BoostModels

January 28, 2022

[17]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import ModelClass

from matplotlib import pyplot as plt
import seaborn as sns
#from sklearnex import patch_sklearn
#patch_sklearn(verbose=False)
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder,␣
 ↪StandardScaler, LabelEncoder, FunctionTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪RandomizedSearchCV, GridSearchCV
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.metrics import plot_confusion_matrix, recall_score,␣
 ↪accuracy_score, precision_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImPipeline
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from xgboost import XGBRegressor
```

[ ]:
```
# Included in separate cell in case cat boost is not installed.
from catboost import CatBoostClassifier
```

[18]:
```
X = pd.read_csv('data/Training-set-values.csv')
y = pd.read_csv('data/Training-set-labels.csv')

X['date_recorded'] = pd.to_datetime(X['date_recorded']).astype(np.int64)
```

Preprocessors

[19]:
```
# Super basic numeric transformer
```

```
numeric_transformer = Pipeline(
    steps=[('imputer', SimpleImputer(strategy='median'))]
)

numeric_preprocessor = ColumnTransformer(
    transformers=[
        ("numeric", numeric_transformer, make_column_selector(dtype_include=np.
 ↪number)),
    ]
)
```

0.0.1 Models

[20]:
```
# Gradient Boost
GradBoost = {'classifier': GradientBoostingClassifier(),'preprocessor':␣
 ↪numeric_preprocessor}
GradBoost2 = {'classifier': GradientBoostingClassifier(),'preprocessor': None}
GradBoost3 = {'classifier': GradientBoostingClassifier(),'preprocessor': None}
# XGradient Boosting
XGBoost = {'classifier': XGBRegressor(objective='reg:squarederror'),␣
 ↪'preprocessor': numeric_preprocessor}
# CatBoost
CatBoost = {'classifier': CatBoostClassifier(max_depth=3),'preprocessor':␣
 ↪numeric_preprocessor}



models = {'GradientBoost': GradBoost,
    'GradientBoost2': GradBoost2,
    'GradientBoost3': GradBoost3,
    'XGBoost': XGBoost,
    'CatBoost': CatBoost
    }
```

0.0.2 Modeler

[21]:
```
model_run = ModelClass.Modeler(models, X=X, y=y)

# after the model_run object is created so we can add onto the default␣
 ↪preprocessor.
log_reg_regularized = {'classifier': LogisticRegression(n_jobs=3),␣
 ↪'preprocessor': model_run.create_default_prep(num_add=[('scaling',␣
 ↪StandardScaler())])}
model_run.add_model('log_reg_regularized', log_reg_regularized)
```

0.0.3 Search parameters and kwargs

[22]:
```
GradBoost_params = dict(n_estimators=np.array(range(100, 400)),
                    criterion=['friedman_mse', 'squared_error'],
                    max_depth=np.array(range(2, 10)),
                    min_samples_split=np.array(range(2, 10)),
                    min_samples_leaf=np.array(range(1, 10)),
                    learning_rate=stats.uniform(loc=0.01, scale=1))

GradBoost3_params = dict(n_estimators=np.array(range(200, 1000)),
                    criterion=['friedman_mse', 'squared_error'],
                    max_depth=np.array(range(2, 10)),
                    min_samples_split=np.array(range(2, 10)),
                    min_samples_leaf=np.array(range(1, 10)),
                    learning_rate=stats.uniform(loc=0.001, scale=1))

XGBoost_params = dict(learning_rate =stats.uniform(loc=0.1, scale=0.1),
                    n_estimators=np.array(range(100,1200)),
                    max_depth=np.array(range(4,30)))

CatBoost_params = dict(max_depth =[3,4,5],
                         n_estimators = [100,200,300])

search_options = {'n_jobs': 3, 'random_state': 9280210, 'n_iter': 20}
```

0.1 RandomizedSearchCV

[23]:
```
model_run.hyper_search('GradientBoost', params=GradBoost_params,␣
 ↪searcher_kwargs=search_options, set_to_train=True)
```

[24]:
```
model_run.hyper_search('GradientBoost2', params=GradBoost_params,␣
 ↪searcher_kwargs=search_options, set_to_train=True)
```

```
/Users/valeriaviscarra/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/joblib/externals/loky/process_executor.py:688: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(
```

[25]:
```
model_run.hyper_search('GradientBoost3', params=GradBoost3_params,␣
 ↪searcher_kwargs=search_options, set_to_train=True)
```

[26]:
```
model_run.hyper_search('XGBoost', params=XGBoost_params,␣
 ↪searcher_kwargs=search_options, set_to_train=True)
```

```
/Users/valeriaviscarra/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/joblib/externals/loky/process_executor.py:688: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
```

```
short worker timeout or by a memory leak.
  warnings.warn(
```

[27]:
```
model_run.hyper_search('CatBoost', params=CatBoost_params,␣
↪searcher_kwargs=search_options, set_to_train=True)
```

```
/Users/valeriaviscarra/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/sklearn/model_selection/_search.py:278: UserWarning: The total space of
parameters 9 is smaller than n_iter=20. Running 9 iterations. For exhaustive
searches, use GridSearchCV.
  warnings.warn(
/Users/valeriaviscarra/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/joblib/externals/loky/process_executor.py:688: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(
```

```
Learning rate set to 0.265612
0:      learn: 0.9862021        total: 72.3ms   remaining: 21.6s
1:      learn: 0.9216414        total: 78.8ms   remaining: 11.7s
2:      learn: 0.8824521        total: 84.7ms   remaining: 8.38s
3:      learn: 0.8555248        total: 90.3ms   remaining: 6.68s
4:      learn: 0.8364673        total: 95.2ms   remaining: 5.62s
5:      learn: 0.8218166        total: 100ms    remaining: 4.9s
6:      learn: 0.8129340        total: 105ms    remaining: 4.39s
7:      learn: 0.8058792        total: 111ms    remaining: 4.03s
8:      learn: 0.7989142        total: 116ms    remaining: 3.75s
9:      learn: 0.7937780        total: 121ms    remaining: 3.52s
10:     learn: 0.7907538        total: 127ms    remaining: 3.33s
11:     learn: 0.7854812        total: 132ms    remaining: 3.17s
12:     learn: 0.7833159        total: 137ms    remaining: 3.02s
13:     learn: 0.7807781        total: 142ms    remaining: 2.9s
14:     learn: 0.7768205        total: 148ms    remaining: 2.8s
15:     learn: 0.7742835        total: 153ms    remaining: 2.71s
16:     learn: 0.7717098        total: 159ms    remaining: 2.64s
17:     learn: 0.7701826        total: 164ms    remaining: 2.57s
18:     learn: 0.7680539        total: 169ms    remaining: 2.5s
19:     learn: 0.7654467        total: 175ms    remaining: 2.44s
20:     learn: 0.7643011        total: 180ms    remaining: 2.39s
21:     learn: 0.7621080        total: 185ms    remaining: 2.34s
22:     learn: 0.7607630        total: 190ms    remaining: 2.29s
23:     learn: 0.7599562        total: 196ms    remaining: 2.25s
24:     learn: 0.7587362        total: 201ms    remaining: 2.21s
25:     learn: 0.7570620        total: 207ms    remaining: 2.18s
26:     learn: 0.7555946        total: 213ms    remaining: 2.15s
27:     learn: 0.7547239        total: 218ms    remaining: 2.12s
28:     learn: 0.7524917        total: 224ms    remaining: 2.09s
29:     learn: 0.7519444        total: 229ms    remaining: 2.06s
```

```
30:     learn: 0.7502119        total: 235ms    remaining: 2.04s
31:     learn: 0.7493089        total: 240ms    remaining: 2.01s
32:     learn: 0.7483707        total: 245ms    remaining: 1.98s
33:     learn: 0.7474882        total: 250ms    remaining: 1.95s
34:     learn: 0.7469998        total: 255ms    remaining: 1.93s
35:     learn: 0.7462165        total: 260ms    remaining: 1.91s
36:     learn: 0.7454923        total: 266ms    remaining: 1.89s
37:     learn: 0.7449400        total: 271ms    remaining: 1.86s
38:     learn: 0.7441302        total: 276ms    remaining: 1.84s
39:     learn: 0.7433239        total: 281ms    remaining: 1.83s
40:     learn: 0.7427218        total: 286ms    remaining: 1.81s
41:     learn: 0.7421874        total: 292ms    remaining: 1.79s
42:     learn: 0.7416821        total: 297ms    remaining: 1.77s
43:     learn: 0.7410810        total: 302ms    remaining: 1.76s
44:     learn: 0.7402066        total: 308ms    remaining: 1.74s
45:     learn: 0.7397162        total: 313ms    remaining: 1.73s
46:     learn: 0.7389758        total: 319ms    remaining: 1.72s
47:     learn: 0.7376477        total: 324ms    remaining: 1.7s
48:     learn: 0.7358570        total: 330ms    remaining: 1.69s
49:     learn: 0.7345929        total: 337ms    remaining: 1.69s
50:     learn: 0.7341370        total: 342ms    remaining: 1.67s
51:     learn: 0.7335331        total: 347ms    remaining: 1.66s
52:     learn: 0.7328099        total: 353ms    remaining: 1.64s
53:     learn: 0.7319877        total: 358ms    remaining: 1.63s
54:     learn: 0.7313519        total: 364ms    remaining: 1.62s
55:     learn: 0.7308233        total: 369ms    remaining: 1.61s
56:     learn: 0.7299381        total: 375ms    remaining: 1.6s
57:     learn: 0.7292388        total: 380ms    remaining: 1.58s
58:     learn: 0.7285104        total: 385ms    remaining: 1.57s
59:     learn: 0.7278373        total: 391ms    remaining: 1.56s
60:     learn: 0.7272494        total: 397ms    remaining: 1.55s
61:     learn: 0.7268308        total: 402ms    remaining: 1.54s
62:     learn: 0.7259207        total: 407ms    remaining: 1.53s
63:     learn: 0.7250652        total: 413ms    remaining: 1.52s
64:     learn: 0.7245775        total: 419ms    remaining: 1.51s
65:     learn: 0.7242131        total: 424ms    remaining: 1.5s
66:     learn: 0.7237149        total: 430ms    remaining: 1.5s
67:     learn: 0.7228607        total: 435ms    remaining: 1.49s
68:     learn: 0.7216970        total: 441ms    remaining: 1.48s
69:     learn: 0.7208336        total: 446ms    remaining: 1.47s
70:     learn: 0.7200949        total: 454ms    remaining: 1.46s
71:     learn: 0.7199579        total: 459ms    remaining: 1.45s
72:     learn: 0.7196629        total: 465ms    remaining: 1.44s
73:     learn: 0.7189871        total: 470ms    remaining: 1.43s
74:     learn: 0.7185068        total: 478ms    remaining: 1.43s
75:     learn: 0.7175748        total: 483ms    remaining: 1.42s
76:     learn: 0.7167162        total: 489ms    remaining: 1.41s
77:     learn: 0.7161447        total: 494ms    remaining: 1.41s
```

```
78:     learn: 0.7157405        total: 499ms    remaining: 1.4s
79:     learn: 0.7155016        total: 503ms    remaining: 1.38s
80:     learn: 0.7152352        total: 509ms    remaining: 1.38s
81:     learn: 0.7144699        total: 515ms    remaining: 1.37s
82:     learn: 0.7139931        total: 520ms    remaining: 1.36s
83:     learn: 0.7136669        total: 525ms    remaining: 1.35s
84:     learn: 0.7132828        total: 530ms    remaining: 1.34s
85:     learn: 0.7130521        total: 535ms    remaining: 1.33s
86:     learn: 0.7128256        total: 540ms    remaining: 1.32s
87:     learn: 0.7122340        total: 546ms    remaining: 1.31s
88:     learn: 0.7117152        total: 551ms    remaining: 1.31s
89:     learn: 0.7111235        total: 557ms    remaining: 1.3s
90:     learn: 0.7105650        total: 562ms    remaining: 1.29s
91:     learn: 0.7101654        total: 568ms    remaining: 1.28s
92:     learn: 0.7097073        total: 573ms    remaining: 1.27s
93:     learn: 0.7093692        total: 579ms    remaining: 1.27s
94:     learn: 0.7087599        total: 585ms    remaining: 1.26s
95:     learn: 0.7083993        total: 590ms    remaining: 1.25s
96:     learn: 0.7079316        total: 595ms    remaining: 1.25s
97:     learn: 0.7075359        total: 601ms    remaining: 1.24s
98:     learn: 0.7069493        total: 607ms    remaining: 1.23s
99:     learn: 0.7066890        total: 613ms    remaining: 1.23s
100:    learn: 0.7062836        total: 618ms    remaining: 1.22s
101:    learn: 0.7057782        total: 623ms    remaining: 1.21s
102:    learn: 0.7052138        total: 629ms    remaining: 1.2s
103:    learn: 0.7048717        total: 635ms    remaining: 1.2s
104:    learn: 0.7043149        total: 640ms    remaining: 1.19s
105:    learn: 0.7037837        total: 645ms    remaining: 1.18s
106:    learn: 0.7035477        total: 651ms    remaining: 1.17s
107:    learn: 0.7031474        total: 656ms    remaining: 1.17s
108:    learn: 0.7028812        total: 661ms    remaining: 1.16s
109:    learn: 0.7024816        total: 667ms    remaining: 1.15s
110:    learn: 0.7023014        total: 672ms    remaining: 1.14s
111:    learn: 0.7020343        total: 677ms    remaining: 1.14s
112:    learn: 0.7014935        total: 683ms    remaining: 1.13s
113:    learn: 0.7008822        total: 688ms    remaining: 1.12s
114:    learn: 0.7001683        total: 694ms    remaining: 1.12s
115:    learn: 0.6997198        total: 699ms    remaining: 1.11s
116:    learn: 0.6994358        total: 704ms    remaining: 1.1s
117:    learn: 0.6989996        total: 709ms    remaining: 1.09s
118:    learn: 0.6982237        total: 714ms    remaining: 1.09s
119:    learn: 0.6979792        total: 720ms    remaining: 1.08s
120:    learn: 0.6974945        total: 725ms    remaining: 1.07s
121:    learn: 0.6973029        total: 730ms    remaining: 1.06s
122:    learn: 0.6965589        total: 736ms    remaining: 1.06s
123:    learn: 0.6958777        total: 741ms    remaining: 1.05s
124:    learn: 0.6954549        total: 747ms    remaining: 1.04s
125:    learn: 0.6948730        total: 752ms    remaining: 1.04s
```

```
126:    learn: 0.6944926        total: 758ms    remaining: 1.03s
127:    learn: 0.6942409        total: 763ms    remaining: 1.02s
128:    learn: 0.6938831        total: 768ms    remaining: 1.02s
129:    learn: 0.6933897        total: 774ms    remaining: 1.01s
130:    learn: 0.6929890        total: 779ms    remaining: 1s
131:    learn: 0.6927727        total: 785ms    remaining: 999ms
132:    learn: 0.6925038        total: 790ms    remaining: 992ms
133:    learn: 0.6922629        total: 795ms    remaining: 985ms
134:    learn: 0.6921055        total: 801ms    remaining: 978ms
135:    learn: 0.6915043        total: 806ms    remaining: 972ms
136:    learn: 0.6911007        total: 812ms    remaining: 966ms
137:    learn: 0.6909734        total: 817ms    remaining: 959ms
138:    learn: 0.6907828        total: 822ms    remaining: 952ms
139:    learn: 0.6903439        total: 827ms    remaining: 946ms
140:    learn: 0.6899446        total: 834ms    remaining: 940ms
141:    learn: 0.6897336        total: 839ms    remaining: 933ms
142:    learn: 0.6890028        total: 844ms    remaining: 926ms
143:    learn: 0.6888017        total: 849ms    remaining: 920ms
144:    learn: 0.6885661        total: 854ms    remaining: 913ms
145:    learn: 0.6883078        total: 860ms    remaining: 907ms
146:    learn: 0.6880787        total: 866ms    remaining: 901ms
147:    learn: 0.6879440        total: 871ms    remaining: 895ms
148:    learn: 0.6877710        total: 876ms    remaining: 888ms
149:    learn: 0.6874680        total: 882ms    remaining: 882ms
150:    learn: 0.6868596        total: 887ms    remaining: 875ms
151:    learn: 0.6866838        total: 892ms    remaining: 868ms
152:    learn: 0.6865547        total: 897ms    remaining: 862ms
153:    learn: 0.6863040        total: 902ms    remaining: 856ms
154:    learn: 0.6860710        total: 908ms    remaining: 849ms
155:    learn: 0.6858833        total: 912ms    remaining: 842ms
156:    learn: 0.6856223        total: 917ms    remaining: 836ms
157:    learn: 0.6855290        total: 923ms    remaining: 829ms
158:    learn: 0.6852198        total: 928ms    remaining: 823ms
159:    learn: 0.6850933        total: 933ms    remaining: 816ms
160:    learn: 0.6847890        total: 939ms    remaining: 811ms
161:    learn: 0.6844359        total: 944ms    remaining: 804ms
162:    learn: 0.6842905        total: 949ms    remaining: 797ms
163:    learn: 0.6837388        total: 954ms    remaining: 791ms
164:    learn: 0.6835274        total: 960ms    remaining: 785ms
165:    learn: 0.6830261        total: 965ms    remaining: 779ms
166:    learn: 0.6829305        total: 970ms    remaining: 772ms
167:    learn: 0.6824248        total: 975ms    remaining: 766ms
168:    learn: 0.6822852        total: 980ms    remaining: 760ms
169:    learn: 0.6821000        total: 986ms    remaining: 754ms
170:    learn: 0.6819819        total: 991ms    remaining: 747ms
171:    learn: 0.6818358        total: 997ms    remaining: 742ms
172:    learn: 0.6815077        total: 1s       remaining: 736ms
173:    learn: 0.6808999        total: 1.01s    remaining: 731ms
```

```
174:    learn: 0.6803657        total: 1.01s    remaining: 725ms
175:    learn: 0.6800620        total: 1.02s    remaining: 718ms
176:    learn: 0.6797150        total: 1.02s    remaining: 712ms
177:    learn: 0.6793064        total: 1.03s    remaining: 706ms
178:    learn: 0.6791988        total: 1.03s    remaining: 700ms
179:    learn: 0.6789792        total: 1.04s    remaining: 693ms
180:    learn: 0.6787075        total: 1.04s    remaining: 688ms
181:    learn: 0.6785000        total: 1.05s    remaining: 682ms
182:    learn: 0.6782578        total: 1.06s    remaining: 676ms
183:    learn: 0.6780926        total: 1.06s    remaining: 669ms
184:    learn: 0.6779609        total: 1.07s    remaining: 663ms
185:    learn: 0.6777202        total: 1.07s    remaining: 657ms
186:    learn: 0.6774143        total: 1.08s    remaining: 651ms
187:    learn: 0.6770613        total: 1.08s    remaining: 645ms
188:    learn: 0.6769468        total: 1.09s    remaining: 639ms
189:    learn: 0.6766225        total: 1.09s    remaining: 633ms
190:    learn: 0.6763521        total: 1.1s     remaining: 627ms
191:    learn: 0.6759093        total: 1.1s     remaining: 621ms
192:    learn: 0.6756899        total: 1.11s    remaining: 615ms
193:    learn: 0.6754265        total: 1.11s    remaining: 609ms
194:    learn: 0.6752286        total: 1.12s    remaining: 603ms
195:    learn: 0.6750735        total: 1.13s    remaining: 597ms
196:    learn: 0.6749758        total: 1.13s    remaining: 591ms
197:    learn: 0.6747822        total: 1.14s    remaining: 585ms
198:    learn: 0.6745848        total: 1.14s    remaining: 579ms
199:    learn: 0.6743064        total: 1.15s    remaining: 573ms
200:    learn: 0.6740929        total: 1.15s    remaining: 567ms
201:    learn: 0.6738261        total: 1.16s    remaining: 561ms
202:    learn: 0.6736186        total: 1.16s    remaining: 555ms
203:    learn: 0.6734258        total: 1.17s    remaining: 550ms
204:    learn: 0.6731006        total: 1.17s    remaining: 544ms
205:    learn: 0.6729178        total: 1.18s    remaining: 538ms
206:    learn: 0.6728008        total: 1.18s    remaining: 532ms
207:    learn: 0.6726040        total: 1.19s    remaining: 526ms
208:    learn: 0.6722880        total: 1.19s    remaining: 520ms
209:    learn: 0.6719992        total: 1.2s     remaining: 514ms
210:    learn: 0.6717141        total: 1.2s     remaining: 508ms
211:    learn: 0.6715837        total: 1.21s    remaining: 503ms
212:    learn: 0.6713350        total: 1.22s    remaining: 497ms
213:    learn: 0.6711283        total: 1.22s    remaining: 491ms
214:    learn: 0.6708570        total: 1.23s    remaining: 485ms
215:    learn: 0.6703870        total: 1.23s    remaining: 479ms
216:    learn: 0.6701590        total: 1.24s    remaining: 474ms
217:    learn: 0.6700615        total: 1.24s    remaining: 468ms
218:    learn: 0.6698384        total: 1.25s    remaining: 462ms
219:    learn: 0.6695544        total: 1.25s    remaining: 456ms
220:    learn: 0.6693170        total: 1.26s    remaining: 451ms
221:    learn: 0.6691462        total: 1.26s    remaining: 445ms
```

```
222:    learn: 0.6688191        total: 1.27s    remaining: 439ms
223:    learn: 0.6684846        total: 1.27s    remaining: 433ms
224:    learn: 0.6683718        total: 1.28s    remaining: 427ms
225:    learn: 0.6681095        total: 1.29s    remaining: 421ms
226:    learn: 0.6678199        total: 1.29s    remaining: 416ms
227:    learn: 0.6675496        total: 1.3s     remaining: 410ms
228:    learn: 0.6672544        total: 1.3s     remaining: 404ms
229:    learn: 0.6670559        total: 1.31s    remaining: 398ms
230:    learn: 0.6666793        total: 1.31s    remaining: 392ms
231:    learn: 0.6665645        total: 1.32s    remaining: 386ms
232:    learn: 0.6663121        total: 1.32s    remaining: 381ms
233:    learn: 0.6660358        total: 1.33s    remaining: 375ms
234:    learn: 0.6658186        total: 1.33s    remaining: 369ms
235:    learn: 0.6654950        total: 1.34s    remaining: 363ms
236:    learn: 0.6653635        total: 1.34s    remaining: 358ms
237:    learn: 0.6651501        total: 1.35s    remaining: 352ms
238:    learn: 0.6649676        total: 1.35s    remaining: 346ms
239:    learn: 0.6647443        total: 1.36s    remaining: 340ms
240:    learn: 0.6646156        total: 1.37s    remaining: 334ms
241:    learn: 0.6644916        total: 1.37s    remaining: 329ms
242:    learn: 0.6644134        total: 1.38s    remaining: 323ms
243:    learn: 0.6642222        total: 1.38s    remaining: 317ms
244:    learn: 0.6639145        total: 1.39s    remaining: 311ms
245:    learn: 0.6636801        total: 1.39s    remaining: 306ms
246:    learn: 0.6632706        total: 1.4s     remaining: 300ms
247:    learn: 0.6631025        total: 1.4s     remaining: 295ms
248:    learn: 0.6629269        total: 1.41s    remaining: 289ms
249:    learn: 0.6627558        total: 1.42s    remaining: 283ms
250:    learn: 0.6625230        total: 1.42s    remaining: 277ms
251:    learn: 0.6622841        total: 1.43s    remaining: 272ms
252:    learn: 0.6620956        total: 1.43s    remaining: 266ms
253:    learn: 0.6618617        total: 1.44s    remaining: 260ms
254:    learn: 0.6617341        total: 1.44s    remaining: 254ms
255:    learn: 0.6615511        total: 1.45s    remaining: 249ms
256:    learn: 0.6613594        total: 1.45s    remaining: 243ms
257:    learn: 0.6611178        total: 1.46s    remaining: 238ms
258:    learn: 0.6608614        total: 1.46s    remaining: 232ms
259:    learn: 0.6604988        total: 1.47s    remaining: 226ms
260:    learn: 0.6602466        total: 1.48s    remaining: 220ms
261:    learn: 0.6599955        total: 1.48s    remaining: 215ms
262:    learn: 0.6597054        total: 1.49s    remaining: 209ms
263:    learn: 0.6595785        total: 1.49s    remaining: 203ms
264:    learn: 0.6593887        total: 1.5s     remaining: 198ms
265:    learn: 0.6592600        total: 1.5s     remaining: 192ms
266:    learn: 0.6591580        total: 1.51s    remaining: 186ms
267:    learn: 0.6589472        total: 1.51s    remaining: 180ms
268:    learn: 0.6587035        total: 1.52s    remaining: 175ms
269:    learn: 0.6585131        total: 1.52s    remaining: 169ms
```

```
270:    learn: 0.6582320        total: 1.53s    remaining: 163ms
271:    learn: 0.6581307        total: 1.53s    remaining: 158ms
272:    learn: 0.6578864        total: 1.54s    remaining: 152ms
273:    learn: 0.6577162        total: 1.54s    remaining: 146ms
274:    learn: 0.6574203        total: 1.55s    remaining: 141ms
275:    learn: 0.6573086        total: 1.55s    remaining: 135ms
276:    learn: 0.6570447        total: 1.56s    remaining: 129ms
277:    learn: 0.6568172        total: 1.56s    remaining: 124ms
278:    learn: 0.6565818        total: 1.57s    remaining: 118ms
279:    learn: 0.6564033        total: 1.57s    remaining: 113ms
280:    learn: 0.6558969        total: 1.58s    remaining: 107ms
281:    learn: 0.6557175        total: 1.59s    remaining: 101ms
282:    learn: 0.6554100        total: 1.59s    remaining: 95.7ms
283:    learn: 0.6553040        total: 1.6s     remaining: 90.1ms
284:    learn: 0.6550741        total: 1.6s     remaining: 84.4ms
285:    learn: 0.6547515        total: 1.61s    remaining: 78.8ms
286:    learn: 0.6545225        total: 1.61s    remaining: 73.2ms
287:    learn: 0.6544488        total: 1.62s    remaining: 67.5ms
288:    learn: 0.6542965        total: 1.63s    remaining: 61.9ms
289:    learn: 0.6539549        total: 1.63s    remaining: 56.2ms
290:    learn: 0.6538154        total: 1.64s    remaining: 50.6ms
291:    learn: 0.6536573        total: 1.64s    remaining: 45ms
292:    learn: 0.6534931        total: 1.65s    remaining: 39.4ms
293:    learn: 0.6533516        total: 1.65s    remaining: 33.7ms
294:    learn: 0.6532494        total: 1.66s    remaining: 28.1ms
295:    learn: 0.6529555        total: 1.66s    remaining: 22.5ms
296:    learn: 0.6527301        total: 1.67s    remaining: 16.9ms
297:    learn: 0.6526237        total: 1.67s    remaining: 11.2ms
298:    learn: 0.6524011        total: 1.68s    remaining: 5.61ms
299:    learn: 0.6522955        total: 1.68s    remaining: 0us
```

0.2 Test Models

[43]:
```
# Gradient Boost
gb_model=model_run.get_model('GradientBoost')['model_pipeline']
Gradient_Boost = gb_model.score(X=model_run._X_test, y=model_run._y_test)
Gradient_Boost
```

[43]:
```
0.705993265993266
```

[44]:
```
# Gradient Boost 2
gb2_model= model_run.get_model('GradientBoost2')['model_pipeline']
Gradient_Boost2 = gb2_model.score(X=model_run._X_test, y=model_run._y_test)
Gradient_Boost2
```

[44]:
```
0.8046464646464646
```

```
[45]: # Gradient Boost 3
      gb3_model= model_run.get_model('GradientBoost3')['model_pipeline']
      Gradient_Boost3 = gb3_model.score(X=model_run._X_test, y=model_run._y_test)
      Gradient_Boost3
```

```
[45]: 0.8052525252525252
```

```
[46]: # XGBoost
      xgb_model= model_run.get_model('XGBoost')['model_pipeline']
      XG_Boost = xgb_model.score(X=model_run._X_test, y=model_run._y_test)
      XG_Boost
```

```
[46]: 0.29846332133089515
```

```
[47]: # CatBoost
      cb_model= model_run.get_model('CatBoost')['model_pipeline']
      Cat_Boost = cb_model.score(X=model_run._X_test, y=model_run._y_test)
      Cat_Boost
```

```
[47]: 0.6808754208754209
```

```
[48]: boost_models = {'Gradient_Boost': 0.705993265993266,
          'Gradient_Boost2': 0.8046464646464646,
          'Gradient_Boost3': 0.8052525252525252,
          'XG_Boost': 0.29846332133089515,
          'Cat_Boost': 0.6808754208754209
          }
```

```
[51]: model_run.test_model('GradientBoost3')
```

```
root - INFO - GradientBoost3 test score: 0.8052525252525252
```

0.3 Plotting

```
[40]: plot_models(self, sns_style='darkgrid', sns_context='talk', palette='coolwarm',␣
       ↪save=None, labels=None):
              """
              Skylar slide style, with thanks to Matt. Has options for seaborn␣
       ↪plotting. If you want to save the plot,
              give the save option a filename, exactly as would be done with plt.
       ↪savefig() Labels must be provided as a
              dictionary with the model names as keys and the Label you'd like to␣
       ↪display as a value.
              """
              logger.removeHandler(c_handler)
              logger.removeHandler(f_handler)
```

```
        xticklabels = [labels[key] for key in self._models.keys()] if labels␣
↪else list(self._models.keys())
        y = [model['test_output'] for model in self._models.values()]

        sns.set_style(sns_style)
        sns.set_context(sns_context)
        fig, ax = plt.subplots(figsize=(20, 10))

        fig.set_tight_layout(True)

        sns.barplot(x=xticklabels, y=y, palette=palette)
        ax.set(ylim=(0, 1))
        ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
↪horizontalalignment='right')

        # ax2 = ax.twinx()
        # sns.lineplot(x=xticklabels, y=x_error, linewidth=5)
        # ax2.set(ylim=(0, 300000))
        # ax2.set_yticks(np.linspace(0,300000,num=6))
        # ax2.set_yticklabels(np.linspace(0,300,num=6,dtype=int))

        ax.set_ylabel('Accuracy Score')
        # ax2.set_ylabel('Error, USD (thousands)')
        ax.set_title('Model Effectiveness');

        if save:
            plt.savefig(save)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-40-03995aa7b32b> in <module>
----> 1 model_run.plot_models(save='boost_models_graph')

~/Desktop/Tanzania-Well-Project/ourfunctions.py in plot_models(self, sns_style,␣
 ↪sns_context, palette, save, labels)
    387
    388         xticklabels = [labels[key] for key in self._models.keys()] if␣
 ↪labels else list(self._models.keys())
--> 389         y = [model['test_output'] for model in self._models.values()]
    390
    391         sns.set_style(sns_style)

~/Desktop/Tanzania-Well-Project/ourfunctions.py in <listcomp>(.0)
    387
    388         xticklabels = [labels[key] for key in self._models.keys()] if␣
 ↪labels else list(self._models.keys())
--> 389         y = [model['test_output'] for model in self._models.values()]
```

```
    390
    391         sns.set_style(sns_style)

KeyError: 'test_output'
```

0.4 Modeler

0.4.1 Model 1

```
[59]: model_run.train_model('GradientBoost3', cv=False)
```

```
root - INFO - GradientBoost3 has been fit.
```

```
[55]: importance_kwargs = dict(n_repeats=10, n_jobs=3)
      model_run.permutation_importance('GradientBoost3',␣
       ↪perm_kwargs=importance_kwargs)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-55-284394354893> in <module>
      1 importance_kwargs = dict(n_repeats=10, n_jobs=3)
----> 2 model_run.permutation_importance('GradientBoost3',␣
 ↪perm_kwargs=importance_kwargs)

~/Desktop/Tanzania-Well-Project/ourfunctions.py in permutation_importance(self,␣
 ↪name, train, perm_kwargs, save_graph)
    363         X_val, y_val = (self._X_train, self._y_train) if train else␣
 ↪(self._X_test, self._y_test)
    364
--> 365         model_permuter = permutation_importance(model_pipeline, X_val,␣
 ↪y_val, **perm_kwargs) if perm_kwargs else␣
 ↪permutation_importance(model_pipeline, X_val, y_val)
    366         model['permuter'] = model_permuter
    367

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/
 ↪validation.py in inner_f(*args, **kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/inspection/
 ↪_permutation_importance.py in permutation_importance(estimator, X, y, scoring ␣
 ↪n_repeats, n_jobs, random_state)
    131
    132     scorer = check_scoring(estimator, scoring=scoring)
```

```
--> 133     baseline_score = scorer(estimator, X, y)
    134 
    135     scores =␣
 ↪Parallel(n_jobs=n_jobs)(delayed(_calculate_permutation_scores)(

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/metrics/
 ↪_scorer.py in _passthrough_scorer(estimator, *args, **kwargs)
    370 def _passthrough_scorer(estimator, *args, **kwargs):
    371     """Function that wraps estimator.score"""
--> 372     return estimator.score(*args, **kwargs)
    373 
    374 

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/
 ↪metaestimators.py in <lambda>(*args, **kwargs)
    117 
    118         # lambda, but not partial, allows help() to work with␣
 ↪update_wrapper
--> 119         out = lambda *args, **kwargs: self.fn(obj, *args, **kwargs)
    120         # update the docstring of the returned function
    121         update_wrapper(out, self.fn)

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/pipeline.py␣
 ↪in score(self, X, y, sample_weight)
    609         if sample_weight is not None:
    610             score_params['sample_weight'] = sample_weight
--> 611         return self.steps[-1][-1].score(Xt, y, **score_params)
    612 
    613     @property

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/base.py in␣
 ↪score(self, X, y, sample_weight)
    497         """
    498         from .metrics import accuracy_score
--> 499         return accuracy_score(y, self.predict(X),␣
 ↪sample_weight=sample_weight)
    500 
    501     def _more_tags(self):

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/ensemble/_gb
 ↪py in predict(self, X)
   1170             The predicted values.
   1171         """
-> 1172         raw_predictions = self.decision_function(X)
   1173         encoded_labels = \
   1174             self.loss_._raw_prediction_to_decision(raw_predictions)
```

```
~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/ensemble/_gb
↪py in decision_function(self, X)
   1126         """
   1127         X = check_array(X, dtype=DTYPE, order="C", accept_sparse='csr')
-> 1128         raw_predictions = self._raw_predict(X)
   1129         if raw_predictions.shape[1] == 1:
   1130             return raw_predictions.ravel()

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/ensemble/_gb
↪py in _raw_predict(self, X)
    617         """Return the sum of the trees raw predictions (+ init␣
↪estimator)."""
    618         raw_predictions = self._raw_predict_init(X)
--> 619         predict_stages(self.estimators_, X, self.learning_rate,

    620                        raw_predictions)
    621         return raw_predictions

sklearn/ensemble/_gradient_boosting.pyx in sklearn.ensemble._gradient_boosting.
↪predict_stages()

sklearn/ensemble/_gradient_boosting.pyx in sklearn.ensemble._gradient_boosting.
↪_predict_regression_tree_stages_sparse()

AttributeError: 'NoneType' object has no attribute 'tree_'
```

[57]:
```
from sklearn.inspection import permutation_importance

model_pipeline = gb3_model

X_val, y_val = (model_run._X_test, model_run._y_test)

model_permuter = permutation_importance(model_pipeline, X_val, y_val,␣
↪**importance_kwargs)

# Plotting
fig, ax = plt.subplots(figsize=(10,4))
perm_imp = pd.Series(model_permuter.importances_mean, index=X_val.columns).
↪sort_values(ascending=False)[:10]
perm_imp.plot(kind="barh", title="Permutation Importances")
ax.set(ylabel="Mean Permutation Importance Score")
ax.invert_yaxis()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-57-f333355db7ff> in <module>
      5 X_val, y_val = (model_run._X_test, model_run._y_test)
      6
----> 7 model_permuter = permutation_importance(model_pipeline, X_val, y_val,␣
 ↪**importance_kwargs)
      8
      9 # Plotting

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/
 ↪validation.py in inner_f(*args, **kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/inspection/
 ↪_permutation_importance.py in permutation_importance(estimator, X, y, scoring ␣
 ↪n_repeats, n_jobs, random_state)
    131
    132     scorer = check_scoring(estimator, scoring=scoring)
--> 133     baseline_score = scorer(estimator, X, y)
    134
    135     scores =␣
 ↪Parallel(n_jobs=n_jobs)(delayed(_calculate_permutation_scores)(

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/metrics/
 ↪_scorer.py in _passthrough_scorer(estimator, *args, **kwargs)
    370 def _passthrough_scorer(estimator, *args, **kwargs):
    371     """Function that wraps estimator.score"""
--> 372     return estimator.score(*args, **kwargs)
    373
    374

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/
 ↪metaestimators.py in <lambda>(*args, **kwargs)
    117
    118         # lambda, but not partial, allows help() to work with␣
 ↪update_wrapper
--> 119         out = lambda *args, **kwargs: self.fn(obj, *args, **kwargs)
    120         # update the docstring of the returned function
    121         update_wrapper(out, self.fn)

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/pipeline.py␣
 ↪in score(self, X, y, sample_weight)
    609         if sample_weight is not None:
    610             score_params['sample_weight'] = sample_weight
--> 611         return self.steps[-1][-1].score(Xt, y, **score_params)
    612
    613     @property
```

```
~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/base.py in␣
 ↪score(self, X, y, sample_weight)
    497         """
    498         from .metrics import accuracy_score
--> 499         return accuracy_score(y, self.predict(X),␣
 ↪sample_weight=sample_weight)
    500 
    501     def _more_tags(self):

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/ensemble/_gb
 ↪py in predict(self, X)
   1170             The predicted values.
   1171         """
-> 1172         raw_predictions = self.decision_function(X)
   1173         encoded_labels = \
   1174             self.loss_._raw_prediction_to_decision(raw_predictions)

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/ensemble/_gb
 ↪py in decision_function(self, X)
   1126         """
   1127         X = check_array(X, dtype=DTYPE, order="C", accept_sparse='csr')
-> 1128         raw_predictions = self._raw_predict(X)
   1129         if raw_predictions.shape[1] == 1:
   1130             return raw_predictions.ravel()

~/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/ensemble/_gb
 ↪py in _raw_predict(self, X)
    617         """Return the sum of the trees raw predictions (+ init␣
 ↪estimator)."""
    618         raw_predictions = self._raw_predict_init(X)
--> 619         predict_stages(self.estimators_, X, self.learning_rate,

    620                        raw_predictions)
    621         return raw_predictions

sklearn/ensemble/_gradient_boosting.pyx in sklearn.ensemble._gradient_boosting.
 ↪predict_stages()

sklearn/ensemble/_gradient_boosting.pyx in sklearn.ensemble._gradient_boosting.
 ↪_predict_regression_tree_stages_sparse()

AttributeError: 'NoneType' object has no attribute 'tree_'
```

[54]:
```
model_run.get_model('GradientBoost3')
```

```
[54]: {'classifier': GradientBoostingClassifier(),
        'preprocessor': None,
        'model_pipeline': Pipeline(steps=[('preprocessor',
                          ColumnTransformer(transformers=[('numeric',
                                                           Pipeline(steps=[('imputer',
       SimpleImputer(strategy='median'))]),
       <sklearn.compose._column_transformer.make_column_selector object at
       0x7fae9a764fa0>),
                                                          ('categorical',
                                                           Pipeline(steps=[('imputer',
       SimpleImputer(fill_value='Missing',
         strategy='constant')),
                                                                           ('casting',
       FunctionTransformer(…create_default_prep.<locals>.to_object at
       0x7fae89c700d0>)),
       ('one_hot_encode',
       OneHotEncoder(handle_unknown='ignore'))]),
       <sklearn.compose._column_transformer.make_column_selector object at
       0x7fae9a764940>)])),
                         ('classifier',
                          GradientBoostingClassifier(learning_rate=0.20435421813016375,
                                                     max_depth=9, min_samples_leaf=9,
                                                     min_samples_split=3,
                                                     n_estimators=428))]),
        'param_distro': {'classifier__n_estimators': array([200, 201, 202, 203, 204,
       205, 206, 207, 208, 209, 210, 211, 212,
                213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225,
                226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238,
                239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251,
                252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264,
                265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277,
                278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290,
                291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303,
                304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316,
                317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329,
                330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342,
                343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355,
                356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368,
                369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381,
                382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394,
                395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407,
                408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420,
                421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433,
                434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446,
                447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459,
                460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472,
                473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485,
                486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498,
                499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511,
                512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524,
                525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537,
                538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550,
                551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563,
                564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576,
                577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589,
                590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602,
                603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615,
                616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628,
                629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641,
                642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654,
                655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667,
                668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680,
                681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693,
                694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706,
                707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719,
                720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732,
                733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745,
                746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758,
                759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771,
                772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784,
                785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797,
                798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810,
                811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823,
                824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836,
                837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849,
                850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862,
                863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875,
                876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888,
                889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901,
                902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914,
                915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927,
                928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940,
                941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953,
                954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966,
                967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979,
                980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992,
                993, 994, 995, 996, 997, 998, 999]),
         'classifier__criterion': ['friedman_mse', 'squared_error'],
         'classifier__max_depth': array([2, 3, 4, 5, 6, 7, 8, 9]),
         'classifier__min_samples_split': array([2, 3, 4, 5, 6, 7, 8, 9]),
         'classifier__min_samples_leaf': array([1, 2, 3, 4, 5, 6, 7, 8, 9]),
         'classifier__learning_rate': <scipy.stats._distn_infrastructure.rv_frozen at
       0x7fae785240d0>},
        'search_classifier': Pipeline(steps=[('preprocessor',
                          ColumnTransformer(transformers=[('numeric',
                                                           Pipeline(steps=[('imputer',
       SimpleImputer(strategy='median'))]),
       <sklearn.compose._column_transformer.make_column_selector object at
       0x7fae9a764fa0>),
                                                         ('categorical',
                                                           Pipeline(steps=[('imputer',
       SimpleImputer(fill_value='Missing',
         strategy='constant')),
                                                                          ('casting',
       FunctionTransformer(…create_default_prep.<locals>.to_object at
       0x7fae89c700d0>)),
       ('one_hot_encode',
       OneHotEncoder(handle_unknown='ignore'))]),
       <sklearn.compose._column_transformer.make_column_selector object at
       0x7fae9a764940>)])),
                        ('classifier',
                          GradientBoostingClassifier(learning_rate=0.20435421813016375,
                                                     max_depth=9, min_samples_leaf=9,
                                                     min_samples_split=3,
                                                     n_estimators=428))]),
        'search_best_params': {'classifier__criterion': 'friedman_mse',
         'classifier__learning_rate': 0.20435421813016375,
         'classifier__max_depth': 9,
         'classifier__min_samples_leaf': 9,
         'classifier__min_samples_split': 3,
         'classifier__n_estimators': 428},
        'search_performed_at': 'Fri Jan 28 02:32:15 2022',
        'time_fit': 'Fri Jan 28 02:32:15 2022',
        'test_output': 0.8052525252525252,
        'time_tested': 'Fri Jan 28 10:36:36 2022',
        'cv_output': array([0.79674523, 0.8016835 , 0.80617284, 0.79887767,
       0.78956229]),
        'time_cross_val': 'Fri Jan 28 11:10:20 2022'}
```

0.4.2 Model 2

```
[ ]: model_run.model_evaluation('GradientBoost2')
```

```
[ ]: importance_kwargs = dict(n_repeats=10, n_jobs=3)
     model_run.permutation_importance('GradientBoost2',␣
      ↪perm_kwargs=importance_kwargs)
```