In [1]:
```
import pandas as pd
import numpy as np
import sys
import os
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 8, 4

module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path + '/')

import Plib.ML.Models as ml
import Plib.Signals.TAnalysis as ta
import Plib.Signals.Filters as flt
import pickle

import warnings
warnings.filterwarnings("ignore")

#feature_names_in
```

In [2]:
```
fname='goldReg.pickle'
with open(fname, 'rb') as handle:
    data = pickle.load(handle)

#Aligna data in sdata
min_data=[]
for k in data.keys(): min_data.append(data[k].index.min())
d1=str(max(min_data))
max_data=[]
for k in data.keys(): max_data.append(data[k].index.max())
d2=str(min(max_data))
sdata={}
for k in data.keys():
    df=data[k]
    df=df[(df.index>=d1) & (df.index<=d2)]
    sdata[k]=df
```

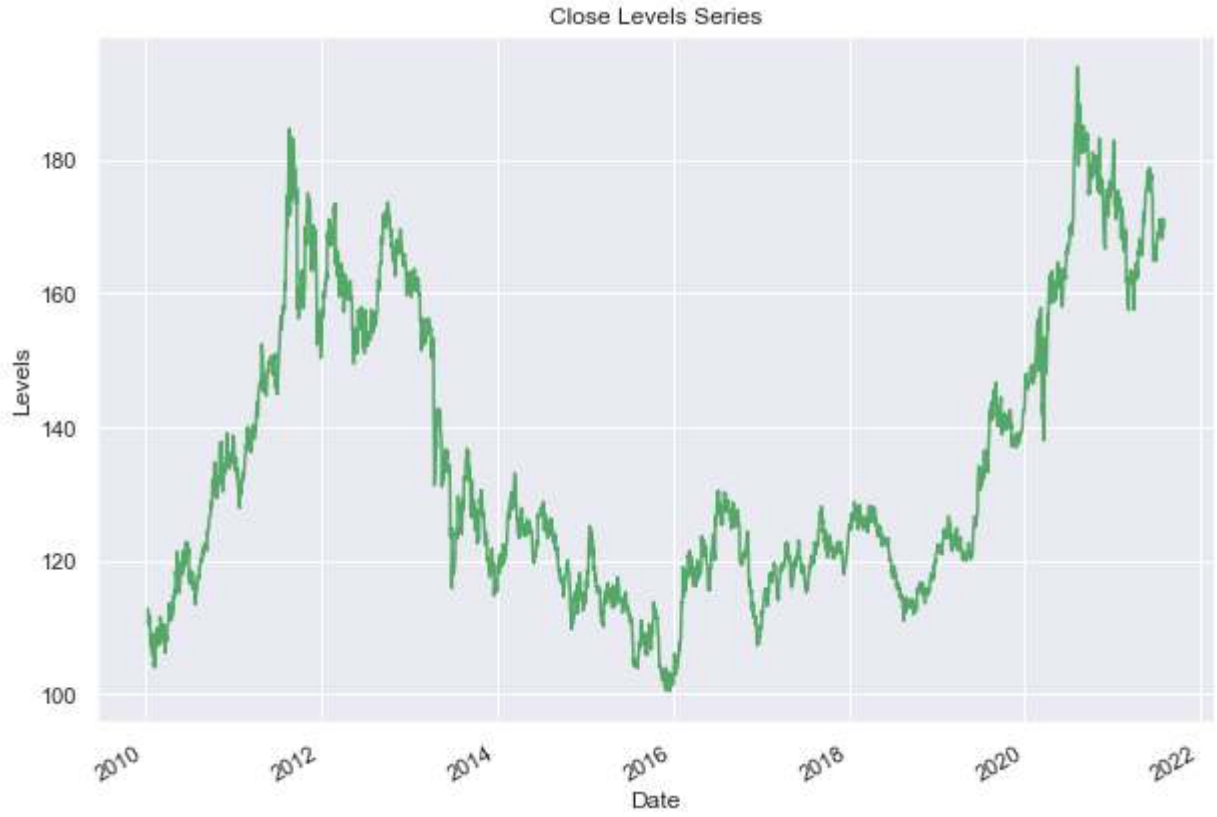
Class definition for strategy

In [3]:
```
class MLTDT(ml.DTC):

    @staticmethod
    def fs_generation(df,l_rets=[3,15,30],w_sma=[3,15,60],drop_nan=True):
        d1=ta.getVolumeGap(df)
        d1=ta.getDailyChange(d1)
        d1=ta.getOpenSpread(d1)
        d1=ta.getLaggedRets(d1,periods=l_rets)
        d1=ta.getSMAs(d1,periods=w_sma)
        d1=ta.getMarketUpDown(d1)
        fast=min(w_sma)
        d1=ta.getSMACorr(d1,lbl_sma='sma_'+str(fast))
        nan_index=max(max(w_sma),max(l_rets))
        if drop_nan:
            d1=d1[nan_index:]
            d1=d1.dropna()
        features=[c for c in d1.columns if c not in ['Open','High','Low','Close','Adj
        return d1,list(features)

    @staticmethod
    def makeStudy(df,features=[],regrnd1='',test_periods=252,split=0.85,table=False):
```

```
        df=df[:-test_periods].copy()
        df1 = MLTDT.scaling(df[[*([regrnd1]+features)]],params={'method':'minmax','ex
        stats = MLTDT.decisionTreeClassifier(df1,regrnd1,t = split,table=table)
        return stats
```

Data and plot

In [4]:
```
# Instantiate Linreg algo
algo=MLTDT(5)

# Select data
gold = sdata['gld_etf']

# Clean data
gold= algo.cleanData(gold)

# Create Features
gold,features=algo.fs_generation(gold)

# Plot the closing price of GLD 11
algo.plotLevels(gold, benchmark_level='Close')
```

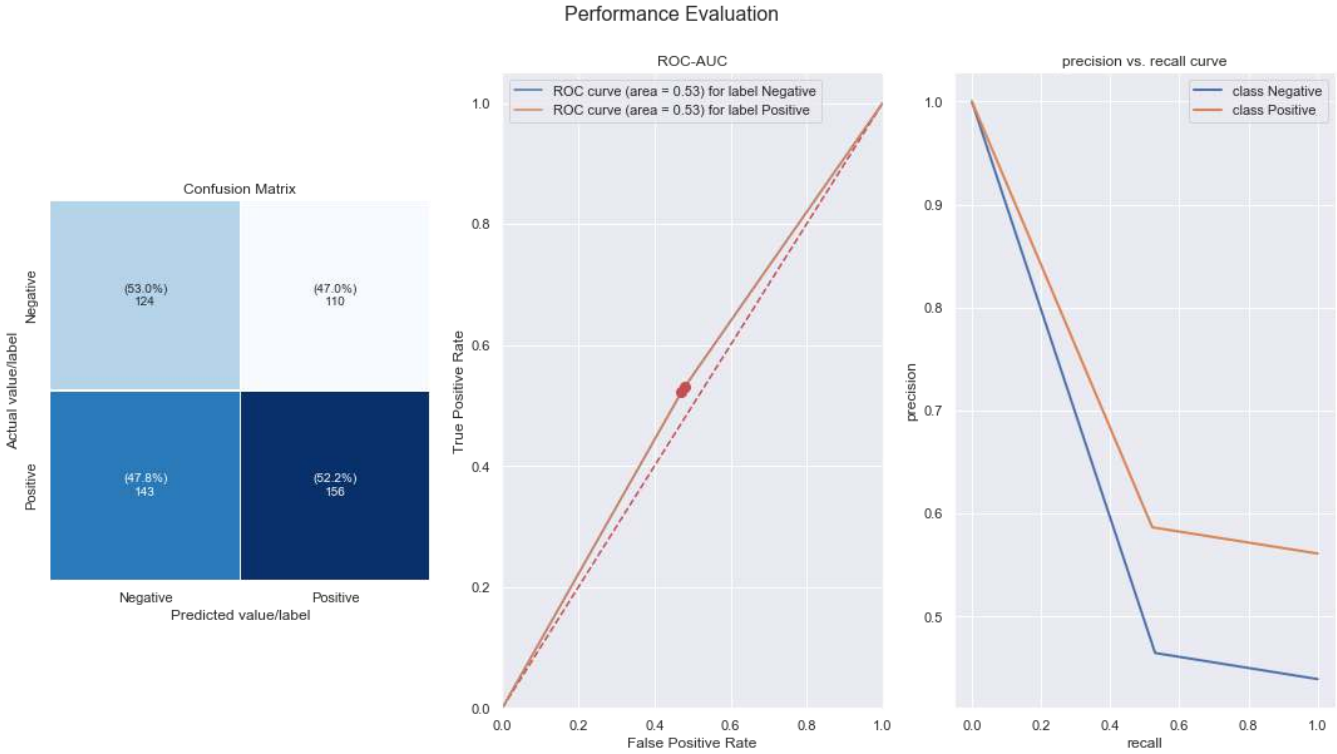


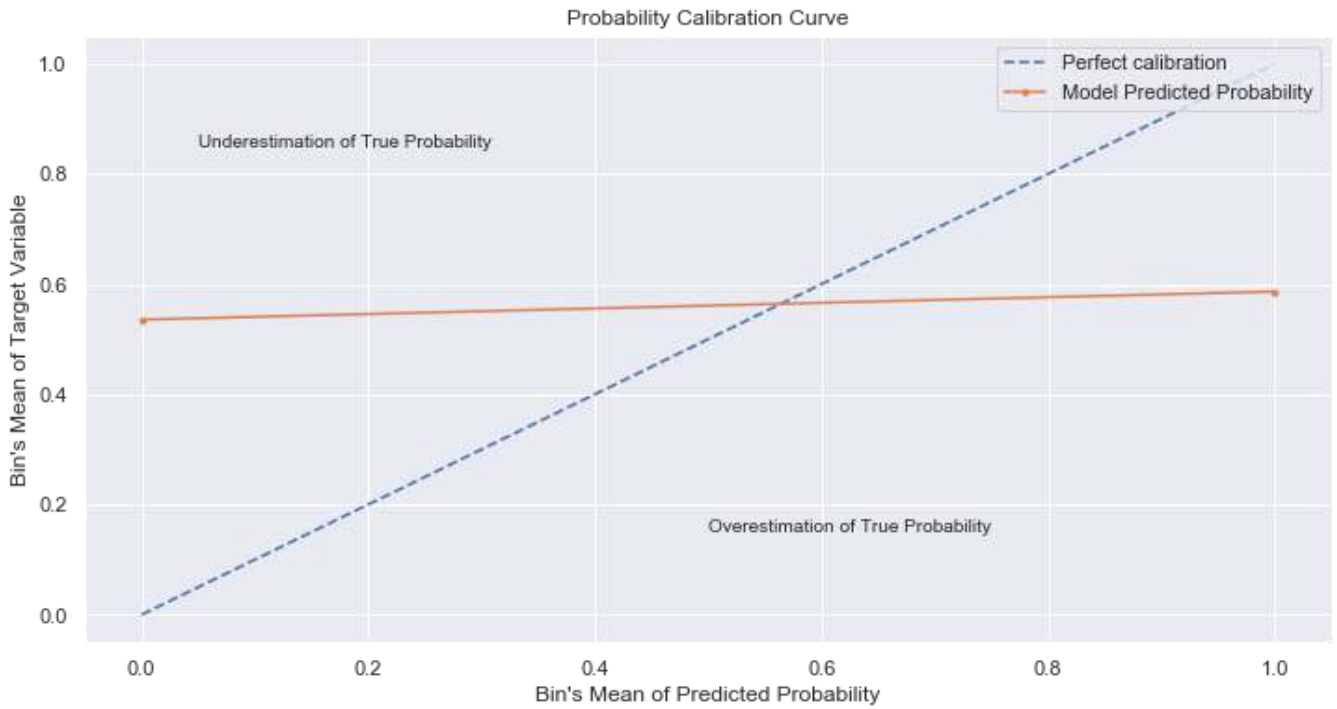
Test and Diagnostics

In [7]:
```
test_periods=252
split=0.8
regrnd1='MarketUpDown_Close1day'
gold_scaled = algo.scaling(gold[[*([regrnd1]+features)]],params={'method':'minmax','e

ret=algo.makeStudy(gold,features,regrnd1,
                        test_periods=test_periods,
                        split=split,table=True)

strategy=algo.applySignal(gold,algo.getProbs(ret['model'],gold_scaled,features),benchr
```

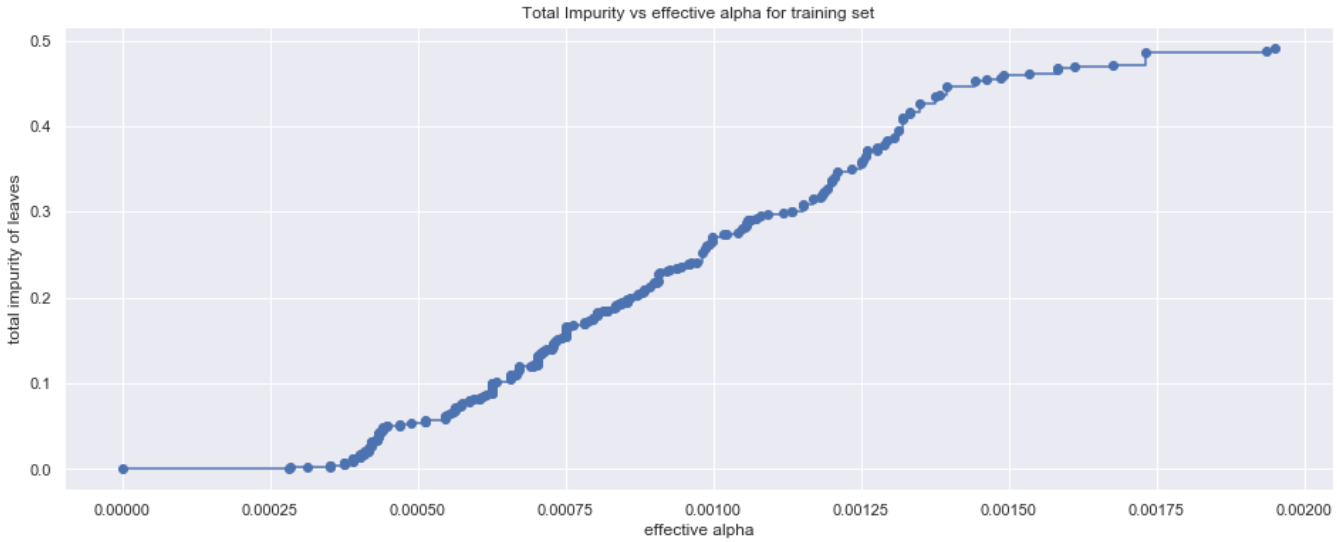
```
algo.plotReturns(strategy)
```

```
-----------------------------------------
Accuracy [0.5253 0.5253]
Recall [0.5299 0.5217]
MCC 0.0513
Specificity [0.5217 0.5299]
F1_score [0.495  0.5522]
Cohens Kappa 0.0509
-----------------------------------------
```






```
Transformation of Probabilities:
-----------------------------------------
Model                                0.47
Model + Isotonic regression          0.0
Model + Logistic regression          0.0
Platt Scaling * Model                0.07
-----------------------------------------
```




```
Number of nodes in the last tree is: 3 with ccp_alpha: 0.0021332426467403076
```

Test

In [8]:
```
test_periods=252
regrnd1='MarketUpDown_Close1day'

# Strategy data
strategy=gold[:int(len(gold[:-test_periods]))].copy()
gold_scaled = algo.scaling(gold[[*([regrnd1]+features)]],params={'method':'minmax','e
ret=algo.makeStudy(gold,features,regrnd1='MarketUpDown_Close1day',
                           test_periods=test_periods,
                           split=0.75,table=False)

# Test the strategy dy-by-day
print('Start: ', str(strategy.tail(1).index.date[0]))
for d in range(1,test_periods-1):
    last_day=algo.applySignal(gold[:-test_periods+d],algo.getProbs(ret['model'],gold_
    strategy=strategy.append(last_day)
print('End: ', str(strategy.tail(1).index.date[0]))

algo.plotReturns(strategy)
```

```
Start:  2020-07-30
End:  2021-07-28
```

Features Selection

In [5]:
```
test_periods=252
gold_study=gold[:-test_periods].copy()
features_set=['sma_3','sma_15','sma_60','volumeGap_','dailyChange_','OD_','OL_',
 'lagRets_3','lagRets_15','lagRets_30','corr_sma_3']

d1,d2=algo.fs_analysis(gold_study[[*features]])
```

```
Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/scikit-learn-intelex)
```

Features with Lowest Variance

LowVar	volumeGap_	dailyChange_	OD_	OL_	fractHigh_	fractLow_	sma
count	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000(
mean	0.000000	0.000000	0.000100	-0.002500	0.000000	0.000000	0.5774
std	0.002100	0.000000	0.005700	0.004900	0.000000	0.000000	0.0058
min	-0.008300	-0.000200	-0.048500	-0.049300	0.000000	0.000000	0.5539
25%	-0.001400	-0.000000	-0.002900	-0.004700	0.000000	0.000000	0.5740
50%	-0.000100	0.000000	0.000200	-0.002100	0.000000	0.000000	0.5773
75%	0.001300	0.000000	0.003200	0.000200	0.000000	0.000000	0.5809
max	0.008300	0.000200	0.042200	0.026500	0.000200	0.000200	0.5986

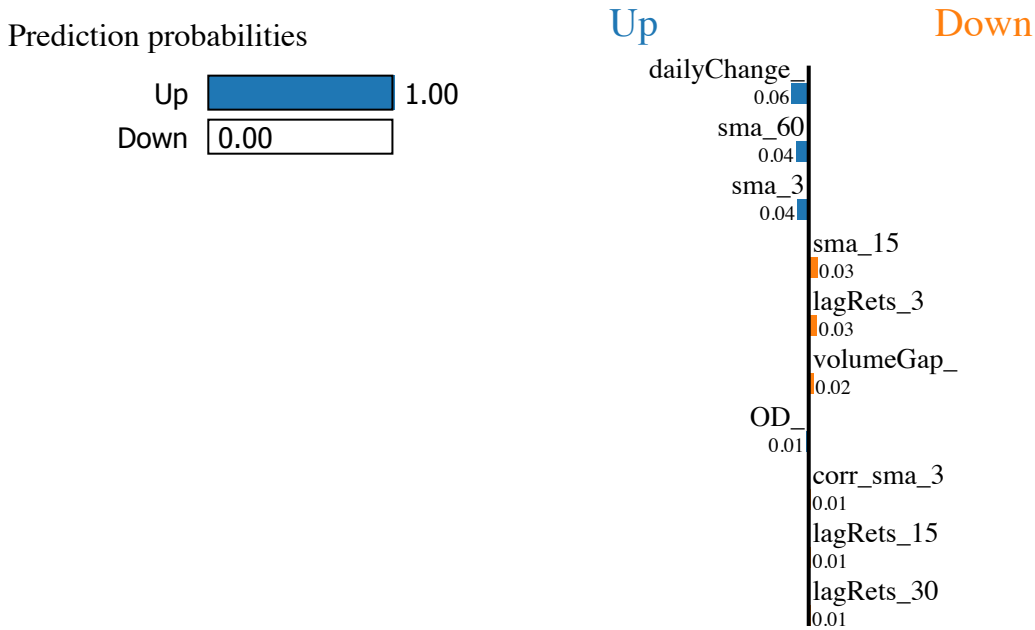
Features with Lowest Correlation

	volumeGap_	dailyChange_	OD_	lagRets_3	lagRets_15	sma_3	cor
count	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666
mean	0.000000	0.000000	0.027000	0.071200	0.338900	131.085700	(
std	0.467400	0.006600	1.358800	1.692900	3.718100	18.592500	(
min	-1.642900	-0.051400	-12.650000	-12.895500	-15.414800	101.576700	-(
25%	-0.322400	-0.003000	-0.620000	-0.888100	-1.895200	117.734200	(
50%	-0.025000	0.000000	0.040000	0.080700	0.294100	124.583300	(
75%	0.295200	0.003000	0.710000	1.024600	2.674200	142.509200	(
max	1.719000	0.053400	10.825000	11.127200	17.036500	184.213300	(

In [ ]:

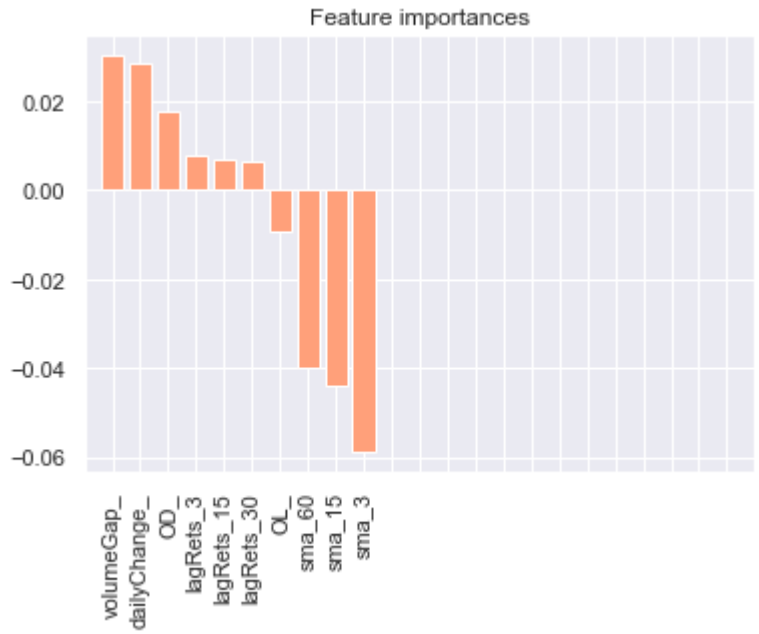
Interpretability and Feature Importance with LIME

In [13]:
```
mymodel=ret['model']
lime_gold=algo.scaling(gold,params={'method':'minmax','exclude':[]})

MLTDT.fs_localInt(lime_gold, features_set, regrnd1,
                  nclasses=['Up','Down'],
                  which_obs=11,model=mymodel)
```



Feature	Value
dailyChange_	0.55
sma_60	0.09
sma_3	0.06
sma_15	0.10
lagRets_3	0.54
volumeGap_	0.47
OD_	0.51
corr_sma_3	0.93
lagRets_15	0.41
lagRets_30	0.36

```
Explanation Score:  0.03283910444792393
```



Out[13]:

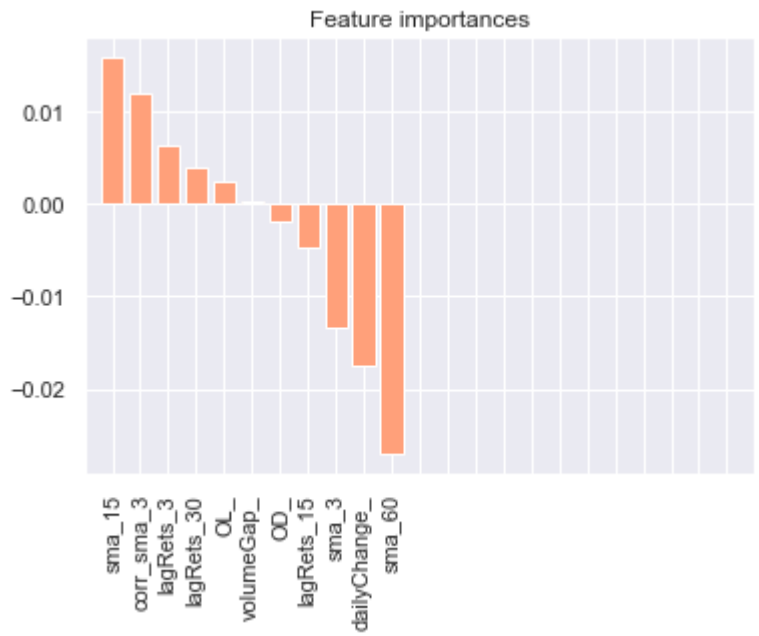
	imp
volumeGap_	0.030244
dailyChange_	0.028755
OD_	0.017600
lagRets_3	0.007685
lagRets_15	0.006781
lagRets_30	0.006467
OL_	-0.009316
sma_60	-0.039821
sma_15	-0.044048
sma_3	-0.058829

In [ ]:

```
```

In [14]:

```
MLTDT.fs_globalInt(lime_gold, features_set, regrnd1,
                   nclasses=['Up','Down'],ninsta=10, model=mymodel,
                   mtype='c', test_periods=252)
```



Out[14]:

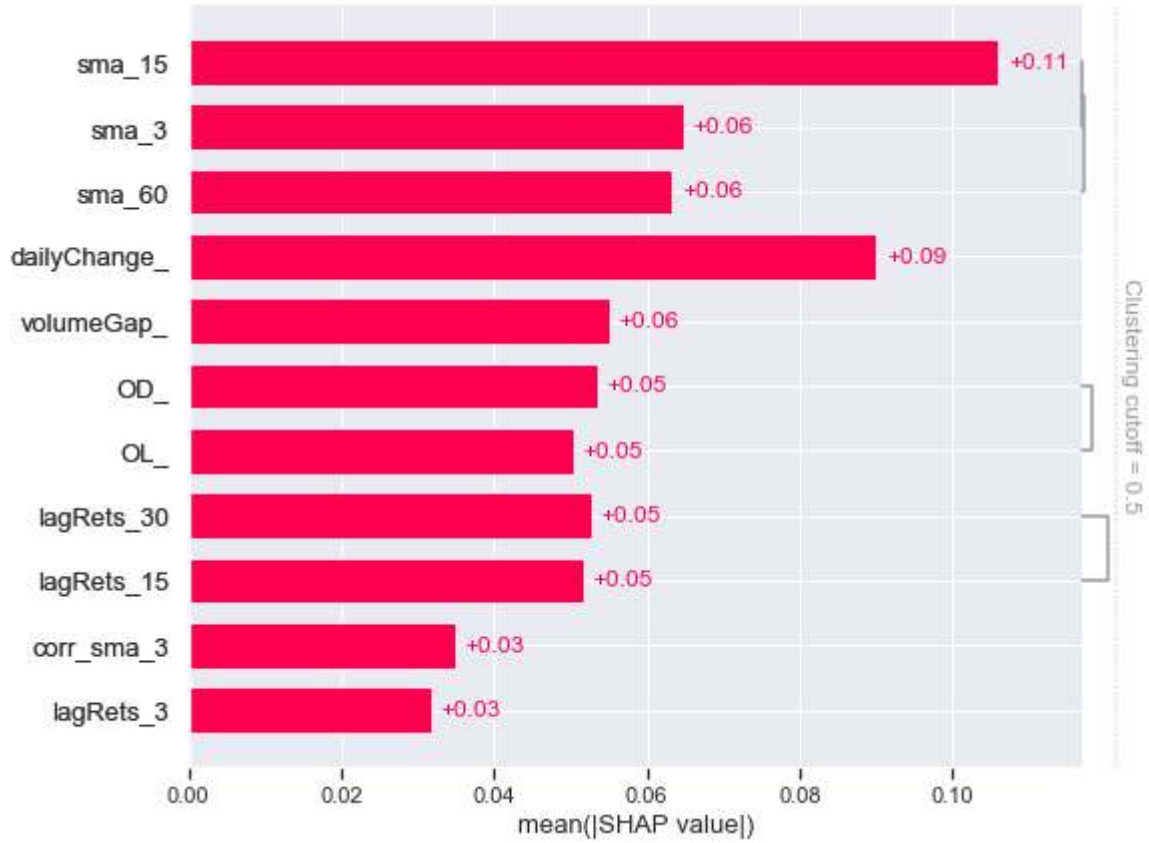
	imp
sma_15	0.015850
corr_sma_3	0.011981
lagRets_3	0.006321
lagRets_30	0.004053
OL_	0.002399
volumeGap_	0.000267
OD_	-0.001933
lagRets_15	-0.004738
sma_3	-0.013418
dailyChange_	-0.017480
sma_60	-0.027013

Interpretability and Feature Importance with SHAP

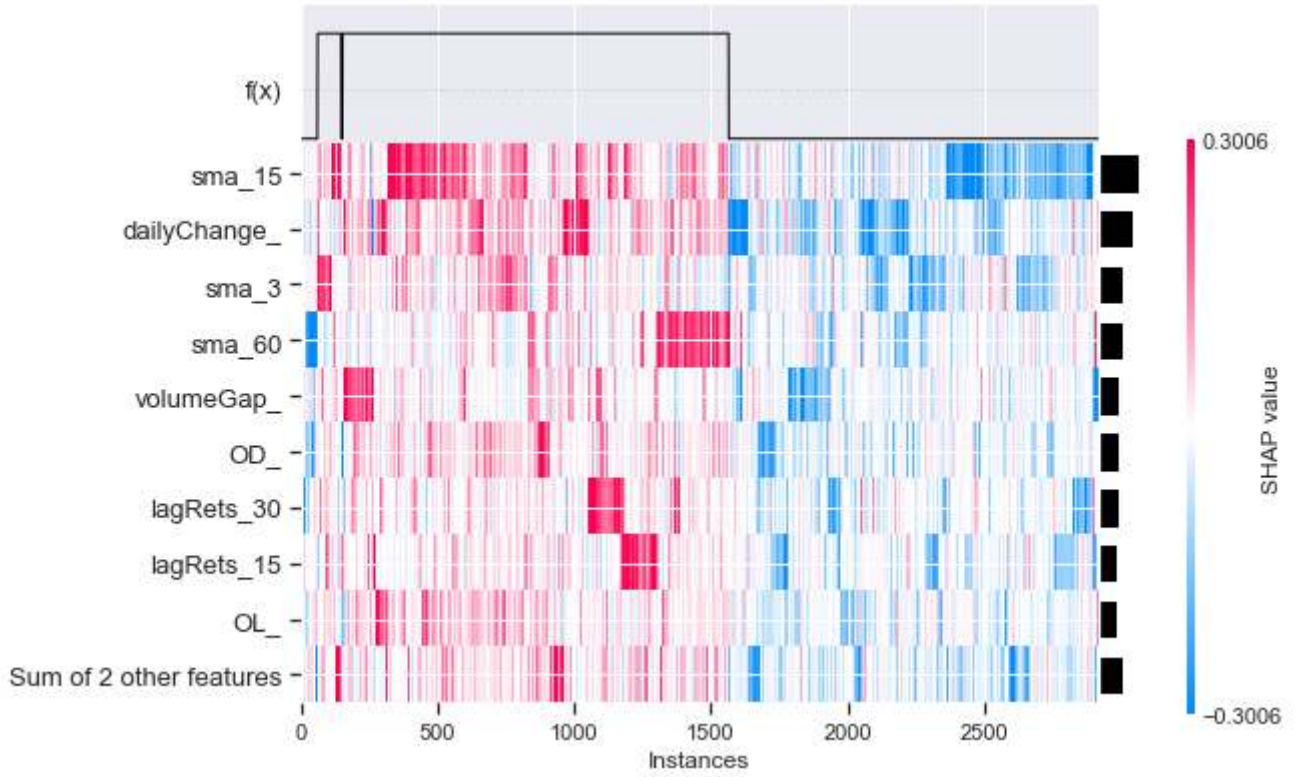
Global Partial Dependence Plots and Feature Importance

In [21]:
```
shap_values=MLTDT.fs_shapley_imp(mymodel, lime_gold[features_set])
```

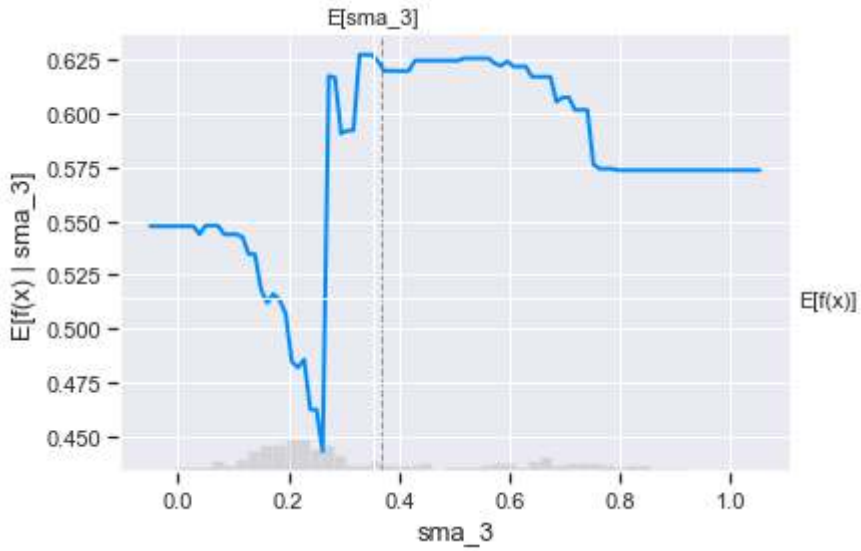
```
Partition explainer: 2919it [00:59, 41.99it/s]
```

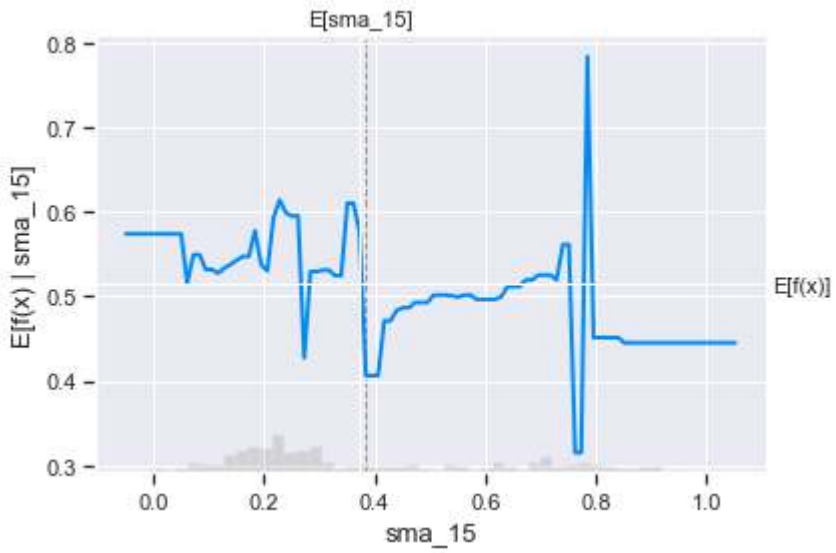


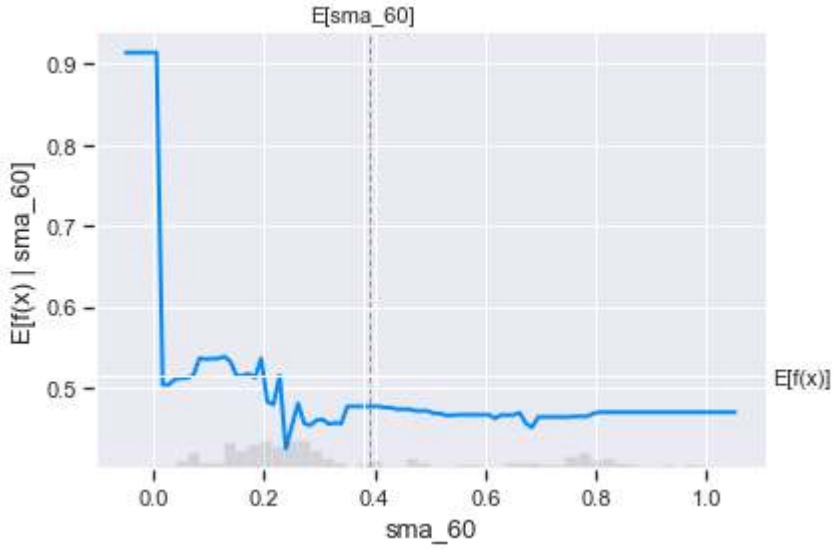
In [22]:
```
MLTDT.fs_shapley_global(shap_values)
```

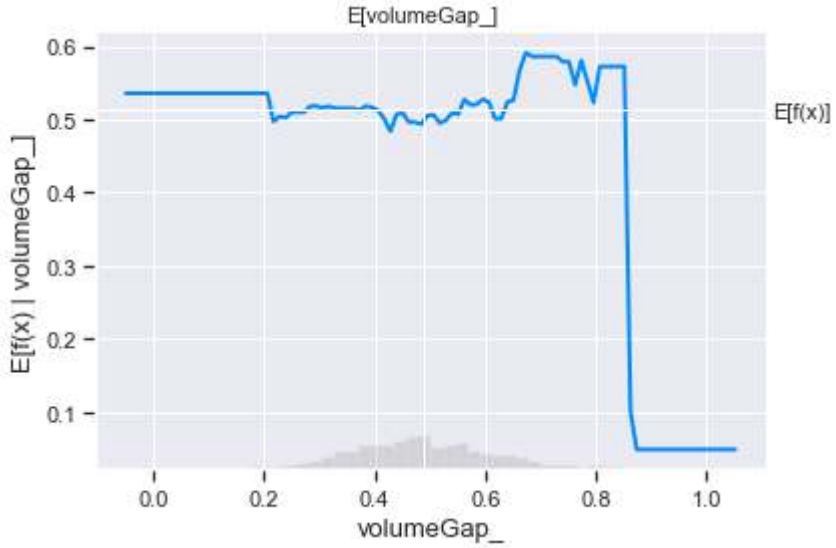


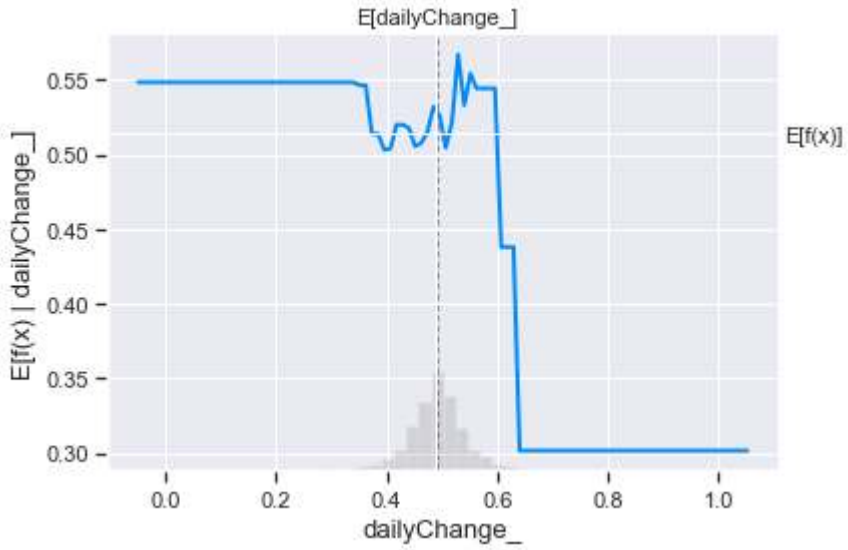
In [23]:
```
for f in features_set:
    MLTDT.fs_shapley_pdp(mymodel, lime_gold[features_set], feat_name=f);
```

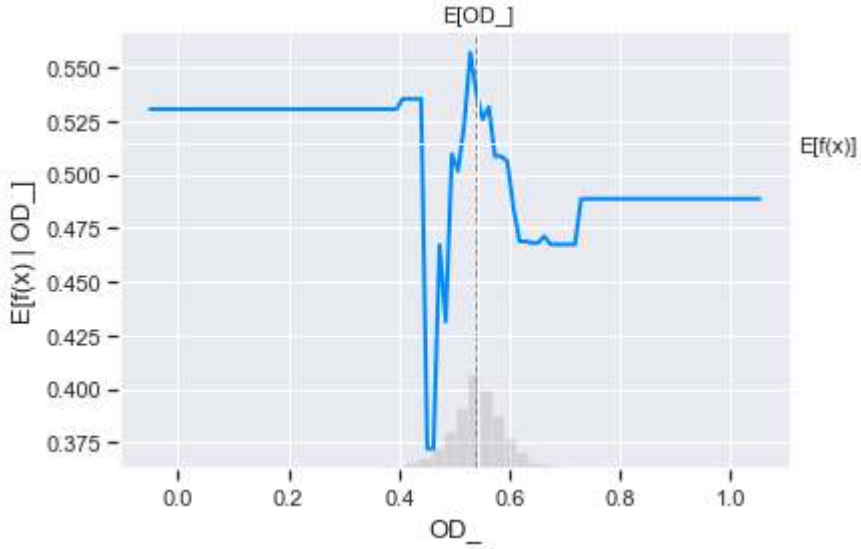


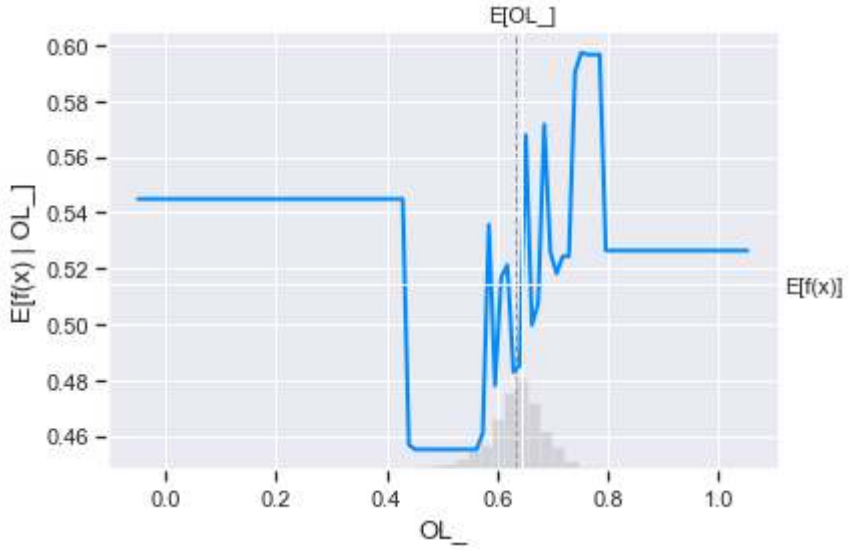


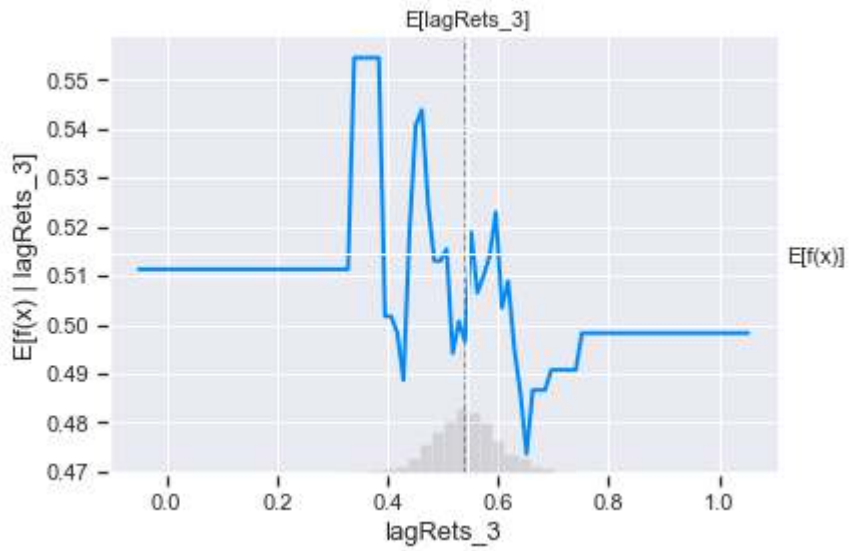

E[sma_60]
E[f(x) | sma_60]
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
sma_60
E[f(x)]

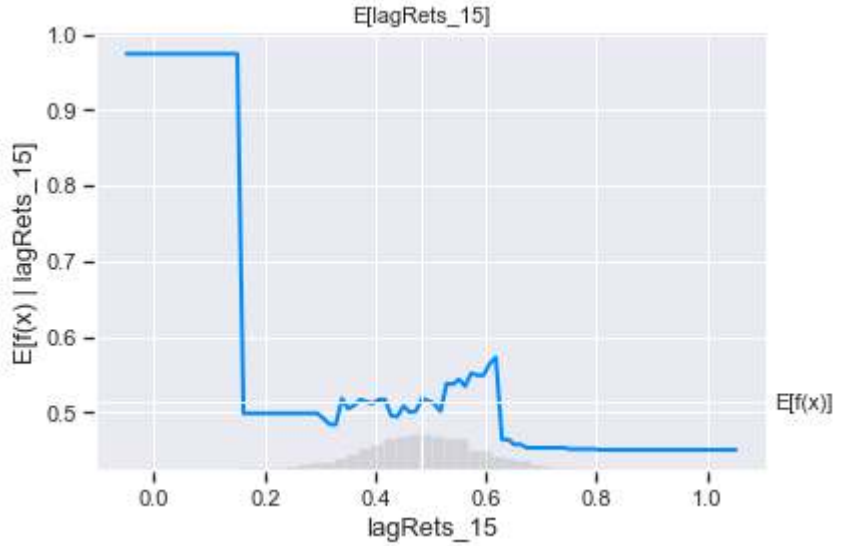

E[volumeGap_]
E[f(x) | volumeGap_]
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0
volumeGap_
E[f(x)]

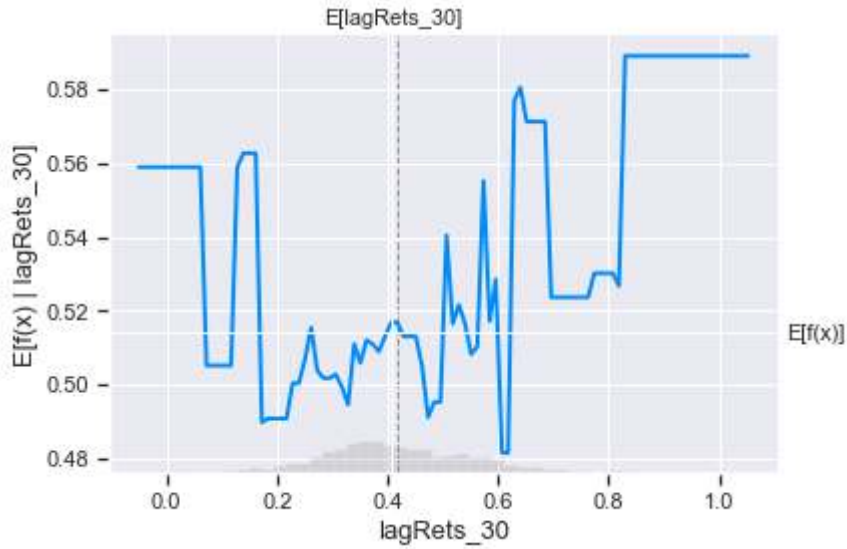

E[dailyChange_]
E[f(x) | dailyChange_]
0.55
0.50
0.45
0.40
0.35
0.30
0.0
0.2
0.4
0.6
0.8
1.0
dailyChange_
E[f(x)]

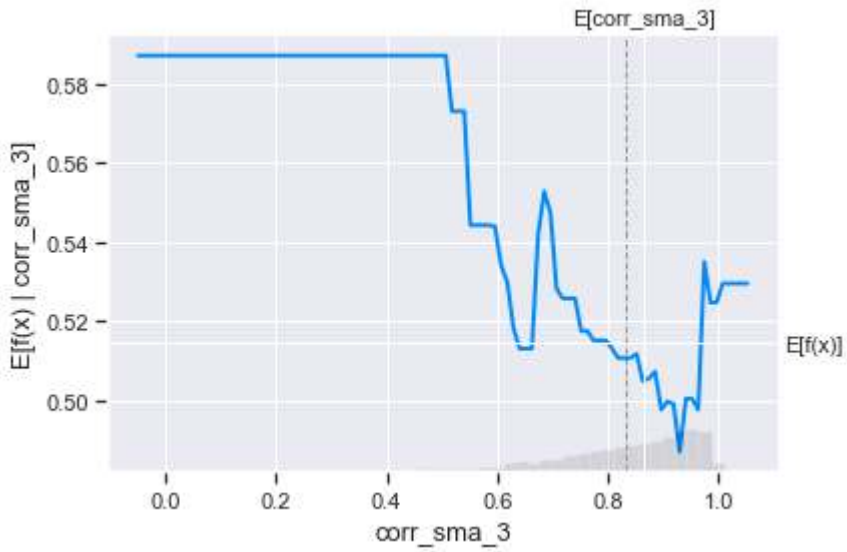

E[OD_]
E[f(x)]
E[f(x) | OD_]
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.375
0.0
0.2
0.4
0.6
0.8
1.0
OD_


E[OL_]
E[f(x)]
E[f(x) | OL_]
0.60
0.58
0.56
0.54
0.52
0.50
0.48
0.46
0.0
0.2
0.4
0.6
0.8
1.0
OL_


E[lagRets_3]
E[f(x)]
E[f(x) | lagRets_3]
0.55
0.54
0.53
0.52
0.51
0.50
0.49
0.48
0.47
0.0
0.2
0.4
0.6
0.8
1.0
lagRets_3



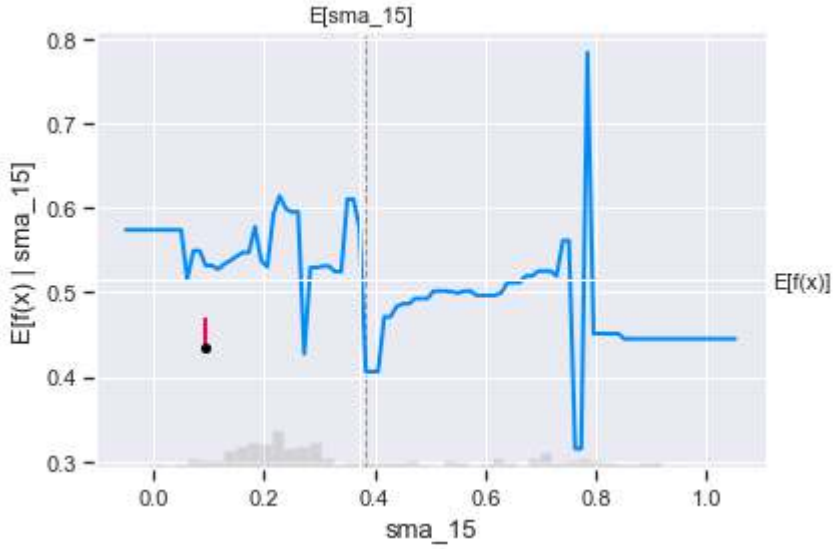




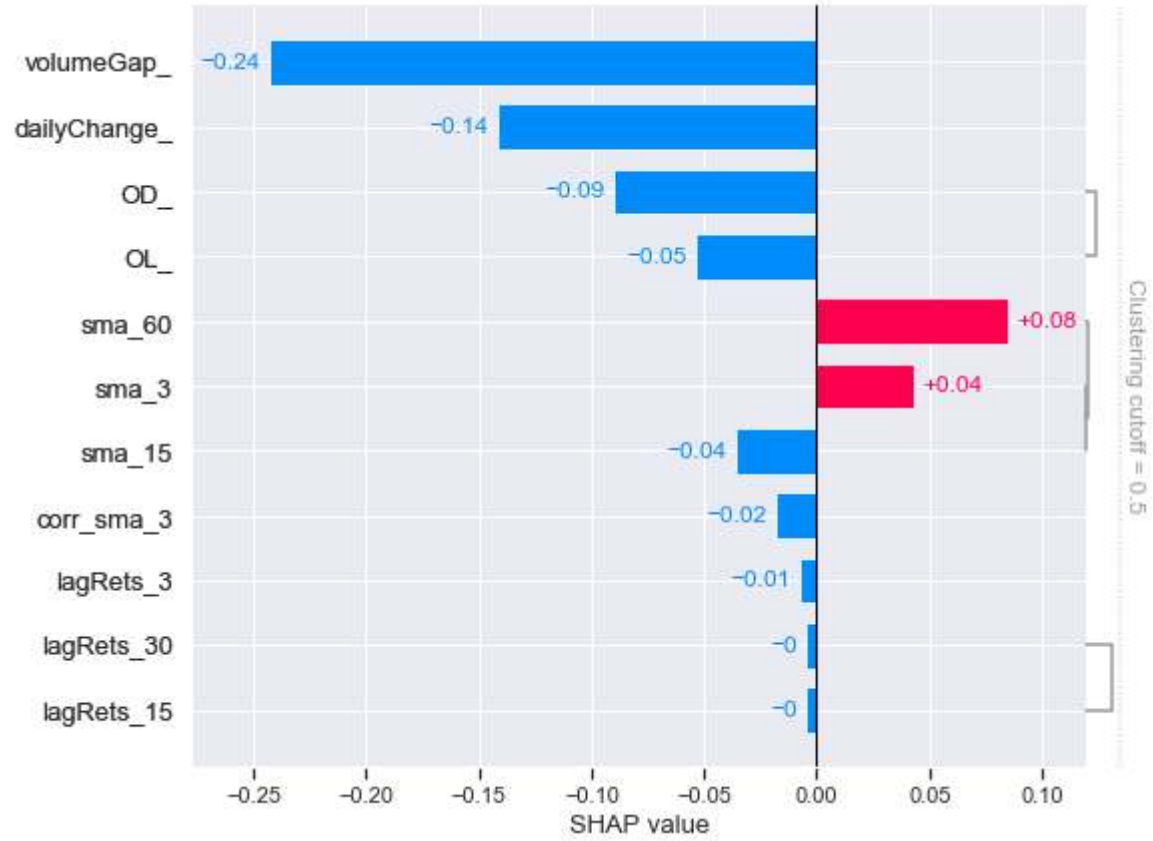
Local Partial Dependence Plots and Feature Importance

In [24]:
```
observation=11
```

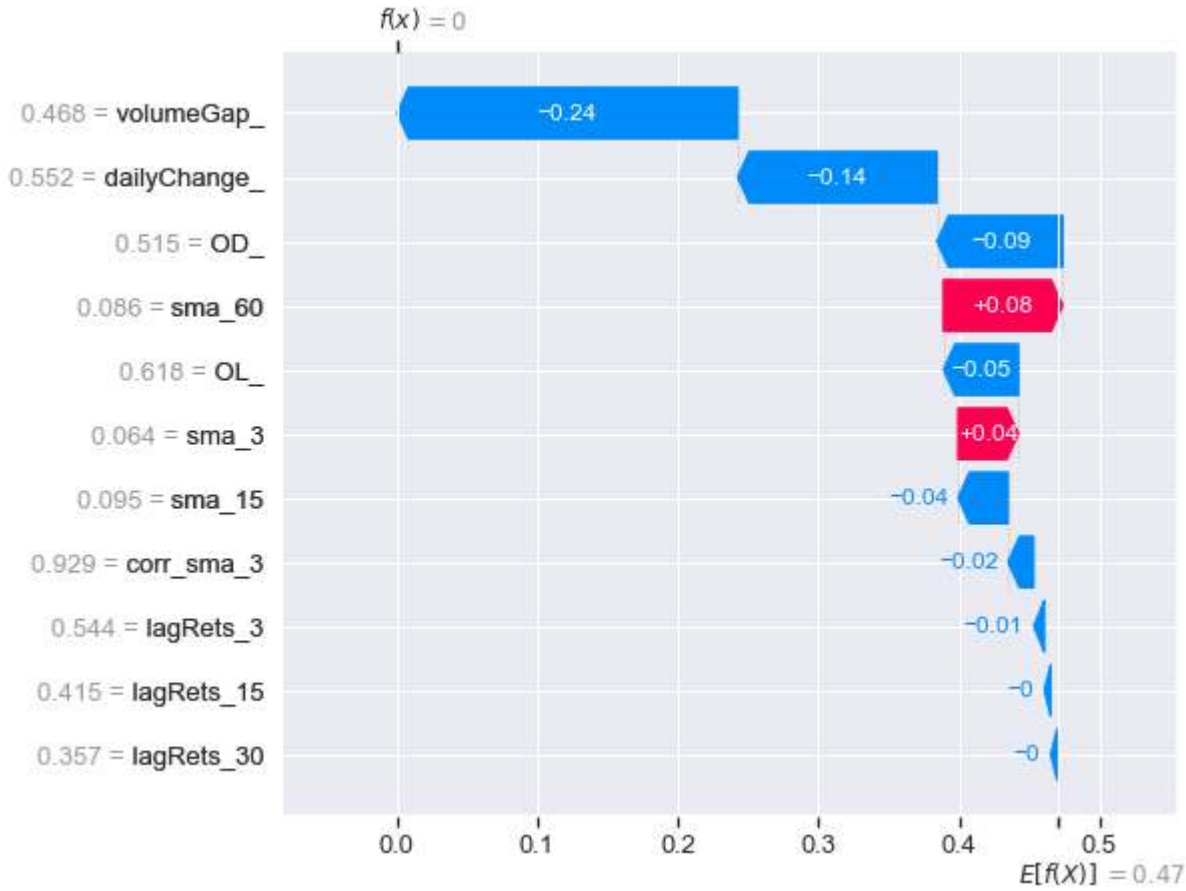
In [25]:
```
MLTDT.fs_shapley_pdp(mymodel, lime_gold[features_set], feat_name='sma_15', sample_ind
```



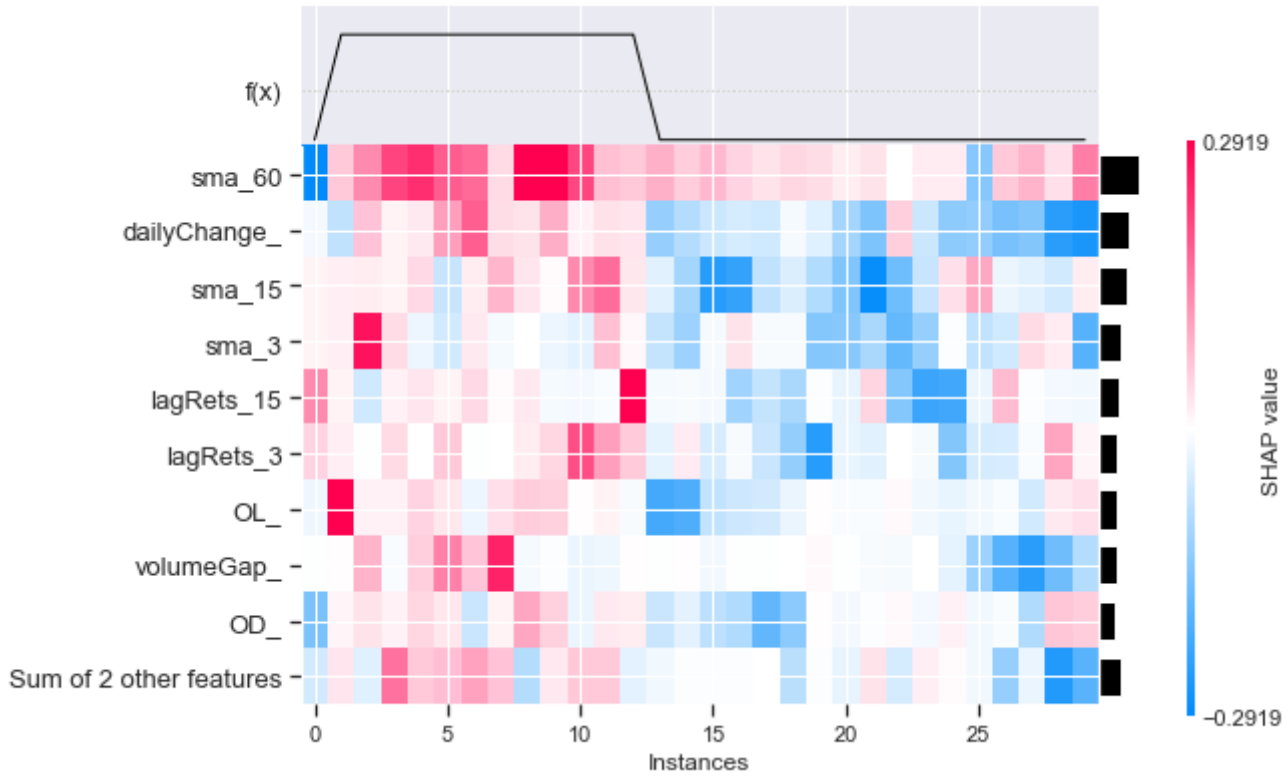
In [26]:
```
MLTDT.fs_shapley_local(shap_values, observation, ptype='bars')
```



In [27]:
```
MLTDT.fs_shapley_local(shap_values, observation, ptype='falls')
```



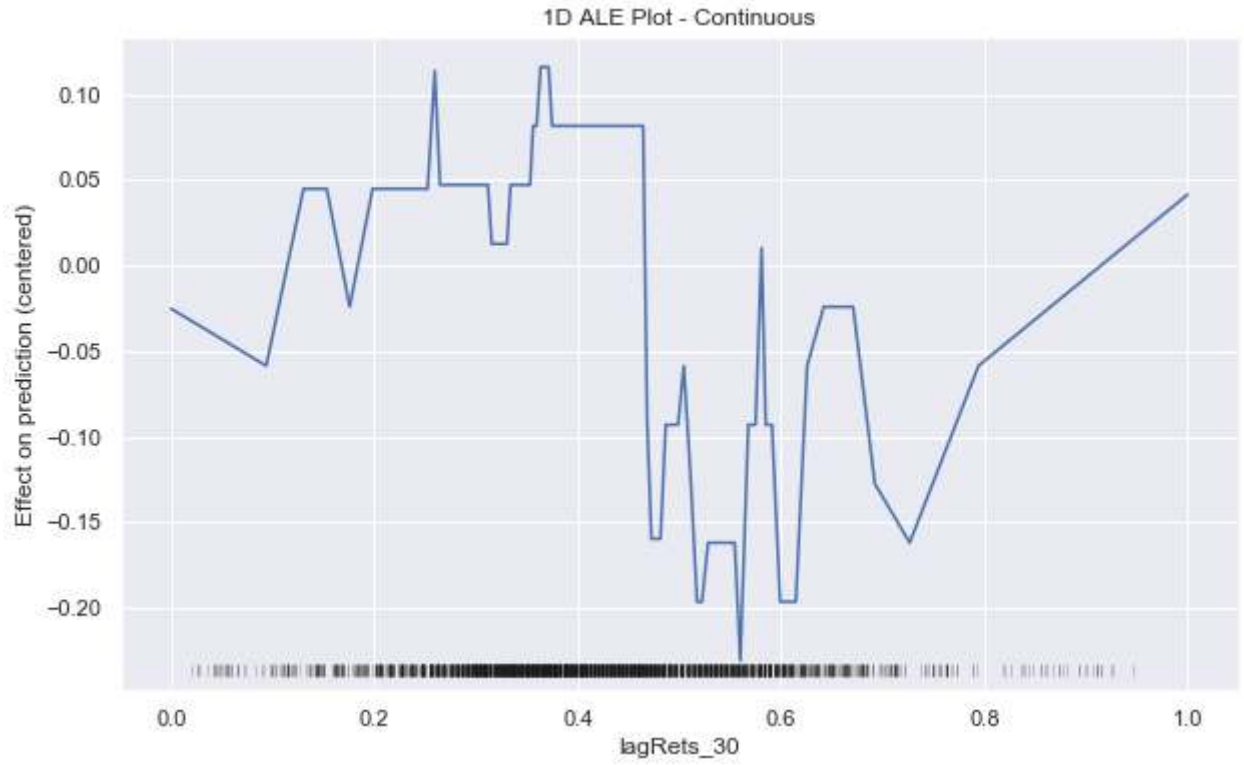
In [28]:
```
MLTDT.fs_shapley_global(shap_values,0,30)
```



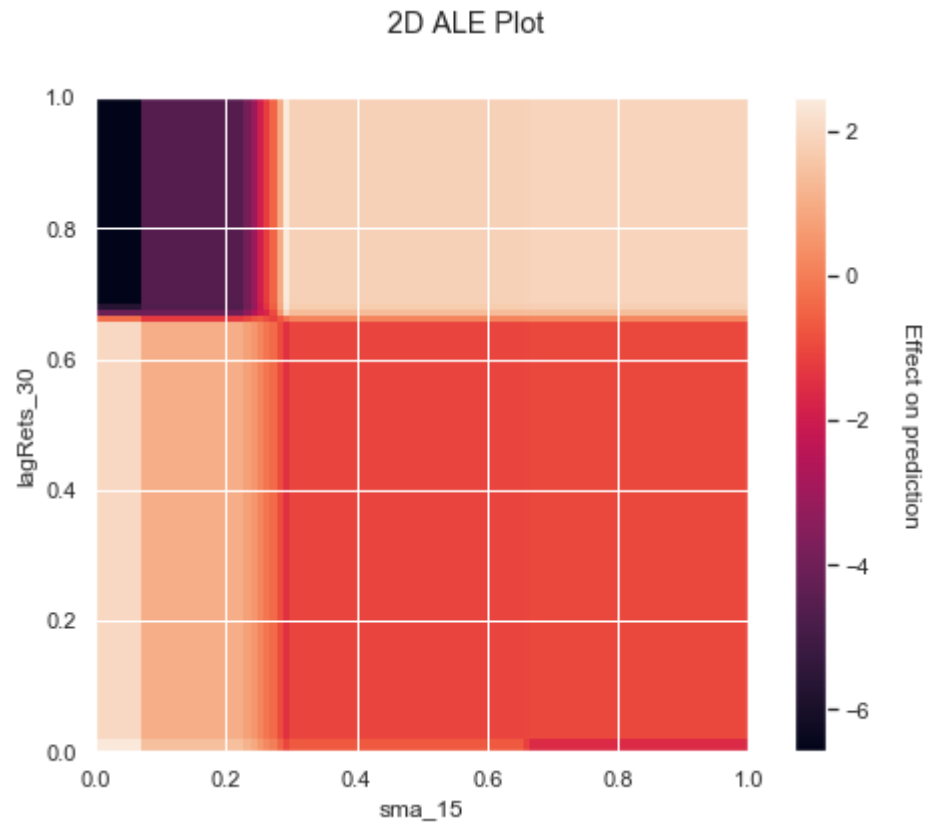
ALE Plots

In [30]:
```
ae=MLTDT.fs_alePlot(mymodel, lime_gold[features_set],"lagRets_30")
```

```
PyALE._ALE_generic:INFO: Continuous feature detected.
```

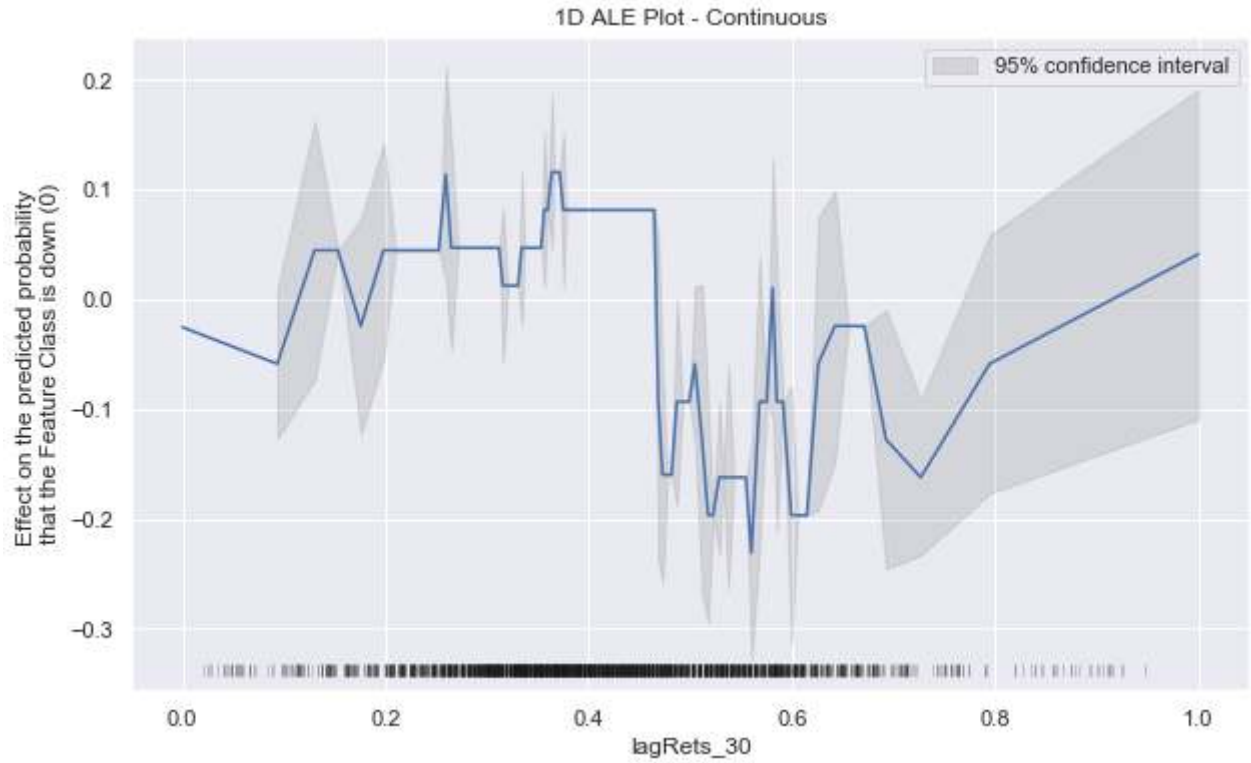


In [32]:
```
ae=MLTDT.fs_alePlot2(mymodel, lime_gold[features_set], feat_names=["lagRets_30", "sma
```



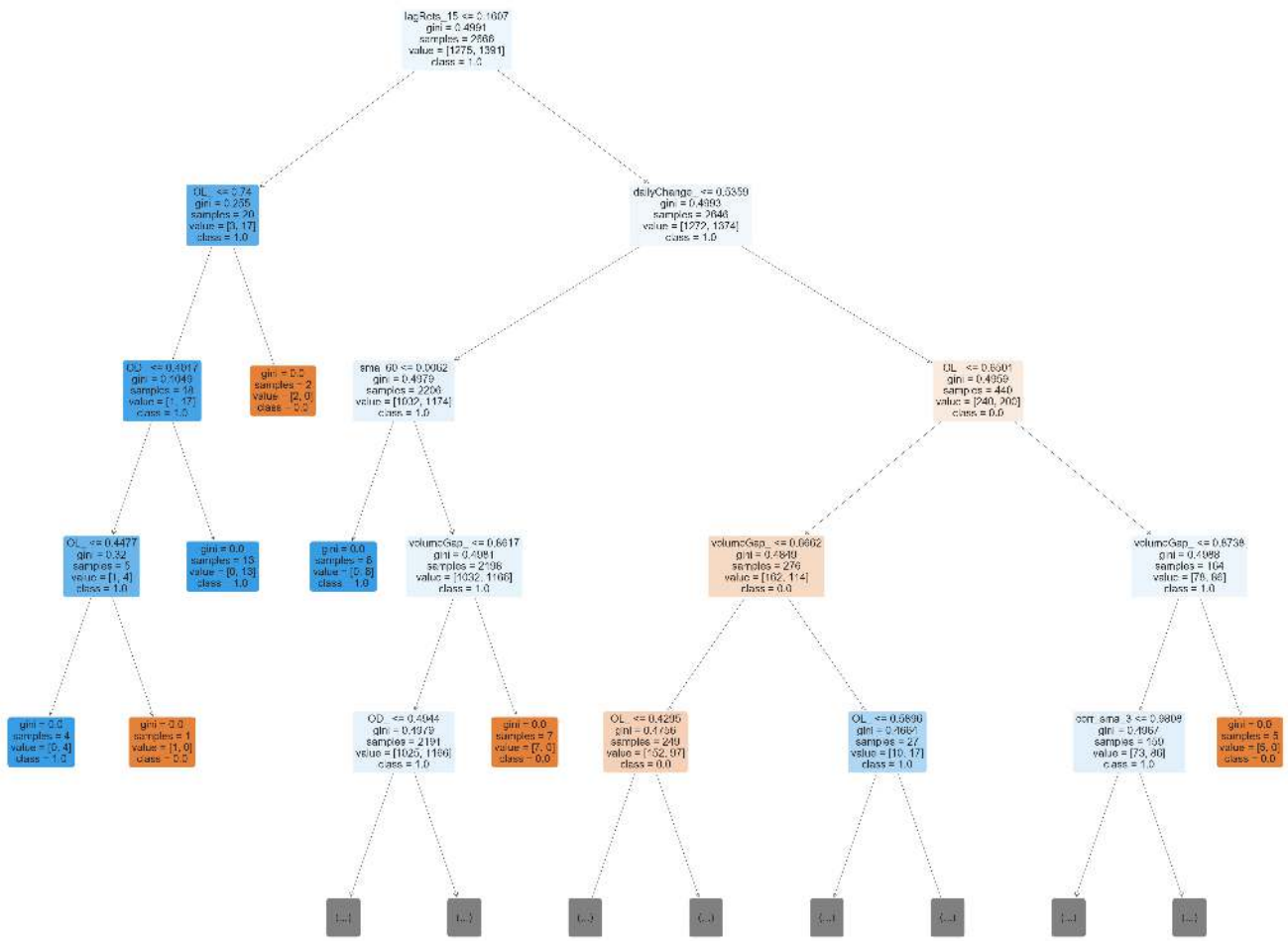
In [33]:
```
fe=MLTDT.fs_alePlotP(mymodel, lime_gold[features_set], feat_names=['lagRets_30'])
```

```
PyALE._ALE_generic:INFO: Continuous feature detected.
```



Plot of the Decision Tree

In [29]:
```
MLTDT.plotDTree(ret['model'], df, regrnd1, features_set ,max_depth=4)
```



In [ ]: