 nvidia-docker_1.0.1-1_amd64.deb.1
-rw-r--r-- 1 root root 2266050 Dec  8 03:30 nvidia-docker_1.0.1-1_amd64.deb.2
-rw-r--r-- 1 root root 2266050 Dec  8 03:30 nvidia-docker_1.0.1-1_amd64.deb.3
```

[**Task**] Load Datasets

[ ]:
```
file_path_X_train = "/notebooks/data/processed/standard_10/X_train.npy"
file_path_y_train = "/notebooks/data/processed/standard_10/y_train.npy"
file_path_X_val = "/notebooks/data/processed/standard_10/X_val.npy"
file_path_y_val = "/notebooks/data/processed/standard_10/y_val.npy"
file_path_X_test = "/notebooks/data/processed/standard_10/X_test.npy"
file_path_y_test = "/notebooks/data/processed/standard_10/y_test.npy"

# Load files into df_training and df_validation Numpy Arrays
X_train = np.load(file_path_X_train, allow_pickle=True)
y_train = np.load(file_path_y_train, allow_pickle=True)
X_val = np.load(file_path_X_val, allow_pickle=True)
y_val = np.load(file_path_y_val, allow_pickle=True)
X_test = np.load(file_path_X_test, allow_pickle=True)
y_test = np.load(file_path_y_test, allow_pickle=True)
```

[ ]:
```
X_train.shape
```

[ ]: (95196, 6)

[ ]:
```
print(X_train)
```

```
[[1.9200e+02 1.2116e-01 7.5000e-01 9.0000e-01 8.7500e-01 8.7500e-01]
 [3.9200e+02 9.6897e-02 6.2500e-01 8.0000e-01 7.5000e-01 7.5000e-01]
 [8.1100e+02 9.6897e-02 7.5000e-01 7.0000e-01 7.5000e-01 7.5000e-01]
 …
 [7.8400e+02 1.3677e-01 7.5000e-01 8.0000e-01 7.5000e-01 8.7500e-01]
 [7.6000e+01 9.5164e-02 5.0000e-01 8.0000e-01 7.5000e-01 7.5000e-01]
 [5.9000e+02 9.5164e-02 7.5000e-01 8.0000e-01 6.2500e-01 7.5000e-01]]
```

[ ]:
```
print(y_train)
```

```
[ 12 102  18 …  17  81  60]
```

1.2 2. Base Line Model

Calculate baseline accuracy

[ ]:
```
vals, counts = np.unique(y_train, return_counts=True)

y_mode_num = np.argwhere(counts == np.max(counts))

y_mode = vals[y_mode_num].flatten().tolist()

print(y_mode)
```

```
[12]
```

[ ]:
```
y_shape = (len(y_train), 1)
y_base = np.full(y_shape, y_mode)
```

[ ]:
```
accuracy_score(y_train, y_base)
```

[ ]: 0.0741102567334762

Our null accuracy is 7.4%.

If we predict 'American IPA', we'll be correct 7.4% of the time.

1.3 Train Catboost Algorithm

[ ]:
```
# cat_features  = Brewery Name
#cat_features = [0]
```

[ ]:
```
train_data = Pool(
                    data=X_train
                  , label = y_train
                 )

eval_dataset = Pool(
                    data=X_val
                  , label = y_val
                 )

test_dataset = Pool(
                    data=X_test
                  , label = y_test
                 )
```

Multiclass CLassifier - Eval Metrics * Logloss * F1 * Accuracy

[ ]:
```
from catboost.utils import get_gpu_device_count
print('I see %i GPU devices' % get_gpu_device_count())
```

```
I see 1 GPU devices
```

[ ]:
```
# Calculate Catboost - Check GPU
from catboost import CatBoostClassifier
model = CatBoostClassifier(
      iterations=30
    , learning_rate=0.2
    , depth = 10
    , loss_function='MultiClass'
    , task_type="GPU"
)
```

[ ]:
```
model.fit(train_data)
print('Model is fitted: ' + str(model.is_fitted()))
print('Model params:')
print(model.get_params())
```

```
0:      learn: 3.9936089        total: 1.25s    remaining: 36.3s
1:      learn: 3.7843863        total: 2.46s    remaining: 34.5s
2:      learn: 3.6580814        total: 3.7s     remaining: 33.3s
3:      learn: 3.5689413        total: 4.91s    remaining: 31.9s
4:      learn: 3.3994371        total: 6.1s     remaining: 30.5s
5:      learn: 3.2970490        total: 7.29s    remaining: 29.2s
6:      learn: 3.2017400        total: 8.45s    remaining: 27.8s
7:      learn: 3.1371128        total: 9.62s    remaining: 26.4s
8:      learn: 3.0876737        total: 10.8s    remaining: 25.2s
9:      learn: 3.0468875        total: 12s      remaining: 24.1s
10:     learn: 3.0079986        total: 13.3s    remaining: 22.9s
```

```
11:     learn: 2.9839148        total: 14.5s    remaining: 21.8s
12:     learn: 2.9622450        total: 15.7s    remaining: 20.5s
13:     learn: 2.9444324        total: 16.8s    remaining: 19.2s
14:     learn: 2.8531418        total: 18s      remaining: 18s
15:     learn: 2.7854542        total: 19.2s    remaining: 16.8s
16:     learn: 2.7347752        total: 20.4s    remaining: 15.6s
17:     learn: 2.6728168        total: 21.6s    remaining: 14.4s
18:     learn: 2.6461608        total: 22.7s    remaining: 13.1s
19:     learn: 2.6258852        total: 24s      remaining: 12s
20:     learn: 2.6034597        total: 25.2s    remaining: 10.8s
21:     learn: 2.5686606        total: 26.4s    remaining: 9.59s
22:     learn: 2.5407114        total: 27.6s    remaining: 8.4s
23:     learn: 2.5006843        total: 28.8s    remaining: 7.19s
24:     learn: 2.4865793        total: 30s      remaining: 6s
25:     learn: 2.4691173        total: 31.2s    remaining: 4.8s
26:     learn: 2.4546391        total: 32.5s    remaining: 3.61s
27:     learn: 2.4434615        total: 33.9s    remaining: 2.42s
28:     learn: 2.4293220        total: 35.1s    remaining: 1.21s
29:     learn: 2.4066810        total: 36.3s    remaining: 0us
Model is fitted: True
Model params:
{'iterations': 30, 'learning_rate': 0.2, 'depth': 10, 'loss_function':
'MultiClass', 'task_type': 'GPU'}
```

[ ]:
```
# Get predicted classes
preds_class = model.predict(eval_dataset)
```

[ ]:
```
# Get predicted probabilities for each class
preds_proba = model.predict_proba(eval_dataset)
```

[ ]:
```
# Get predicted RawFormulaVal
preds_raw = model.predict(eval_dataset,
                          prediction_type='RawFormulaVal')
```

[ ]:
```
print(model.get_best_score())
```

```
{'learn': {'MultiClass': 2.4066810186352368}}
```

[ ]:
```
# Get Model with Best Iteration Score
print(model.get_params())
```

```
{'iterations': 30, 'learning_rate': 0.2, 'depth': 10, 'loss_function':
'MultiClass', 'task_type': 'GPU'}
```

1.4 Model Comparison

[ ]:
```
# Create pools for catboost configuration

train_data1 = Pool(
                    data=X_train
                  , label = y_train
                  , cat_features = cat_features
                 )

eval_dataset1 = Pool(
                    data=X_val
                  , label = y_val
                  , cat_features = cat_features
                 )

test_dataset1 = Pool(
                    data=X_test
                  , label = y_test
                  , cat_features = cat_features
                 )
```

[ ]:
```
# Check learning rate between 0.01 and 0.7 to assist with understanding which␣
 ↪end to work through
from catboost import CatBoostClassifier
model1 = CatBoostClassifier(
      iterations=100
    , learning_rate=0.7
    , depth = 10
    , loss_function='MultiClass'
    , random_seed=8
    , task_type="GPU"
    , train_dir='data/processed/catboost_10/learing_rate_0.7'
)

model2 = CatBoostClassifier(
      iterations=100
    , learning_rate=0.01
    , depth = 10
    , loss_function='MultiClass'
    , random_seed=8
    , task_type="GPU"
    , train_dir='data/processed/catboost_10/learing_rate_0.01'
)
```

[ ]:
```
model1.fit(train_data1)
print('Model is fitted: ' + str(model1.is_fitted()))
```

```
print('Model params:')
print(model1.get_params())
```

```
0:      learn: 7.4549167        total: 1.24s    remaining: 2m 2s
1:      learn: 122.7575318      total: 2.48s    remaining: 2m 1s
2:      learn: 132.1236607      total: 3.7s     remaining: 1m 59s
3:      learn: 196.7856633      total: 4.94s    remaining: 1m 58s
4:      learn: 225.1364553      total: 6.17s    remaining: 1m 57s
5:      learn: 255.8165889      total: 7.39s    remaining: 1m 55s
6:      learn: 268.8214001      total: 8.65s    remaining: 1m 54s
7:      learn: 249.8037943      total: 9.9s     remaining: 1m 53s
8:      learn: 227.2701374      total: 11.2s    remaining: 1m 52s
9:      learn: 204.7495273      total: 12.4s    remaining: 1m 51s
10:     learn: 225.2398210      total: 13.6s    remaining: 1m 50s
11:     learn: 235.8132905      total: 14.9s    remaining: 1m 49s
12:     learn: 222.9627715      total: 16.2s    remaining: 1m 48s
13:     learn: 211.6460986      total: 17.4s    remaining: 1m 47s
14:     learn: 211.6242279      total: 18.7s    remaining: 1m 45s
15:     learn: 211.4810496      total: 20s      remaining: 1m 44s
16:     learn: 217.6911845      total: 21.2s    remaining: 1m 43s
17:     learn: 217.0951721      total: 22.5s    remaining: 1m 42s
18:     learn: 200.5099584      total: 23.7s    remaining: 1m 41s
19:     learn: 206.0497920      total: 25s      remaining: 1m 39s
20:     learn: 199.0605067      total: 26.2s    remaining: 1m 38s
21:     learn: 197.0240346      total: 27.5s    remaining: 1m 37s
22:     learn: 244.8074709      total: 28.7s    remaining: 1m 36s
23:     learn: 231.8423883      total: 30s      remaining: 1m 34s
24:     learn: 232.8920123      total: 31.2s    remaining: 1m 33s
25:     learn: 211.9735493      total: 32.5s    remaining: 1m 32s
26:     learn: 227.9637800      total: 33.7s    remaining: 1m 31s
27:     learn: 250.7089163      total: 34.9s    remaining: 1m 29s
28:     learn: 251.0724820      total: 36.2s    remaining: 1m 28s
29:     learn: 222.7207446      total: 37.5s    remaining: 1m 27s
30:     learn: 213.2188958      total: 38.8s    remaining: 1m 26s
31:     learn: 230.6727594      total: 40s      remaining: 1m 25s
32:     learn: 230.4263625      total: 41.3s    remaining: 1m 23s
33:     learn: 231.7014160      total: 42.5s    remaining: 1m 22s
34:     learn: 217.1020001      total: 43.8s    remaining: 1m 21s
35:     learn: 242.8046346      total: 45s      remaining: 1m 20s
36:     learn: 225.4065927      total: 46.3s    remaining: 1m 18s
37:     learn: 221.7152191      total: 47.5s    remaining: 1m 17s
38:     learn: 208.8975797      total: 48.8s    remaining: 1m 16s
39:     learn: 235.1795874      total: 50s      remaining: 1m 14s
40:     learn: 230.7047565      total: 51.2s    remaining: 1m 13s
41:     learn: 225.1719820      total: 52.5s    remaining: 1m 12s
42:     learn: 217.4918904      total: 53.7s    remaining: 1m 11s
43:     learn: 221.6341443      total: 55s      remaining: 1m 9s
```

```
44:     learn: 235.6794613      total: 56.2s    remaining: 1m 8s
45:     learn: 224.5661162      total: 57.5s    remaining: 1m 7s
46:     learn: 198.2115005      total: 58.7s    remaining: 1m 6s
47:     learn: 191.0419135      total: 59.9s    remaining: 1m 4s
48:     learn: 211.6378419      total: 1m 1s    remaining: 1m 3s
49:     learn: 213.4753351      total: 1m 2s    remaining: 1m 2s
50:     learn: 195.4090928      total: 1m 3s    remaining: 1m 1s
51:     learn: 185.9306063      total: 1m 4s    remaining: 59.9s
52:     learn: 183.1858481      total: 1m 6s    remaining: 58.6s
53:     learn: 212.7396109      total: 1m 7s    remaining: 57.3s
54:     learn: 203.0108618      total: 1m 8s    remaining: 56.1s
55:     learn: 189.8099920      total: 1m 9s    remaining: 54.8s
56:     learn: 175.9719526      total: 1m 11s   remaining: 53.6s
57:     learn: 213.5586789      total: 1m 12s   remaining: 52.3s
58:     learn: 212.5330896      total: 1m 13s   remaining: 51.1s
59:     learn: 212.3646792      total: 1m 14s   remaining: 49.8s
60:     learn: 191.5893735      total: 1m 15s   remaining: 48.6s
61:     learn: 209.7313122      total: 1m 17s   remaining: 47.3s
62:     learn: 214.0484894      total: 1m 18s   remaining: 46s
63:     learn: 205.1408463      total: 1m 19s   remaining: 44.8s
64:     learn: 195.3578932      total: 1m 20s   remaining: 43.6s
65:     learn: 192.0887012      total: 1m 22s   remaining: 42.3s
66:     learn: 198.8175554      total: 1m 23s   remaining: 41.1s
67:     learn: 185.8608555      total: 1m 24s   remaining: 39.8s
68:     learn: 198.1352368      total: 1m 25s   remaining: 38.6s
69:     learn: 189.1728434      total: 1m 27s   remaining: 37.3s
70:     learn: 194.4369511      total: 1m 28s   remaining: 36.1s
71:     learn: 189.0998571      total: 1m 29s   remaining: 34.8s
72:     learn: 207.5759696      total: 1m 30s   remaining: 33.6s
73:     learn: 194.7083701      total: 1m 32s   remaining: 32.3s
74:     learn: 188.2541703      total: 1m 33s   remaining: 31.1s
75:     learn: 187.9687382      total: 1m 34s   remaining: 29.8s
76:     learn: 215.5023110      total: 1m 35s   remaining: 28.6s
77:     learn: 217.6172108      total: 1m 36s   remaining: 27.3s
78:     learn: 206.0416824      total: 1m 38s   remaining: 26.1s
79:     learn: 186.3416110      total: 1m 39s   remaining: 24.8s
80:     learn: 186.2162066      total: 1m 40s   remaining: 23.6s
81:     learn: 221.5595613      total: 1m 41s   remaining: 22.4s
82:     learn: 214.2646330      total: 1m 43s   remaining: 21.1s
83:     learn: 199.0184251      total: 1m 44s   remaining: 19.9s
84:     learn: 181.6188916      total: 1m 45s   remaining: 18.6s
85:     learn: 209.9688852      total: 1m 46s   remaining: 17.4s
86:     learn: 204.3869070      total: 1m 48s   remaining: 16.1s
87:     learn: 206.1898819      total: 1m 49s   remaining: 14.9s
88:     learn: 188.4594101      total: 1m 50s   remaining: 13.7s
89:     learn: 205.6380310      total: 1m 51s   remaining: 12.4s
90:     learn: 203.7495483      total: 1m 52s   remaining: 11.2s
91:     learn: 197.5630489      total: 1m 54s   remaining: 9.93s
```

```
92:     learn: 186.4439682      total: 1m 55s   remaining: 8.69s
93:     learn: 194.4401866      total: 1m 56s   remaining: 7.45s
94:     learn: 197.3033951      total: 1m 57s   remaining: 6.2s
95:     learn: 209.9979411      total: 1m 59s   remaining: 4.96s
96:     learn: 203.1254675      total: 2m       remaining: 3.72s
97:     learn: 196.5569982      total: 2m 1s    remaining: 2.48s
98:     learn: 209.0934703      total: 2m 2s    remaining: 1.24s
99:     learn: 201.9806294      total: 2m 4s    remaining: 0us
Model is fitted: True
Model params:
{'iterations': 100, 'learning_rate': 0.7, 'depth': 10, 'loss_function':
'MultiClass', 'random_seed': 8, 'train_dir': 'learing_rate_0.7', 'task_type':
'GPU'}
```

[ ]:
```
model2.fit(train_data1)
print('Model is fitted: ' + str(model2.is_fitted()))
print('Model params:')
print(model2.get_params())
```

```
0:      learn: 4.5908317        total: 1.22s    remaining: 2m 1s
1:      learn: 4.5465878        total: 2.44s    remaining: 1m 59s
2:      learn: 4.5092825        total: 3.7s     remaining: 1m 59s
3:      learn: 4.4739400        total: 4.94s    remaining: 1m 58s
4:      learn: 4.4395070        total: 6.12s    remaining: 1m 56s
5:      learn: 4.4086181        total: 7.34s    remaining: 1m 55s
6:      learn: 4.3779741        total: 8.49s    remaining: 1m 52s
7:      learn: 4.3497268        total: 9.65s    remaining: 1m 50s
8:      learn: 4.3234931        total: 10.9s    remaining: 1m 49s
9:      learn: 4.2993058        total: 12.1s    remaining: 1m 49s
10:     learn: 4.2756651        total: 13.3s    remaining: 1m 47s
11:     learn: 4.2553268        total: 14.6s    remaining: 1m 47s
12:     learn: 4.2341715        total: 15.8s    remaining: 1m 45s
13:     learn: 4.2137472        total: 17s      remaining: 1m 44s
14:     learn: 4.1947807        total: 18.3s    remaining: 1m 43s
15:     learn: 4.1765982        total: 19.5s    remaining: 1m 42s
16:     learn: 4.1587343        total: 20.7s    remaining: 1m 41s
17:     learn: 4.1414499        total: 21.9s    remaining: 1m 39s
18:     learn: 4.1250561        total: 23.1s    remaining: 1m 38s
19:     learn: 4.1087004        total: 24.3s    remaining: 1m 37s
20:     learn: 4.0934618        total: 25.5s    remaining: 1m 36s
21:     learn: 4.0786345        total: 26.7s    remaining: 1m 34s
22:     learn: 4.0641190        total: 27.9s    remaining: 1m 33s
23:     learn: 4.0501787        total: 29.1s    remaining: 1m 32s
24:     learn: 4.0372193        total: 30.3s    remaining: 1m 31s
25:     learn: 4.0239808        total: 31.5s    remaining: 1m 29s
26:     learn: 4.0116355        total: 32.8s    remaining: 1m 28s
27:     learn: 3.9982355        total: 34s      remaining: 1m 27s
28:     learn: 3.9862596        total: 35.2s    remaining: 1m 26s
```

```
29:     learn: 3.9749993        total: 36.4s    remaining: 1m 24s
30:     learn: 3.9635167        total: 37.6s    remaining: 1m 23s
31:     learn: 3.9522951        total: 38.8s    remaining: 1m 22s
32:     learn: 3.9411648        total: 40s      remaining: 1m 21s
33:     learn: 3.9306352        total: 41.1s    remaining: 1m 19s
34:     learn: 3.9209844        total: 42.4s    remaining: 1m 18s
35:     learn: 3.9111153        total: 43.7s    remaining: 1m 17s
36:     learn: 3.9009458        total: 44.8s    remaining: 1m 16s
37:     learn: 3.8912172        total: 46s      remaining: 1m 15s
38:     learn: 3.8815805        total: 47.3s    remaining: 1m 13s
39:     learn: 3.8718739        total: 48.4s    remaining: 1m 12s
40:     learn: 3.8624765        total: 49.6s    remaining: 1m 11s
41:     learn: 3.8530391        total: 50.8s    remaining: 1m 10s
42:     learn: 3.8445310        total: 52.1s    remaining: 1m 9s
43:     learn: 3.8367972        total: 53.3s    remaining: 1m 7s
44:     learn: 3.8283673        total: 54.5s    remaining: 1m 6s
45:     learn: 3.8207150        total: 55.7s    remaining: 1m 5s
46:     learn: 3.8124869        total: 56.9s    remaining: 1m 4s
47:     learn: 3.8050076        total: 58.1s    remaining: 1m 2s
48:     learn: 3.7972578        total: 59.3s    remaining: 1m 1s
49:     learn: 3.7895520        total: 1m       remaining: 1m
50:     learn: 3.7811745        total: 1m 1s    remaining: 59.2s
51:     learn: 3.7742418        total: 1m 2s    remaining: 58s
52:     learn: 3.7667818        total: 1m 4s    remaining: 56.8s
53:     learn: 3.7590392        total: 1m 5s    remaining: 55.6s
54:     learn: 3.7520858        total: 1m 6s    remaining: 54.3s
55:     learn: 3.7452289        total: 1m 7s    remaining: 53.1s
56:     learn: 3.7382949        total: 1m 8s    remaining: 51.9s
57:     learn: 3.7310224        total: 1m 9s    remaining: 50.7s
58:     learn: 3.7244051        total: 1m 11s   remaining: 49.4s
59:     learn: 3.7181585        total: 1m 12s   remaining: 48.3s
60:     learn: 3.7121347        total: 1m 13s   remaining: 47s
61:     learn: 3.7052929        total: 1m 14s   remaining: 45.8s
62:     learn: 3.6987659        total: 1m 15s   remaining: 44.6s
63:     learn: 3.6929946        total: 1m 17s   remaining: 43.4s
64:     learn: 3.6873513        total: 1m 18s   remaining: 42.2s
65:     learn: 3.6817080        total: 1m 19s   remaining: 41s
66:     learn: 3.6758674        total: 1m 20s   remaining: 39.8s
67:     learn: 3.6700492        total: 1m 21s   remaining: 38.6s
68:     learn: 3.6650319        total: 1m 23s   remaining: 37.4s
69:     learn: 3.6598537        total: 1m 24s   remaining: 36.2s
70:     learn: 3.6550485        total: 1m 25s   remaining: 35s
71:     learn: 3.6495687        total: 1m 26s   remaining: 33.8s
72:     learn: 3.6443456        total: 1m 28s   remaining: 32.6s
73:     learn: 3.6388074        total: 1m 29s   remaining: 31.3s
74:     learn: 3.6336883        total: 1m 30s   remaining: 30.1s
75:     learn: 3.6288444        total: 1m 31s   remaining: 28.9s
76:     learn: 3.6239998        total: 1m 32s   remaining: 27.7s
```

```
77:     learn: 3.6196147        total: 1m 34s   remaining: 26.6s
78:     learn: 3.6148253        total: 1m 35s   remaining: 25.3s
79:     learn: 3.6100765        total: 1m 36s   remaining: 24.1s
80:     learn: 3.6056426        total: 1m 37s   remaining: 22.9s
81:     learn: 3.6007415        total: 1m 38s   remaining: 21.7s
82:     learn: 3.5955345        total: 1m 40s   remaining: 20.5s
83:     learn: 3.5910506        total: 1m 41s   remaining: 19.3s
84:     learn: 3.5862746        total: 1m 42s   remaining: 18.1s
85:     learn: 3.5815945        total: 1m 43s   remaining: 16.9s
86:     learn: 3.5771372        total: 1m 44s   remaining: 15.7s
87:     learn: 3.5727719        total: 1m 46s   remaining: 14.5s
88:     learn: 3.5683166        total: 1m 47s   remaining: 13.3s
89:     learn: 3.5640642        total: 1m 48s   remaining: 12s
90:     learn: 3.5599346        total: 1m 49s   remaining: 10.8s
91:     learn: 3.5557370        total: 1m 50s   remaining: 9.64s
92:     learn: 3.5515286        total: 1m 52s   remaining: 8.43s
93:     learn: 3.5478303        total: 1m 53s   remaining: 7.23s
94:     learn: 3.5394006        total: 1m 54s   remaining: 6.03s
95:     learn: 3.5303558        total: 1m 55s   remaining: 4.82s
96:     learn: 3.5218946        total: 1m 56s   remaining: 3.62s
97:     learn: 3.5133379        total: 1m 58s   remaining: 2.41s
98:     learn: 3.5051177        total: 1m 59s   remaining: 1.21s
99:     learn: 3.4963424        total: 2m       remaining: 0us
Model is fitted: True
Model params:
{'iterations': 100, 'learning_rate': 0.01, 'depth': 10, 'loss_function':
'MultiClass', 'random_seed': 8, 'train_dir': 'learing_rate_0.01', 'task_type':
'GPU'}
```

[ ]:
```
! pip install ipywidgets
```

[ ]:
```
! jupyter nbextension enable --py widgetsnbextension
```

```
Enabling notebook extension jupyter-js-widgets/extension…
      - Validating: OK
```

[ ]:
```
from catboost import MetricVisualizer
```

Visualise two Models

[ ]:
```
MetricVisualizer(['data/processed/catboost_10/learing_rate_0.01', 'data/
↪processed/catboost_10/learing_rate_0.7']).start()
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
```

Issue Visualisation did not work

1.5 Optimise Catboost with HyperOpt

[ ]:
```
# Create pools for processing catboost using HyperOpt
train_data4 = Pool(
                    data=X_train
                  , label = y_train
                 )

eval_dataset4 = Pool(
                    data=X_val
                  , label = y_val
                 )

test_dataset4 = Pool(
                    data=X_test
                  , label = y_test
                )
```

[ ]:
```
N_HYPEROPT_PROBES = 60
HYPEROPT_ALGO = tpe.suggest
```

[ ]:
```
def get_catboost_params(space):
    params = dict()
    params['learning_rate'] = space['learning_rate']
    params['depth'] = int(space['depth'])
    params['l2_leaf_reg'] = space['l2_leaf_reg']
    return params
```

[ ]:
```
obj_call_count = 0
cur_best_loss = np.inf
log_writer = open( 'catboost-hyperopt-log.txt', 'w' )
```

[ ]:
```
def objective(space):
    global obj_call_count, cur_best_loss

    obj_call_count += 1

    print('\nCatBoost objective call #{} cur_best_loss={:7.5f}'.
 ↪format(obj_call_count,cur_best_loss) )

    params = get_catboost_params(space)

    sorted_params = sorted(space.items(), key=lambda z: z[0])
    params_str = str.join(' ', ['{}={}'.format(k, v) for k, v in sorted_params])
    print('Params: {}'.format(params_str) )

    model = CatBoostClassifier( iterations=100,
```

```
                                learning_rate=params['learning_rate'],
                                depth=int(params['depth']),
                                loss_function='MultiClass',
                                use_best_model=True,
                                task_type="GPU",
                                #eval_metric='F1',
                                l2_leaf_reg=params['l2_leaf_reg'],
                                early_stopping_rounds=30,
                                verbose=False
                                )

    model.fit(train_data4, eval_set=eval_dataset4, verbose=False)
    nb_trees = model.tree_count_

    print('nb_trees={}'.format(nb_trees))

    y_pred = model.predict_proba(eval_dataset4)
    test_loss = sklearn.metrics.log_loss(eval_dataset4.get_label(), y_pred)
    acc = sklearn.metrics.accuracy_score(eval_dataset4.get_label(), np.
 ↪argmax(y_pred, axis=1))
    #auc = sklearn.metrics.roc_auc_score(eval_dataset4.get_label(), y_pred[:,1])

    log_writer.write('loss={:<7.5f} acc={} Params:{} nb_trees={}\n'.
 ↪format(test_loss, acc, params_str, nb_trees ))
    log_writer.flush()

    if test_loss<cur_best_loss:
        cur_best_loss = test_loss
        print(colorama.Fore.GREEN + 'NEW BEST LOSS={}'.format(cur_best_loss) +␣
 ↪colorama.Fore.RESET)


    return{'loss':test_loss, 'status': STATUS_OK }
```

[ ]:
```
space = {
        'depth': hp.quniform("depth", 1, 10, 2),
        'learning_rate' : hp.quniform('learning_rate', 0.1, 0.45, 0.05),
        'l2_leaf_reg': hp.uniform('l2_leaf_reg', 3, 8),
       }

trials = Trials()
best = hyperopt.fmin(fn=objective,
                     space=space,
                     algo=HYPEROPT_ALGO,
                     max_evals=N_HYPEROPT_PROBES,
                     trials=trials,
                     verbose=True)
```

```
print('-'*50)
print('The best params:')
print( best )
print('\n\n')
```

```
CatBoost objective call #61 cur_best_loss=2.35771
Params: depth=8.0 l2_leaf_reg=3.327477819460049 learning_rate=0.25
nb_trees=100
NEW BEST LOSS=2.2274805505619084

CatBoost objective call #62 cur_best_loss=2.22748
Params: depth=6.0 l2_leaf_reg=7.789753896297718
learning_rate=0.15000000000000002
nb_trees=100

CatBoost objective call #63 cur_best_loss=2.22748
Params: depth=4.0 l2_leaf_reg=3.4586038967773276 learning_rate=0.25
nb_trees=100

CatBoost objective call #64 cur_best_loss=2.22748
Params: depth=6.0 l2_leaf_reg=4.418050006473141 learning_rate=0.25
nb_trees=100

CatBoost objective call #65 cur_best_loss=2.22748
Params: depth=8.0 l2_leaf_reg=4.638180461008128
learning_rate=0.35000000000000003
nb_trees=1

CatBoost objective call #66 cur_best_loss=2.22748
Params: depth=2.0 l2_leaf_reg=7.879502582896789 learning_rate=0.25
nb_trees=100

CatBoost objective call #67 cur_best_loss=2.22748
Params: depth=4.0 l2_leaf_reg=7.396828127449793 learning_rate=0.1
nb_trees=100

CatBoost objective call #68 cur_best_loss=2.22748
Params: depth=8.0 l2_leaf_reg=5.755033379517931 learning_rate=0.4
nb_trees=1

CatBoost objective call #69 cur_best_loss=2.22748
Params: depth=4.0 l2_leaf_reg=3.0043102867912994
learning_rate=0.35000000000000003
nb_trees=2
```

```
CatBoost objective call #70 cur_best_loss=2.22748
Params: depth=2.0 l2_leaf_reg=6.173102118754071 learning_rate=0.4
nb_trees=1

CatBoost objective call #71 cur_best_loss=2.22748
Params: depth=2.0 l2_leaf_reg=6.775164986776307 learning_rate=0.2
nb_trees=100

CatBoost objective call #72 cur_best_loss=2.22748
Params: depth=8.0 l2_leaf_reg=7.183987588984721
learning_rate=0.35000000000000003
nb_trees=1

CatBoost objective call #73 cur_best_loss=2.22748
Params: depth=4.0 l2_leaf_reg=5.726049396776741
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #74 cur_best_loss=2.22748
Params: depth=4.0 l2_leaf_reg=5.050578999643852
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #75 cur_best_loss=2.22748
Params: depth=10.0 l2_leaf_reg=3.3519049047952993 learning_rate=0.25
nb_trees=100
NEW BEST LOSS=2.129656636850949

CatBoost objective call #76 cur_best_loss=2.12966
Params: depth=8.0 l2_leaf_reg=4.1859492497200055 learning_rate=0.25
nb_trees=100

CatBoost objective call #77 cur_best_loss=2.12966
Params: depth=4.0 l2_leaf_reg=3.157694022590875
learning_rate=0.15000000000000002
nb_trees=100

CatBoost objective call #78 cur_best_loss=2.12966
Params: depth=8.0 l2_leaf_reg=7.345989987643113
learning_rate=0.15000000000000002
nb_trees=100

CatBoost objective call #79 cur_best_loss=2.12966
Params: depth=4.0 l2_leaf_reg=3.755866957813675 learning_rate=0.4
nb_trees=1

CatBoost objective call #80 cur_best_loss=2.12966
Params: depth=8.0 l2_leaf_reg=6.064032462991701 learning_rate=0.45
```

```
nb_trees=1

CatBoost objective call #81 cur_best_loss=2.12966
Params: depth=10.0 l2_leaf_reg=3.82102385505783 learning_rate=0.2
nb_trees=100

CatBoost objective call #82 cur_best_loss=2.12966
Params: depth=10.0 l2_leaf_reg=3.8856781035357235 learning_rate=0.2
nb_trees=100

CatBoost objective call #83 cur_best_loss=2.12966
Params: depth=10.0 l2_leaf_reg=5.151957543438876 learning_rate=0.2
nb_trees=100

CatBoost objective call #84 cur_best_loss=2.12966
Params: depth=10.0 l2_leaf_reg=3.737879871572038 learning_rate=0.2
nb_trees=100

CatBoost objective call #85 cur_best_loss=2.12966
Params: depth=10.0 l2_leaf_reg=4.794487941186591 learning_rate=0.1
nb_trees=100

CatBoost objective call #86 cur_best_loss=2.12966
Params: depth=10.0 l2_leaf_reg=4.2004928271442585
learning_rate=0.30000000000000004
nb_trees=100
NEW BEST LOSS=2.1252383064960614

CatBoost objective call #87 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=4.214905831606052
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #88 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=5.287913189884847
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #89 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=3.3125923391175967
learning_rate=0.35000000000000003
nb_trees=1

CatBoost objective call #90 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=3.035289623381385
learning_rate=0.30000000000000004
nb_trees=2
```

```
CatBoost objective call #91 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=3.511483040535479 learning_rate=0.25
nb_trees=100

CatBoost objective call #92 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=4.192405881104776
learning_rate=0.35000000000000003
nb_trees=2

CatBoost objective call #93 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=4.721112290316173
learning_rate=0.15000000000000002
nb_trees=100

CatBoost objective call #94 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=4.506545652908024 learning_rate=0.45
nb_trees=1

CatBoost objective call #95 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=3.5223703985380506 learning_rate=0.25
nb_trees=100

CatBoost objective call #96 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=4.1345914865966495 learning_rate=0.25
nb_trees=100

CatBoost objective call #97 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=4.869820521623543
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #98 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=5.536880857648674 learning_rate=0.4
nb_trees=100

CatBoost objective call #99 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=3.9779306510885792
learning_rate=0.35000000000000003
nb_trees=1

CatBoost objective call #100 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=3.2708578295910744
learning_rate=0.15000000000000002
nb_trees=100

CatBoost objective call #101 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=4.432509350693938 learning_rate=0.25
nb_trees=100
```

```
CatBoost objective call #102 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=6.661169097255667 learning_rate=0.2
nb_trees=100

CatBoost objective call #103 cur_best_loss=2.12524
Params: depth=2.0 l2_leaf_reg=3.0565996937185176
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #104 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=3.570820980083205 learning_rate=0.1
nb_trees=100

CatBoost objective call #105 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=5.369474410643529
learning_rate=0.35000000000000003
nb_trees=2

CatBoost objective call #106 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=4.9132128519574 learning_rate=0.4
nb_trees=1

CatBoost objective call #107 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=6.063445821306359 learning_rate=0.25
nb_trees=100

CatBoost objective call #108 cur_best_loss=2.12524
Params: depth=2.0 l2_leaf_reg=4.578629842399955 learning_rate=0.25
nb_trees=100

CatBoost objective call #109 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=7.639878357607344
learning_rate=0.35000000000000003
nb_trees=100

CatBoost objective call #110 cur_best_loss=2.12524
Params: depth=4.0 l2_leaf_reg=4.341910428255893
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #111 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=6.374622859727744 learning_rate=0.4
nb_trees=1

CatBoost objective call #112 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=3.9838181995161284 learning_rate=0.2
nb_trees=100
```

```
CatBoost objective call #113 cur_best_loss=2.12524
Params: depth=8.0 l2_leaf_reg=5.71132088463541 learning_rate=0.2
nb_trees=100

CatBoost objective call #114 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=5.051006405360694 learning_rate=0.45
nb_trees=1

CatBoost objective call #115 cur_best_loss=2.12524
Params: depth=6.0 l2_leaf_reg=3.6595591240263805
learning_rate=0.15000000000000002
nb_trees=100

CatBoost objective call #116 cur_best_loss=2.12524
Params: depth=4.0 l2_leaf_reg=3.3834497107105888
learning_rate=0.30000000000000004
nb_trees=100

CatBoost objective call #117 cur_best_loss=2.12524
Params: depth=10.0 l2_leaf_reg=3.2476472009101376 learning_rate=0.25
nb_trees=100
NEW BEST LOSS=2.1222088878962806

CatBoost objective call #118 cur_best_loss=2.12221
Params: depth=8.0 l2_leaf_reg=3.978366241716321
learning_rate=0.35000000000000003
nb_trees=1

CatBoost objective call #119 cur_best_loss=2.12221
Params: depth=10.0 l2_leaf_reg=3.167499017890776 learning_rate=0.1
nb_trees=100

CatBoost objective call #120 cur_best_loss=2.12221
Params: depth=2.0 l2_leaf_reg=6.978684680541214 learning_rate=0.25
nb_trees=100
100%|      | 60/60 [18:13<00:00, 18.22s/trial, best loss:
2.1222088878962806]
--------------------------------------------------
The best params:
{'depth': 10.0, 'l2_leaf_reg': 3.2476472009101376, 'learning_rate': 0.25}
```

[ ]:
```
print("Best: ", best)
```

```
Best:  {'border_count': 103.13861460449897, 'depth': 6.0, 'l2_leaf_reg':
```

```
5.493351474911527, 'learning_rate': 0.30000000000000004}
```

[ ]:
```
# Create best model 5
   model5 = CatBoostClassifier( iterations=100,
                                learning_rate=0.25,
                                depth=6,
                                loss_function='MultiClass',
                                use_best_model=True,
                                task_type="GPU",
                                #eval_metric='F1',
                                l2_leaf_reg=5,
                                early_stopping_rounds=30,
                                #border_count=125,
                                verbose=False,
                                train_dir='data/processed/catboost_10/model5'
                                )
```

[ ]:
```
# Fit and evaluate model
model5.fit(train_data4, eval_set=eval_dataset4, verbose=False)

nb_trees = model5.tree_count_

print('nb_trees={}'.format(nb_trees))

y_pred = model5.predict_proba(eval_dataset4)
test_loss = sklearn.metrics.log_loss(eval_dataset4.get_label(), y_pred)
acc = sklearn.metrics.accuracy_score(eval_dataset4.get_label(), np.
 ↪argmax(y_pred, axis=1))
#auc = sklearn.metrics.roc_auc_score(eval_dataset4.get_label(), y_pred[:,1])

log_writer.write('loss={:<7.5f} acc={} nb_trees={}\n'.format(test_loss, acc,␣
 ↪nb_trees ))
log_writer.flush()
```

```
nb_trees=100
```

[ ]:
```
test_loss
```

[ ]:
```
2.4601216032586684
```

[ ]:
```
model5.get_feature_importance(data=None,
                        reference_data=None,
                        type=EFstrType.FeatureImportance,
                        prettified=True,
                        thread_count=-1,
                        verbose=False)
```

[ ]:

	Feature Id	Importances
0	1	63.192964
1	0	24.629285
2	2	6.342161
3	3	4.579032
4	5	0.674004
5	4	0.582554

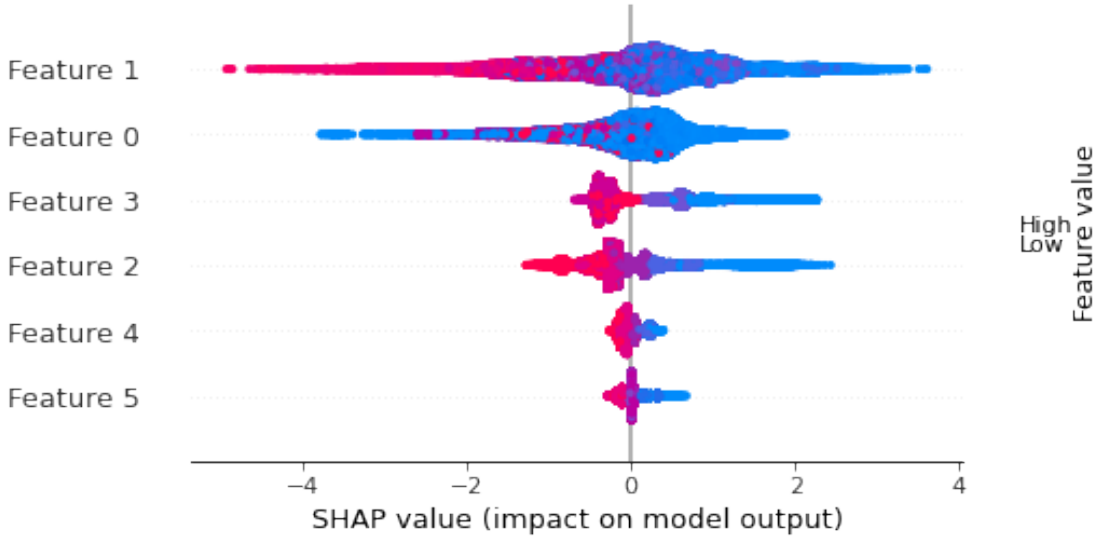
Feature Importance * 1 - beer_abv (63.2) * 0 - brewery_id (24.6) * 2 - review_aroma (6.3) * 3 - review_appearance (4.6) * 5 - review_taste (0.7) * 4 - review_palate (0.6)

[ ]:
```
shap.initjs()
```

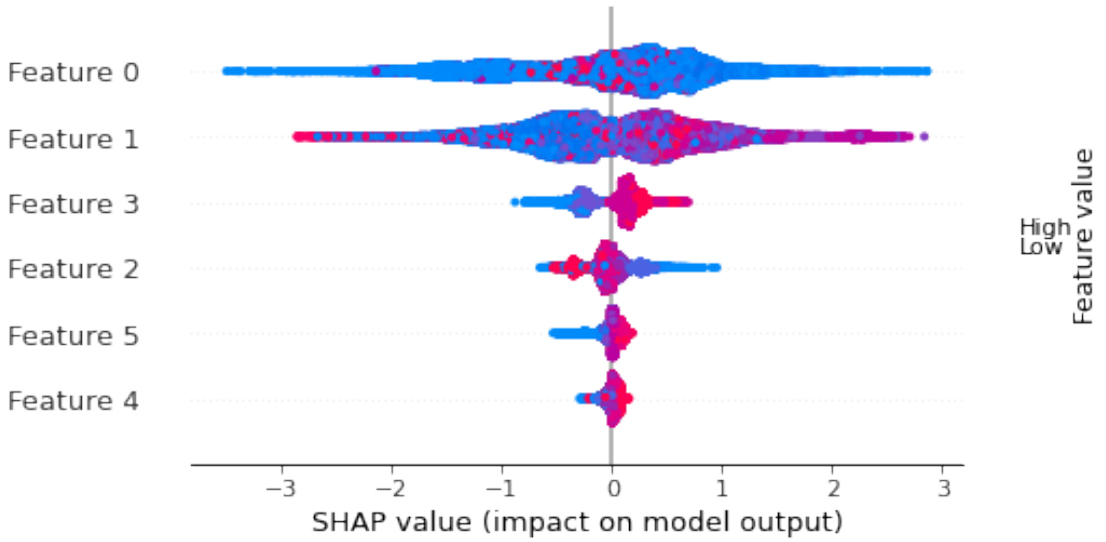
```
<IPython.core.display.HTML object>
```

[ ]:
```
explainer = shap.TreeExplainer(model5)
shap_values = explainer.shap_values(Pool(X_train, y_train))
```

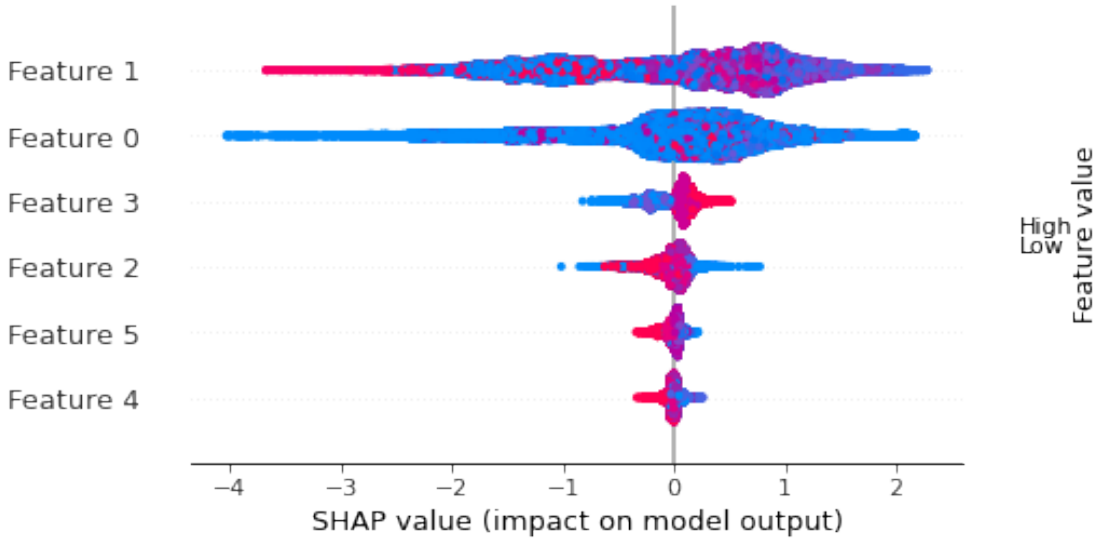
[ ]:
```
# visualize the first prediction's explanation
shap.summary_plot(shap_values[1], X_train)
```



[ ]:
```
# visualize the first prediction's explanation
shap.summary_plot(shap_values[0], X_train)
```



[ ]:
```
# visualize the first prediction's explanation
shap.summary_plot(shap_values[2], X_train)
```



[ ]:
```
from sklearn.inspection import permutation_importance
```

[ ]:
```
r = permutation_importance(
    model5, X_train, y_train,
    n_repeats=30,
    random_state=8
)
```

[ ]:
```
for i in r.importances_mean.argsort()[::-1]:
     print(f"{i}: {r.importances_mean[i]:.5f}")
```

```
1: 0.33468
0: 0.24136
3: 0.01892
2: 0.01683
5: 0.00317
4: 0.00180
```

[ ]:
```
dump(model5,'models/catboost_model5.joblib')
```

[ ]:
```
['models/catboost_model5.joblib']
```

[ ]:
```
model5.predict(test_dataset4,
    prediction_type='Class',
    ntree_start=0,
    ntree_end=0,
    thread_count=-1,
    verbose=None)
```

[ ]:
```
array([[ 9],
       [36],
       [99],
       …,
       [ 9],
       [ 1],
       [12]])
```

[ ]:
```
y_pred = model5.predict_proba(test_dataset4)
test_loss = sklearn.metrics.log_loss(test_dataset4.get_label(), y_pred)
acc = sklearn.metrics.accuracy_score(test_dataset4.get_label(), np.
 ↪argmax(y_pred, axis=1))
```

[ ]:
```
test_loss
```

[ ]:
```
2.4457597540904583
```

[ ]:
```
acc
```

[ ]:
```
0.39980461979642645
```

Test Model

[ ]:
```
# 11
df_test_record = [13014,0.121900,0.750,1.0,1.000,0.875]
```

[ ]:
```
model5.predict(df_test_record,prediction_type='Class',verbose=1)
```

[ ]:
```
array([12])
```

[ ]:
```
# 14
df_test_record_kolsh = [743,0.096897,0.625,0.8,0.625,0.625]
```

[ ]:
```
# Kolsh
model5.predict(df_test_record_kolsh,prediction_type='Class')
```

[ ]:
```
array([17])
```

[ ]:
```
#89
df_test_record_belgian = [11031,0.176634,0.750,0.9,1.000,0.750]
model5.predict(df_test_record_belgian,prediction_type='Class')
```

[ ]:
```
array([89])
```

[ ]:
```
# 25
df_test_record_weitbier = [694,0.190501,0.875,0.8,0.750,0.750]
model5.predict(df_test_record_weitbier,prediction_type='Class')
```

[ ]:
```
array([25])
```

1.6 Build Pipeline

Assess building pipeline to deploy to Fast API platform

Min Max Scaling did not lend the pipeline to deployment on the Fast API model

This was due to not being able to minmax scale.

[ ]:
```
df_train_pipeline = pd.DataFrame(X_train, columns =␣
↪['brewery_id','beer_abv','review_aroma','review_appearance','review_palate','review_taste']
df_val_pipeline = pd.DataFrame(X_val, columns =␣
↪['brewery_id','beer_abv','review_aroma','review_appearance','review_palate','review_taste']
df_test_pipeline = pd.DataFrame(X_test, columns =␣
↪['brewery_id','beer_abv','review_aroma','review_appearance','review_palate','review_taste']
```

[ ]:
```
df_train_pipeline['brewery_id'] = df_train_pipeline['brewery_id'].astype(int)
df_val_pipeline['brewery_id'] = df_val_pipeline['brewery_id'].astype(int)
df_test_pipeline['brewery_id'] = df_test_pipeline['brewery_id'].astype(int)
```

[ ]:
```
df_train_pipeline.describe()
```

[ ]:

	brewery_id	beer_abv	review_aroma	review_appearance	\
count	95196.000000	95196.000000	95196.000000	95196.000000	
mean	3166.500630	0.121878	0.683827	0.768432	
std	5607.113775	0.039352	0.174153	0.122968	
min	1.000000	0.000693	0.000000	0.200000	

```
25%      143.000000      0.091697      0.625000           0.700000
50%      433.000000      0.115271      0.750000           0.800000
75%     2432.000000      0.145432      0.750000           0.800000
max    27927.000000      0.710522      1.000000           1.000000

       review_palate  review_taste
count    95196.00000  95196.000000
mean         0.68618      0.698657
std          0.17053      0.182448
min          0.00000      0.000000
25%          0.62500      0.625000
50%          0.75000      0.750000
75%          0.75000      0.875000
max          1.00000      1.000000
```

[ ]:
```
df_train_pipeline.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95196 entries, 0 to 95195
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   brewery_id         95196 non-null  int64
 1   beer_abv           95196 non-null  float64
 2   review_aroma       95196 non-null  float64
 3   review_appearance  95196 non-null  float64
 4   review_palate      95196 non-null  float64
 5   review_taste       95196 non-null  float64
dtypes: float64(5), int64(1)
memory usage: 4.4 MB
```

[ ]:
```
df_train_target = pd.DataFrame(y_train, columns = ['beer_style_cat'])
df_eval_target = pd.DataFrame(y_val, columns = ['beer_style_cat'])
df_test_target = pd.DataFrame(y_test, columns = ['beer_style_cat'])
```

[ ]:
```
train_data6 = Pool(
                    data=df_train_pipeline
                  , label = df_train_target
                 )

eval_dataset6 = Pool(
                    data=df_val_pipeline
                  , label = df_eval_target
                 )

test_dataset6 = Pool(
                    data=df_test_pipeline
```

```
                  , label = df_test_target
                )
```

[ ]:
```
model_pipeline = CatBoostClassifier( iterations=100,
                                    learning_rate=0.25,
                                    depth=6,
                                    loss_function='MultiClass',
                                    l2_leaf_reg=5,
                                    early_stopping_rounds=30,
                                    verbose=False
                                    )
```

[ ]:
```
# Load saved catboost model into cb_model
cb_model_saved = load('models/catboost_model5.joblib')
```

[ ]:
```
cb_num_cols =␣
 ↪['beer_abv','review_aroma','review_appearance','review_palate','review_taste']
```

[ ]:
```
MinMaxScaleFrame(df_train_5438_65, minmax_scale_cols, inplace=True)
```

```
cols:  ['beer_abv', 'review_palate', 'review_aroma', 'review_appearance',
'review_taste']
```

[ ]:
```
   brewery_id  beer_abv  review_palate  review_aroma  review_appearance  \
0        5438  0.086655          0.125          0.25                0.5

   review_taste
0         0.125
```

[ ]:
```
df_train_5438_65.head()
```

[ ]:
```
   brewery_id  beer_abv  review_palate  review_aroma  review_appearance  \
0        5438       5.0            1.5           2.0                2.5

   review_taste  beer_abv_std  beer_abv_scaled  review_palate_std  \
0           1.5      0.086655              5.0              0.125

   review_palate_scaled  review_aroma_std  review_aroma_scaled  \
0                   1.5              0.25                  2.0

   review_appearance_std  review_appearance_scaled  review_taste_std  \
0                    0.5                       2.5             0.125

   review_taste_scaled
0                  1.5
```

[ ]:
```
# Task: Create a Pipeline called num_transformer with one step that contains␣
↪MinMaxScaler
num_transformer = Pipeline(
    steps=[
        ('scaler', MinMaxScaler())
    ]
)
```

[ ]:
```
# Task: Create a ColumnTransformer called preprocessor with 1 steps containing␣
↪num_transformer that will be applied to num_cols
preprocessor = ColumnTransformer(
    transformers=[
        ('num_cols', num_transformer, cb_num_cols)
    ]
)
```

[ ]:
```
# cb_pipe
cb_pipe = Pipeline(
    steps=[
        ('catboost', model_pipeline)
    ]
)
```

[ ]:
```
cb_pipe.fit(df_train_pipeline,df_train_target)
```

[ ]:
```
Pipeline(steps=[('catboost',
                 <catboost.core.CatBoostClassifier object at 0x7f2c043805e0>)])
```

[ ]:
```
df_beer_min_max_scale = pd.read_csv('data/reference/beer_min_max_scale.csv',␣
↪index_col='col')
df_beer_style = pd.read_csv('data/reference/beer_style.csv')
df_beer_brewery = pd.read_csv('data/reference/breweries.csv')
```

[ ]:
```
df_beer_min_max_scale.head()
```

[ ]:

col	min	max
review_aroma	1	5.0
review_appearance	0	5.0
review_palate	1	5.0
review_taste	1	5.0
beer_abv	0	57.7

[ ]:
```
minmax_scale_cols =␣
↪['beer_abv','review_palate','review_aroma','review_appearance','review_taste']
```

```
[ ]: def MinMaxScaleFrame(df, cols, inplace=True):

         print('cols: ', cols)

         for col in cols:
             x_min = df_beer_min_max_scale.loc[col,'min']
             x_max = df_beer_min_max_scale.loc[col,'max']
             col_std = col + '_std'
             col_scaled = col + '_scaled'

             if inplace==True:
                 df[col] = (df[col] - x_min)/(x_max - x_min)
             else:
                 df[col_std] = (df[col] - x_min)/(x_max - x_min)

         return df
```

```
[ ]: # Vecchio Birraio (5438) - Hefeweizen (65)
     df_train_5438_65 = pd.DataFrame([[5438,5.0,1.5,2.0,2.5,1.5]], columns =␣
      ↪['brewery_id','beer_abv','review_palate','review_aroma','review_appearance','review_taste']
     df_train_5438_65.head()
```

```
[ ]:    brewery_id  beer_abv  review_palate  review_aroma  review_appearance  \
     0        5438       5.0            1.5           2.0                2.5

        review_taste
     0           1.5
```

Example Record: Russian Imperial Stout (89) from Caldera Brewing Company (1075)

brewery_name,brewery_name_cat,beer_abv,review_palate,review_aroma,review_appearance,review_taste,bee

Caldera Brewing Company,1480,8.8,4.0,4.0,3.5,4.0,Russian Imperial Stout

```
[ ]: # Caldera Brewing Company (1075) - Russian Imperial Stout (89)
     test_record_type_1480_89 = [[1480,8.8,4.0,4.0,3.5,4.0]]
     test_record_type_1480_89
```

```
[ ]: [[1480, 8.8, 4.0, 4.0, 3.5, 4.0]]
```

```
[ ]: df_train_1480_89 = pd.DataFrame(test_record_type_1480_89, columns =␣
      ↪['brewery_id','beer_abv','review_palate','review_aroma','review_appearance','review_taste']
```

```
[ ]: df_train_1480_89.head()
```

```
[ ]:    brewery_id  beer_abv  review_palate  review_aroma  review_appearance  \
     0        1480       8.8            4.0           4.0                3.5
```

```
   review_taste
0           4.0
```

[ ]:
```
cb_pipe.predict(df_train_1480_89)
```

[ ]:
```
array([[11]])
```

[ ]:
```
df_train_1480_89 = MinMaxScaleFrame(df_train_5438_65, minmax_scale_cols,␣
↪inplace=True)
```

```
cols:  ['beer_abv', 'review_palate', 'review_aroma', 'review_appearance',
'review_taste']
```

[ ]:
```
cb_pipe.predict(df_train_1480_89)
```

[ ]:
```
array([[77]])
```

[ ]:
```
cb_model_saved.predict(df_train_1480_89)
```

[ ]:
```
array([[77]])
```

[ ]:
```
# Vecchio Birraio (5438) - Hefeweizen (65)
df_train_5438_65 = pd.DataFrame([[5438,5.0,1.5,2.0,2.5,1.5]], columns =␣
↪['brewery_id','beer_abv','review_palate','review_aroma','review_appearance','review_taste']
df_train_5438_65.head()
```

[ ]:
```
   brewery_id  beer_abv  review_palate  review_aroma  review_appearance  \
0        5438       5.0            1.5           2.0                2.5

   review_taste
0           1.5
```

[ ]:
```
cb_pipe.predict(df_train_5438_65)
```

[ ]:
```
array([[11]])
```

[ ]:
```
df_train_5438_65 = MinMaxScaleFrame(df_train_5438_65, minmax_scale_cols,␣
↪inplace=True)
```

```
cols:  ['beer_abv', 'review_palate', 'review_aroma', 'review_appearance',
'review_taste']
```

[ ]:
```
cb_pipe.predict(df_train_5438_65)
```

[ ]:
```
array([[53]])
```

```
[ ]: cb_model_saved.predict(df_train_5438_65)
```

```
[ ]: array([[53]])
```

```
[ ]: # Brasserie d'Orval S.A. (911) - Hefeweizen (65)
df_train_911_24 = pd.DataFrame([[911,6.9,2.0,2.0,3.0,2.5]], columns =␣
  ↪['brewery_id','beer_abv','review_palate','review_aroma','review_appearance','review_taste']
df_train_911_24['brewery_id'] = df_train_911_24['brewery_id'].astype(int)
df_train_911_24.head()
```

```
[ ]:    brewery_id  beer_abv  review_palate  review_aroma  review_appearance  \
    0         911       6.9            2.0           2.0                3.0

       review_taste
    0           2.5
```

```
[ ]: cb_pipe.predict(df_train_911_24)
```

```
[ ]: array([[11]])
```

```
[ ]: cb_model_saved.predict(df_train_911_24)
```

```
[ ]: array([[11]])
```

```
[ ]: df_test_record_weitbier = [694,0.190501,0.875,0.8,0.750,0.750]
```

```
[ ]: # 25
cb_model_saved.predict(df_test_record_weitbier,prediction_type='Class')
```

```
[ ]: array([25])
```

```
[ ]: cb_model_saved.predict(df_train_911_24)
```

```
[ ]: array([[11]])
```

```
[ ]: # 392,0.09689720922170218,0.625,0.8,0.75,0.75,102
df_train_392_102 = pd.DataFrame([[392,0.09689720922170218,0.625,0.8,0.75,0.
  ↪75]], columns =␣
  ↪['brewery_id','beer_abv','review_palate','review_aroma','review_appearance','review_taste']
df_train_392_102['brewery_id'] = df_train_392_102['brewery_id'].astype(int)
```

```
[ ]: # 102: Winter Warmer (102) - Predicted: American Porter (17)
cb_model_saved.predict(df_train_392_102,prediction_type='Class')
```

```
[ ]: array([[17]])
```

[ ]:
```
# 102: Winter Warmer (102) - Predicted: American Pale Ale (APA) (14)
cb_pipe.predict(df_train_392_102)
```

[ ]:
```
array([[14]])
```

[ ]:
```
# 700,0.12116484659386374,0.75,0.8,0.625,0.75,14
df_train_700_14 = pd.DataFrame([[700,0.12116484659386374,0.75,0.8,0.625,0.75]],␣
 ↪columns =␣
 ↪['brewery_id','beer_abv','review_palate','review_aroma','review_appearance','review_taste']
df_train_700_14['brewery_id'] = df_train_700_14['brewery_id'].astype(int)
```

[ ]:
```
# Actual: American Pale Ale (APA) (14) - Predicted: American IPA (12)
cb_model_saved.predict(df_train_700_14,prediction_type='Class')
```

[ ]:
```
array([[12]])
```

[ ]:
```
# Actual: American Pale Ale (APA) (14) - Predicted: American IPA (12)
cb_pipe.predict(df_train_700_14)
```

[ ]:
```
array([[12]])
```